

Вінницький національний технічний університет

Міністерство освіти і науки України

Кваліфікаційна наукова

праця на правах рукопису

**МЕЛЬНИК ОЛЕКСАНДР ВАСИЛЬОВИЧ**

УДК 004.921

**ДИСЕРТАЦІЯ**

**МЕТОДИ ТА ЗАСОБИ ФОРМУВАННЯ ГРАФІЧНИХ ПРИМІТИВІВ  
НА ГЕКСАГОНАЛЬНОМУ РАСТРІ**

05.13.05 – Комп'ютерні системи та компоненти

Подається на здобуття наукового ступеня кандидата технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

\_\_\_\_\_ О. В. Мельник

Науковий керівник Романюк Олександр Никифорович доктор технічних наук, професор

Вінниця – 2024

## АНОТАЦІЯ

*Мельник О. В.* Методи та засоби формування графічних примітивів на гексагональному растрі. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.05 «Комп'ютерні системи та компоненти». – Вінницький національний технічний університет, Вінниця, 2024.

У дисертаційній роботі вирішено актуальну науково-прикладну задачу, що полягає в розробці високопродуктивних методів і засобів формування графічних примітивів на гексагональному растрі.

На основі проведеного аналізу та оцінювання сучасного стану задачі обґрунтовано необхідність розробки методів і засобів формування графічних примітивів на гексагональному растрі.

Подальшого розвитку отримав метод оцінювальної функції для формування відрізків прямих на гексагональному растрі, який відрізняється від відомих визначенням в кожному інтерполяційному такті подвійних крокових приростів, що дозволило до двох разів зменшити час лінійної інтерполяції.

Отримано співвідношення для визначення типів крокових переміщень для еліпсів і кіл залежно від ділянки формування крокової траєкторії, що спрощує методи формування примітивів за рахунок влучення надлишкових обчислень.

Вперше запропоновано метод кругової інтерполяції на гексагональному растрі, особливість якого полягає у використанні апріорно визначеному стохастичного розподілу крокових приростів залежно від ділянки формування крокової траєкторії, що дало можливість в 1,7 збільшити швидкодію інтерполяції за рахунок прогнозування найбільш вірогідної комбінації кроків.

Подальшого розвитку отримав метод антиаліазингу векторів на

гексагональному растрі, який відрізняється від відомих використанням для обчислення площі покриття пікселя додаткових оцінювальних функцій, що дозволило виконувати антиаліайзинг безпосередньо під час формування зображення крокової траєкторії та усунути етап постобробки, і, як наслідок, підвищити продуктивність формування зображення згладженої траєкторії.

Подальшого розвитку отримав метод антиаліайзингу векторів на гексагональному растрі, який відрізняється від відомих визначенням площі покриття пікселя шляхом визначення знаків оцінювальних функцій в центрах субпікселів, на які розбивається піксель, що дозволило розпаралелити процес антиаліайзингу та підвищити його продуктивність.

На основі запропонованих методів і моделей розроблено засоби та систему формування графічних примітивів на гексагональному растрі.

На основі отриманих в дисертації теоретичних положень запропоновано алгоритми та розроблено апаратні і програмні засоби формування графічних примітивів на гексагональному растрі для комп'ютерних систем візуалізації зображень.

Розроблено структурні схеми пристроїв для формування графічних примітивів на гексагональному растрі, графічний редактор для формування зображень на гексагональному растрі.

Впровадження результатів досліджень підтверджується відповідними актами. Результати досліджень використовуються на таких підприємствах «ДРЕССЛАБ Юей», ТОВ «ЗД Дженерайшн Юей»; ПП «Фотоніка плюс», кафедрі програмного забезпечення Вінницького національного технічного університету для використання у навчальному процесі.

**Ключові слова:** гексагональний растр, оцінювальна функція, лінійна інтерполяція, кругова інтерполяція, еліптична інтерполяція, крокова траєкторія, гексагон, антиаліайзинг.

## ABSTRACT

*Melnyk O.V.* Methods and means of forming graphic primitives on a hexagonal grid. – Qualifying scientific paper as a manuscript.

Thesis for PhD degree in technical sciences on specialty 05.13.05 "Computer systems and components". – Vinnytsia National Technical University. – Vinnytsia, 2024.

The thesis solves an actual scientific and applied problem, which consists in the development of high-performance methods and means of forming graphic primitives on a hexagonal grid.

In connection with the increase in the productivity of graphic tools, the issue of increasing the realism of image formation in order to more accurately reproduce real objects and processes is particularly acute. One of the cardinal directions is related to increasing the resolution capacity of screens. To increase the number of pixels on the screen, a hexagonal raster is often used instead of a square raster. Discrete representation of curves on a hexagonal grid better conveys the shape of the trajectory.

When using the hexagonal pixel model, the number of points on the screen increases, which implies the use of a larger number of pixels for the formation of primitives, which affects the speed of primitive formation. Interpolation methods on a hexagonal grid are more complex compared to using a square grid. This is explained by the different ordinate dimensions of the hexagonal pixel.

Improving the performance of methods and means of forming images on a hexagonal raster is important in the generation of dynamic graphic images and in an interactive mode, when it is assumed that the user can influence the process of image formation in real time. The high speed of creating graphic scenes is necessary for hexagonal games, which have become widespread.

At this stage of the development of computer graphics, the development of new high-performance methods and means of forming realistic images is an important task, since traditional methods and means do not always provide the necessary performance for the dynamics of images.

On the basis of the analytical analysis and evaluation of the current state of

the problem, the necessity of developing methods and means of forming graphic primitives on the hexagonal grid is substantiated.

The areas of use of the hexagonal raster, pixel models, methods of forming graphic primitives and their anti-aliasing are analyzed.

Analytical dependencies between the elements of the hexagonal raster were obtained for further use in the development of methods for forming graphic primitives.

Features of the step-by-step formation of line segments on a hexagonal grid are considered. Formulas for calculating evaluation functions are derived. The peculiarities of the formation of straight line segments for the characteristic sections into which the coordinate space is divided are considered. To increase the accuracy of the formation of the step trajectory, it is proposed to use the initial value of the evaluation function, which allows to symmetrize the error within the digital segment.

The method of the evaluation function for forming segments of straight lines on the hexagonal grid, which differs from the known ones in the definition of double step increments in each interpolation cycle, was further developed, which made it possible to reduce the time of linear interpolation up to two times.

It is proposed to use two independent evaluation functions to form step trajectories of straight line segments. One of them is used to form even points of the trajectory, and the other is used for odd points of the step trajectory.

For the first time, we obtained ratios for determining the types of step movements for circles and ellipses depending on the area of the step trajectory formation, which simplifies the methods of forming ellipses by eliminating redundant calculations.

The estimation function method for forming circles on a hexagonal grid has been modified.

For the first time, a method of circular interpolation on a hexagonal grid was proposed, the feature of which is the use of a priori defined stochastic distribution of step increments depending on the area of step trajectory formation, which made it possible to reduce the speed of interpolation by 1.7 by predicting the most likely

combination of steps.

The method of anti-aliasing vectors on a hexagonal raster was further developed, which differs from the known ones in the use of additional evaluation functions for calculating the pixel coverage area, which made it possible to perform anti-aliasing directly during the formation of the step trajectory image and eliminate the post-processing stage, and, as a result, to increase the performance of the smoothed image formation trajectories.

The method of anti-aliasing vectors on a hexagonal raster was further developed, which differs from the known ones by determining the pixel coverage area by taking into account the signs of the evaluation functions in the centers of the subpixels into which the pixel is divided, which made it possible to parallelize the anti-aliasing process and increase its performance.

A modification of the Gaussian pixel model is proposed, which does not require the use of pre-calculated cross-sectional volume values.

Approximate expressions for calculating cross-sectional volumes using only multiplication and addition operations have been obtained, which makes it possible to implement them in hardware.

A graphic editor for forming graphic images on a hexagonal raster has been developed. The editor has enough functionality to create images on a hexagonal grid.

A suitable convector has been developed to convert images from a rectangular to a hexagonal raster.

The program has developed means for forming segments of straight lines and circles on a hexagonal grid. Structural diagrams of devices for forming graphic primitives on a hexagonal grid, as well as for anti-aliasing images of step trajectories, have been developed.

The achieved characteristics and parameters of the developed tools confirm the correctness of scientific provisions and the adequacy of the proposed models and methods.

**Keywords:** hexagonal raster, evaluation function, linear interpolation, circular interpolation, elliptical interpolation, step trajectory, hexagon, antialiasing.

## СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

- [1] O. Melnik, O. Romanyuk, and V. Savratsky, *Applying of hexagonal raster in image formation. Monography*, Boston, USA, 2020.
- [2] S. O. Romanyuk, S. V. Pavlov, and O. V. Melnyk, “New method to control color intensity for antialiasing”, *Control and Communications (SIBCON)*, 2015 International Siberian Conference on 02 July 2015, DOI: 10.1109/SIBCON.2015.7147194.
- [3] О. Н. Романюк, О. В. Романюк, та О. В. Мельник, “Формування відрізків прямих на гексагональному растрі”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*, № 2, с. 69-72, 2016.
- [4] О. Н. Романюк, та О. В. Мельник, “Особливості використання гексагонального растра при побудові пристроїв відображення”, *Міжнародний науково-технічний журнал «Вимірювальна та обчислювальна техніка в технологічних процесах»*, Хмельницький, №3 (56), с. 105-109, 2016.
- [5] О. Н. Романюк, О. В. Романюк, та О. В. Мельник, “Реалізація кругової інтерполяції при використанні гексагонального растру”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*, № 1(24), с. 53-58, 2017.
- [6] О. Н. Романюк, І. В. Абрамчук, та О. В. Мельник, “Визначення типів крокових приростів для побудови кола на гексагональному растрі”, *Вісник Хмельницького національного університету. Серія: Технічні науки*, Хмельницький, ХНУ, №3(249), с. 172-176, 2017.
- [7] О. Н. Романюк, О. В. Мельник, І. В. Абрамчук, та Н. С. Костюкова, “Особливості формування еліпсів на гексагональному растрі”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*, № 1(26), с. 86-90, 2018.

[8] О. Н. Романюк, О. В. Мельник, та С. І. Вяткін, “Класифікація методів антиаліаїзінгу”, *Вісник Херсонського національного технічного університету*, Херсон, ХНТУ, №3(50), с. 154-159, 2014 р.

[9] O. N. Romanyuk, A. V. Melnik, A. P. Goncharuk, and Y. L. Lyashenko, “Effective Models for the Specular Color Constituent Computing”, *Journal of Computer Science and Engineering*, Volume 2, Issue 2, pp. 25-29, august 2010.

[10] O. Romanyuk, S. Pavlov, O. Melnyk, S. Romanyuk, A. Smolarz, and M. Bazarova, “Method of anti-aliasing with the use of the new pixel model”, *Optical Fibers and Their Applications*, Lublin, Poland, pp. 274-278, 2015. <https://doi.org/10.1117/12.2229013>.

[11] О. Н. Романюк, М. С. Курінний, О. В. Мельник, та Н. С. Костюкова, “Високопродуктивна конусна модель пікселя для антиаліаїзінгу відрізків прямих”, *Наукові праці Донецького національного технічного університету. Серія «Інформатика, кібернетика та обчислювальна техніка»*, Випуск 14(188), Донецьк, с. 216-220, 2011.

[12] О. Н. Романюк, М. С. Курінний, О. В. Мельник, та С. В. Олійник, “Кругова модель пікселя для задач антиаліаїзінгу”, *Наукові праці Донецького національного технічного університету. Серія «Інформатика, кібернетика та обчислювальна техніка»*, Випуск 15(203), Донецьк, с. 90-94, 2012.

[13] О. Н. Романюк, М. С. Курінний, та О. В. Мельник, “Антиаліаїзінг зображення відрізків прямих з використанням нової моделі пікселя”, *Оптико-електронні інформаційно-енергетичні технології. Міжнародний науково-технічний журнал. №2(20)*, с. 30-35, Вінниця, ВНТУ, 2010.

[14] О. Н. Романюк, О. В. Романюк, та О. В. Мельник, “Метод антиаліаїзінгу зображень відрізків прямих з використанням додаткових оцінювальних функцій”, *Міжнародний науково-технічний журнал «Вимірвальна та обчислювальна техніка в технологічних процесах»*, Хмельницький, ХНУ, №2 (47), с. 210-214, 2014.



[15] О. Н. Романюк, О. В. Мельник, І. В. Абрамчук, та О. О. Дудник, “Модифікація гаусівської моделі пікселя для задач антиаліайзингу”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*. № 1, с. 84-88, 2015.

[16] О. Н. Романюк, С. О. Романюк, О. В. Мельник, та Д. А. Озерчук, “Особливості формування еліпсів, повернутих на заданий кут, на гексагональному растрі”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*, № 2(31), с. 23-28, 2020.

[17] О. Н. Романюк, О. В. Мельник, С. В. Котлик, С. О. Романюк, та Р. Ю. Чехместрук, “Методи підвищення продуктивності формування векторів на гексагональному растрі”, *Автоматизація технологічних і бізнес-процесів*, Volume 14, Issue 3, с. 27-36, 2022. DOI: 10.15673/atbp.v14i3.2350

[18] О. Н. Романюк, О. В. Мельник, та В. А. Шмалюх, “Метод прискореної кругової інтерполяції на гексагональному растрі”, *Вісник ВПІ*, № 2, с. 81–88, 2023. doi.org/10.31649/1997-9266-2023-167-2-81-88

[19] О. Н. Романюк, О. В. Мельник, С. О. Романюк, Т. І. Коробейнікова, та О. П. Прозор, “Антиаліайзинг зображення крокової траєкторії відрізка прямої на гексагональному растрі”, *Modern engineering and innovative technologies*, Issue 22, Part 1, с. 113-121, 2022. DOI: 10.30890/2567-5273.2022-22-01-043

[20] С. О. Романюк, О. Н. Романюк, С. В. Павлов, та О. В. Мельник, “Модель для відтворення спекулярної складової кольору в засобах комп’ютерної графіки”, *Інформаційні технології та комп’ютерна інженерія*, ВНТУ, №3(34), с. 50-57, 2015.

[21] О. Н. Романюк, Ю. О. Панфілова, та О. В. Мельник, “Використання гексагонів у комп’ютерних іграх” на *XLIX науково-технічній конференції підрозділів ВНТУ*, Вінниця, 27-28 квітня 2020 р. [Електронний ресурс]. Доступно:<https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/view/10431>. Дата звертання: 01.11.2023.

[22] С. О. Романюк, О. В. Мельник, та І. Г. Бабій, “Антиаліайзинг дуги кола з використанням модифікованої оцінювальної функції”, на *II Всеукраїнській науково-технічній конференції студентів, аспірантів та молодих вчених “Інформаційні управляючі системи та комп’ютерний моніторинг”*, Донецьк, ДонНТУ, 2011, с. 37-41.

[23] О. Н. Романюк, та О. В. Мельник, “Морфологічний антиаліайзинг для гексагонального растру”, на *III Міжнародній науково-практичній конференції «Актуальні проблеми гуманітарних та природничих наук»*, Київ, 28-29 жовтня 2016, с. 112-115.

[24] О. Н. Романюк, О. В. Мельник, та О. В. Стукач, “Моделювання гексагонального растра на квадратному растрі”, на *VII Міжнародній науково-технічній конференції «Моделювання і комп’ютерна графіка»*, Київ, 18-24 вересня 2017, с.18-24.

[25] О. Н. Романюк, О. В. Мельник, та Ю. О. Панфілова, “Використання гексагонального растру в комп’ютерних іграх”, на *XII Міжнародній науково-технічній конференції «Інформаційні комп’ютерні технології – 2021 (ІКТ-2021)»*, Житомир: Житомирська політехніка, 1-3 квітня 2021, с. 5-6.

[26] О. Н. Романюк, О. В. Мельник, та Ю. О. Панфілова, “Формування параболи на гексагональному растрі”, на *Міжнародній науково-практичній Інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ»*, Суми-Вінниця, 9-10 листопада 2021, с. 196-198.

[27] О. Н. Романюк, О. В. Романюк, О. В. Мельник, та Р. Ю. Чехмestрук, “Суперсемплінг зображень, сформованих на гексагональному растрі”, in *Modern research in world science Proceedings of the 3rd International scientific and practical conference. SPC “Sci-conf.com.ua”*, Lviv, Ukraine, 2022, pp. 517-522.

[28] О. Н. Романюк, О. В. Мельник, та Л. Г. Коваль, “Використання гексагональних комірок у видавничій справі”, на *V Міжнародній науковій*

конференції «Інформація, комунікація та управління знаннями в глобалізованому світі», Київ, 23-24 червня 2022, с. 45-47.

[29] О. Н. Романюк, О. В. Мельник, та В. А. Шмалюх, “Розробка застосунку для формування відрізків на гексагональному растрі”, на *Всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи»*, Вінниця, 2022, с 1-7.

[30] О. Н. Романюк, М. Д. Захарчук, О. В. Мельник, О. В. Романюк, та С. В. Котлик “Аналіз гексагональних ігор”, на *II Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів “Комп’ютерні ігри та мультимедіа як інноваційний підхід до комунікації”*, Одеса, 29-30 вересня 2022, с. 139-143.

[31] О. Н. Романюк, М. В. Козубенко, О. В. Мельник, та С. В. Котлик, “Використання гексагонального растру в картографії», на *XV Міжнародній науково-практичній конференції Інформаційні технології і автоматизація – 2022*, Одеса, 20-21 жовтня 2022, с. 30-33.

[32] О. Н. Романюк, О. В. Романюк, та О. В. Мельник, “Програмна реалізація графічного редактора на гексагональному растрі”, in *The 8th International scientific and practical conference “Modern research in world science”*, (October 29-31, 2022) SPC “Sci-conf.com.ua”, Lviv, Ukraine, 2022, pp. 389-392.

[33] О. Н. Романюк, О. В. Мельник, Р. Ю. Чехместрук, та В. А. Шмалюх, “Програмний модуль для формування кіл на гексагональному растрі”, in *The 12th International scientific and practical conference “Modern research in world science”*, (February 26-28, 2023) SPC “Sci-conf.com.ua”, Lviv, Ukraine, 2023, pp. 326-332.

[34] О. В. Мельник, “Формування кола незалежними оцінювальними функціями”, на *XXXI Міжнародній науково-практичній конференції MicroCAD-2023*, Харків, 2023, с. 1187-1188.

[35] О. В. Мельник, “Стохастичний розподіл комбінованих крокових приростів при формуванні кіл на гексагональному растрі” на *Scientific*

*Research and Innovation: Proceedings of the 2nd International Scientific and Practical Internet Conference*, Дніпро, 2023, с. 247-249.

[36] О. Н. Романюк, О. В. Мельник, Р. Ю. Чехмestruc та С. О. Романюк, “Основні співвідношення гексагонального растру”, на *VII Міжнародній науково-практичній конференції Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі*, Київ, 2022, с. 59-61.

[37] О. Н. Романюк, В. А. Шмалюх, та О. В. Мельник, “Дисплейні технології формування гексагонального растру в сучасних пристроях відображення інформації”, in *Global Society in Formation of New Security System and World Order: Proceedings of the 2nd International Scientific and Practical Internet Conference*, July 27-28, 2023, Dnipro, Ukraine, с. 341-344.

[38] О. Н. Романюк, М. С. Курінний, О. В. Мельник та Ю. Л. Ляшенко, “Комп’ютерна програма «Програма для антиаліайзингу кривих другого порядку заданих рівнянням у загальній формі з використанням нового методу мультисемплінгу”, *Свідоцтво про реєстрацію авторського права на твір* №35655 від 11.11.2010.

[39] О. Н. Романюк, О. В. Мельник та В. М. Чорний, “Комп’ютерна програма “Формування відрізків прямих з використанням для антиаліайзингу моделі пікселя у вигляді додекагона”, *Свідоцтво про реєстрацію авторського права на твір* №57041 від 17.10.2014.

[40] О. В. Мельник, “Комп’ютерна програма для формування відрізків прямих на гексагональному растрі”, *Свідоцтво про реєстрацію авторського права на твір* №74884 від 21.11.2017.

[41] О. Н. Романюк, О. В. Мельник та О. О. Дудник, “Комп’ютерна програма для формування дуг кіл на гексагональному растрі”, *Свідоцтво про реєстрацію авторського права на твір* №74925 від 22.11.2017.

## ЗМІСТ

ВСТУП.....	15
<b>РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ І ЗАСОБІВ ФОРМУВАННЯ</b>	
<b>ГРАФІЧНИХ ЗОБРАЖЕНЬ НА ГЕКСАГОНАЛЬНОМУ РАСТРІ.....</b>	<b>23</b>
1.1 Аналіз моделей пікселів.....	23
1.2 Обробка зображень на гексагональному растрі.....	36
1.3 Використання гексагональних елементів у системах візуалізації	41
1.4 Аналіз методів побудови графічних примітивів.....	47
1.5 Висновки до розділу 1.....	54
<b>РОЗДІЛ 2 МЕТОДИ ФОРМУВАННЯ ВЕКТОРІВ НА</b>	
<b>ГЕКСАГОНАЛЬНОМУ РАСТРІ.....</b>	<b>56</b>
2.1 Особливості формування векторів на гексагональному растрі...	56
2.2 Формування відрізків прямих на гексагональному растрі.....	61
2.3 Формування відрізків прямих комбінованими приростами на гексагональному растрі.....	65
2.4 Висновки до розділу 2.....	72
<b>РОЗДІЛ 3 МЕТОДИ ФОРМУВАННЯ КІЛ НА</b>	
<b>ГЕКСАГОНАЛЬНОМУ РАСТРІ.....</b>	<b>74</b>
3.1 Визначення типів крокових приростів при побудові кола на гексагональному растрі.....	74
3.2 Формування кіл за методом оцінювальної функції.....	80
3.3 Метод прискореного формування кіл на гексагональному растрі.....	90
3.4 Особливості формування еліпсів на гексагональному растрі.....	98
3.5 Висновки до розділу 3.....	107
<b>РОЗДІЛ 4 МЕТОДИ АНТИАЛІАЙЗИНГУ ЗОБРАЖЕНЬ</b>	
<b>ГРАФІЧНИХ ПРИМІТИВІВ.....</b>	<b>109</b>
4.1 Суперсемплінг графічних зображень, які використовують гексагональну модель пікселя.....	109

4.2	Метод антиаліайзингу з визначенням оцінювальних функцій для субпікселів.....	114
4.3	Метод антиаліайзингу зображення з використанням додаткових оцінювальних функцій.....	120
4.4	Використання гаусівської моделі пікселя для задач антиаліайзингу.....	126
4.5	Висновки до розділу 4.....	134
<b>РОЗДІЛ 5 ПРАКТИЧНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНИХ МЕТОДІВ ФОРМУВАННЯ ЗОБРАЖЕНЬ НА ГЕКСАГОНАЛЬНОМУ РАСТРІ...</b>		
5.1	Графічний редактор на гексагональному растрі.....	135
5.2	Конвертація зображення для гексагонального растру.....	138
5.3	Програмні засоби для формування примітивів.....	141
5.4	Апаратні засоби для формування примітивів.....	148
5.5	Висновки до розділу 5.....	155
<b>ВИСНОВКИ.....</b>		156
<b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....</b>		158
<b>ДОДАТКИ.....</b>		172
Додаток А Акти впровадження.....		173
Додаток Б Список публікацій за темою дисертації.....		177
Додаток В Лістинги програм.....		183

## ВСТУП

**Обґрунтування вибору теми дослідження.** Графічні зображення отримали значне поширення в багатьох галузях діяльності людини завдяки їх високій інформативності та наочності. При формуванні таких зображень необхідно забезпечити високу реалістичність відтворення об'єктів і процесів за прийнятний для конкретної задачі час.

При формуванні графічних зображень гостро стоїть питання підвищення реалістичності формування зображень з метою більш реалістичного відтворення реальних об'єктів і процесів. Для збільшення роздільної здатності екранів часто замість квадратного растру використовують гексагональний растр. При використанні гексагонального растру приблизно на 13% збільшується роздільна здатність екранів, забезпечується узгоджене з'єднання сусідніх елементів і краща кутова корекція порівняно з прямолінійною квадратною сіткою.

Важливим фактором, який заслуговує на дослідження гексагональної решітки для подання зображення, є її подібність до зорової системи людини. Очна ямка очей людини складається з гексагональних колбочок, тому при поданні зображення в гексагональному домені комп'ютерний зір буде наближений до зору людини.

Використання гексагонального растру дозволяє збільшити точність виявлення та відтворення контурів зображень.

Широке використання гексагональних елементів у приладах з зарядковим зв'язком дає можливість розширити динамічний діапазон і роздільну здатність формування зображень. Використання гексагональних елементів забезпечує ефективні схеми зберігання для зображення з використанням деревоподібних і пірамідальних структур даних. При відтворенні зображень забезпечується більший спектр передачі кольору та дозволяє поєднати фільтри CMY і RGB.

У гексагональному растрі центр пікселя знаходяться на однаковій відстані від центрів його сосудів, що спрощує семантичну обробку

зображень. Розроблено процедури для швидкого розпізнавання образів тривимірних об'єктів за допомогою паралельної обробки на гексагональній сітці.

Використання гексагональних елементів забезпечує ефективні схеми зберігання для зображення з використанням деревоподібних і пірамідальних структур даних.

Якщо раніше основною перешкодою до широко впровадження гексагонального растру була відсутність комерційно доступних пристроїв, то сьогодні ситуація кардинально змінилася. Так, зокрема, широкого поширення отримала інноваційна технологія флексографічного друку. Використання гексагональних комірок забезпечує значне збільшення оптичної щільності та інтенсивності кольору відповідно до стандартних фарб. Розроблено та масово випускаються прилади з використання ПЗС на основі гексагональних структур.

Останні роки характеризуються особливою увагою до використання гексагональних структур для виготовлення екранів. Технологія  $\mu$ -LED з використанням гексагональних елементів вважається наступним поколінням технології відображення, яка має характеристики високої мініатюризації, інтеграції, а також високу яскравість, контрастність, високу швидкість реагування індикаторних елементів, а також тривалий термін служби. Важлива властивість нової технології полягає в тому, що пікселі можуть залишатися підсвіченими протягом усього кадру. Інтенсивні дослідження проводяться щодо використання гексагональних структур для запам'ятовування інформації на основі гексагонального нітриду бору (h-BN).

Таким чином, гексагональне подання інформації має ряд переваг, інтенсивно впроваджується для візуалізації даних, що передбачає необхідність розробки відповідних методів і засобів.

При використанні гексагональної моделі пікселя збільшується кількість точок на екрані, що передбачає використання для формування примітивів більшої кількості пікселів. Це зменшує швидкодію формування примітивів. Методи інтерполяції на гексагональному растрі більш складні порівняно з



використанням квадратного растру. Це пояснюється різними ординатними розмірами гексагонального пікселя.

Підвищення продуктивності методів і засобів формування зображень на гексагональному растрі необхідно для генерації динамічних графічних зображень в інтерактивному режимі, коли користувач у реальному часі впливає на процес формування зображення. Висока швидкодія формування графічних сцен необхідна для гексагональних ігор, які отримали широку популярність завдяки збільшенню напрямків переміщення та кращому інтелектуальному сприйнятті зображення.

Таким чином, розробка високопродуктивних методів і засобів формування графічних примітивів на гексагональному растрі є актуальною задачею, яка має важливе практичне значення.

**Зв'язок роботи з науковими програмами, планами, темами.** Результати досліджень були використані для розробки методів і засобів формування графічних зображень на гексагональному растрі при реалізації держбюджетної науково-дослідної роботи: № 05/561-4 «Апаратне програмне забезпечення інформаційних технологій» (номер державної реєстрації 0118U100181).

**Мета і завдання дослідження.** Метою роботи є підвищення продуктивності та реалістичності формування графічних зображень на гексагональному растрі. Основними задачами дослідження є:

- провести аналіз існуючих методів і засобів формування графічних зображень на гексагональному растрі;
  - встановити нові аналітичні залежності між параметрами пікселів;
  - визначити особливості формування крокових приростів при формування графічних примітивів;
- розробити нові та модифікувати існуючі:
- методи прискореного формування векторів на гексагональному растрі;
  - високопродуктивні методи формування кіл;
  - методи визначення типів крокових приростів для траєкторії кіл і

еліпсів;

- методи високопродуктивного контурного антиаліазингу зображень графічних примітивів;

- визначити особливості суперсемплінгу графічних зображень, які використовують гексагональну модель пікселя;

- розробити:

- конвертор зображень з прямокутного в гексагональний растр;

- графічний редактор для формування графічних зображень на гексагональному растрі;

- програмні та апаратні засоби для формування графічних зображень на гексагональному растрі;

*Об'єкт дослідження* – процес формування графічних зображень на гексагональному растрі.

*Предмет дослідження* – методи та засоби формування графічних зображень на гексагональному растрі.

**Методи дослідження.** У дисертаційні роботі використовувалися: теорія алгоритмів; теорія інтерполювання та апроксимації функцій; алгебраїчна та аналітична теорії чисел; дискретна математика; лінійна алгебра для розробки моделей пікселів і методів формування графічних примітивів; комп'ютерне моделювання для аналізу та валідації запропонованих теоретичних положень.

### **Наукова новизна отриманих результатів:**

1. Вперше запропоновано метод кругової інтерполяції на гексагональному растрі, особливість якого полягає у використанні апріорно визначеного стохастичного розподілу крокових приростів залежно від ділянки формування крокової траєкторії, що дало можливість в 1,7 раз збільшити швидкодію інтерполяції за рахунок прогнозування найбільш вірогідної комбінації кроків.

2. Вперше отримані співвідношення для визначення типів крокових переміщень для еліпсів і кіл залежно від ділянки формування крокової

траєкторії, що спрощує методи формування еліпсів за рахунок вилучення надлишкових обчислень.

3. Подальшого розвитку отримав метод оцінювальної функції для формування відрізків прямих на гексагональному растрі, який відрізняється від відомих визначенням в кожному інтерполяційному такті подвійних крокових приростів, що дозволило до двох разів зменшити час лінійної інтерполяції.

4. Подальшого розвитку отримав метод антиаліазингу векторів на гексагональному растрі, який відрізняється від відомих використанням для обчислення площі покриття пікселя додаткових оцінювальних функцій, розрахованих у виділених точках контуру гексагону, що дозволило виконувати антиаліазинг безпосередньо під час формування зображення крокової траєкторії та усунути етап постобробки, і, як наслідок, підвищити продуктивність формування зображення та його реалістичність за рахунок згладження крокової траєкторії.

5. Подальшого розвитку отримав метод антиаліазингу векторів на гексагональному растрі, який відрізняється від відомих визначенням площі покриття пікселя шляхом аналізу знаків оцінювальних функцій в центрах субпікселів, на які розбивається піксель, що дозволило розпаралелити процес антиаліазингу та підвищити його продуктивність. Використання антиаліазингу при формуванні векторних зображень забезпечує підвищення реалістичності за рахунок усунення ступінчатого ефекту.

**Практичне значення отриманих результатів** полягає в тому, що на основі отриманих в дисертації теоретичних положень запропоновано алгоритми та розроблено апаратні та програмні засоби формування графічних примітивів на гексагональному растрі для комп'ютерних систем візуалізації зображень.

Впровадження результатів досліджень підтверджується відповідними актами. Результати досліджень використовуються на таких підприємствах «ДРЕССЛАБ Юей», ТОВ «ЗД Дженерайшн Юей»; ПП «Фотоніка плюс»,

кафедрі програмного забезпечення Вінницького національного технічного університету для використання у навчальному процесі.

**Особистий внесок здобувача.** Усі наукові результати, викладені у дисертаційній роботі, отримані автором особисто.

У публікаціях, виданих у співавторстві, автору належать наступні результати: [1] – аналіз галузей використання гексагонального растру; [2] – метод антиаліайзингу; [3] – формули визначення крокових приростів; [4] – формули розрахунку оцінювальних функцій; [5] – модифікація методу, крокові прирости в секторах на гексагональному растрі [6] – модифікація методу, залежність типів крокових приростів від октанту [7] – модифікація методу розрахунку крокових приростів формування еліпса на гексагональному растрі; [8] – класифікація методів; [9] – формули розрахунку дистрибутивної функції; [10] – метод, значення оцінювальних функцій, структурну схему пристрою; [11] – алгоритм антиаліайзингу відрізка прямої; [12] – апроксимаційні формули; [13] – алгоритм градації інтенсивності кольору; [14] – метод антиаліайзингу з визначенням додаткових оцінювальних функцій в попередньо визначених точках зображення; [15] – структурні схеми блоків, апроксимаційні формули; [16] – формули розрахунку крокових приростів; [17] – формули для розрахунків; [18] – використання для інтерполяції стохастичного розподілу крокових приростів; [19] – формули розрахунків; [20] – алгоритм для складової кольору; [21] – аналіз інформаційних технологій; [22] – формули розрахунку модифікованої оцінювальної функції; [23] – метод морфологічного антиаліайзингу гексагонального растру; [24] – формули розрахунку гексагональних координат; [25] – аналіз стратегічних ігор; [26] – алгоритм параболічної інтерполяції; [27] – структури тайлів; [28] – аналіз використання гексагональних структур у флексографічному друці; [29] – алгоритм формування відрізків; [30] – аналіз гексагональних логічних ігор; [31] – особливості накладання гексагонального растру на карти; [32] – алгоритм реалізації; [33] – алгоритм формування кіл; [36] – проаналізовано

співвідношення; [37] – аналіз OLED-технологій; [38] – алгоритм мультисемплінгу; [39] – алгоритм розрахунку інтенсивності кольору; [41] – алгоритм формування дуг кіл.

**Апробація матеріалів дисертації.** Основні положення дисертаційної роботи доповідалися та обговорювалися на таких конференціях:

International Conference on Optoelectronic Information Technologies «PHOTONICS-ODS 2010» (Vinnytsia, 2010), Міжнародна науково-практична конференція «Перспективні інновації у науці, освіті, виробництві та транспорті», (м. Одеса, 2010), Міжнародна науково-практична Інтернет-конференція «Молодь в технічних науках: дослідження, проблеми, перспективи», (м. Вінниця 2015), VI Міжнародна конференція «Моделювання та комп'ютерна графіка" ДонНТУ", (м. Красноармійськ, 2015), Інтернет-конференція "Молодь в технічних науках: дослідження, проблеми, перспективи (м. Вінниця, 2016), VII Міжнародна науково-технічна конференція «Моделювання і комп'ютерна графіка», (м. Покровськ, 2017), Всеукраїнська науково-практична Інтернет-конференція "Молодь в науці: дослідження, проблеми, перспективи" (м. Вінниця, 2020), Республіканська науково-практична конференція «Інформаційні технології в освіті, техніці та промисловості, (м. Івано-Франківськ, 2020), Міжнародна науково-практична Інтернет-конференція «Електронні інформаційні ресурси: створення, використання (м. Вінниця, 2021), XII Міжнародна науково-технічна конференція «Інформаційно-комп'ютерні технології – 2021» (м. Житомир, 2021), I Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (м. Вінниця, 2021), VII Міжнародна науково-практична конференція «Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі»: (м. Київ, 2022), V Міжнародна наукова конференція «Інформація, комунікація та управління знаннями в глобалізованому світі» (м. Київ, 2022), Всеукраїнська науково-практична Інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи» (м. Вінниця, 2022), II Всеукраїнська науково-технічна

конференції молодих вчених, аспірантів та студентів «Комп'ютерні ігри та мультимедіа як інноваційний підхід до комунікації» (м. Одеса, 2022), XV Міжнародна науково-практична конференція «Інформаційні технології і автоматизація» (м. Одеса, 2022), VIII International scientific and practical conference “Modern research in world science” (Lviv, 2022), The 12th International scientific and practical conference “Modern research in world science” (Lviv, 2023), II International Scientific and Practical Internet Conference "Scientific Research and Innovation" devoted to modern achievements in science (Dnipro, 2023), XXXI Міжнародна науково-практична конференція «Інформаційні технології: наука, техніка, технологія, освіта, здоров'я. MicroCAD-2023» (м. Харків, 2023).

**Публікації.** Основні результати досліджень опубліковано в 41 наукових працях, у тому числі в закордонній монографії, 2 статтях, що входять до міжнародної наукометричної бази Scopus, 18 статтях у фахових виданнях України, 17 матеріалах конференцій, 4 авторських свідоцтвах про реєстрацію авторського права на комп'ютерну програму.

**Структура та обсяг роботи.** Дисертація включає вступ, п'ять розділів, висновки, список літератури, який нараховує 115 найменувань, і додатки. Основний зміст дисертації викладено на 143 сторінках і містить 9 таблиць і 111 рисунків. Додатки містять акти впровадження результатів роботи, лістинги програм.

## РОЗДІЛ 1

### АНАЛІЗ МЕТОДІВ І ЗАСОБІВ ФОРМУВАННЯ ГРАФІЧНИХ ЗОБРАЖЕНЬ НА ГЕКСАГОНАЛЬНОМУ РАСТРІ

На даному етапі розвитку комп'ютерної графіки [1-5] велику увагу приділяють формуванню реалістичних зображень за прийнятний час. При цьому часто використовують гексагональний растр [6-8], який забезпечує підвищення роздільної здатності екранної системи координат і більш високу згладженість контурів. Сьогодні гексагональний растр широко використовують у різних галузях, зокрема, у системах відображення інформації [9-10], у комп'ютерних іграх [11], [12], сенсорах [13-15], флексографічному друці [16], [17], геопросторових системах [18], [19], оптичних приладах [20], [21], окулярах віртуальної реальності [22], телескопах і т.д.

Детальний аналіз галузей використання гексагональних структур наведено в [7].

У даному розділі проаналізовано моделі пікселів, методи формування графічних примітивів, використання гексагональних структур у комп'ютерних системах формування та оброблення зображень.

#### **1.1 Аналіз моделей пікселів**

У засобах комп'ютерної графіки піксель [1-5] розглядають як обмежену контуром ділянку дискретного простору. Така ділянка може мати різну геометричну форму залежно від того, яка математична модель пікселя використовується.

Математична модель пікселя описує в просторі принцип розподілення світла, що випромінює точкове джерело [2]. Основна мета приведення пікселя до математичної моделі – це встановлення інтенсивності та (або) кольору певного пікселя. Математичні моделі пікселів дають змогу розробляти відносно прості для обчислень методи антиаліазингу [23].

Найпоширенішими є такі математичні моделі пікселів [23-27]: квадратна, кругова, конусна, гаусівська, гексагональна.

Використання тієї чи іншої моделі пікселів залежить від вимог до точності, графічних потужностей та від фізичної реалізації пікселів.

В аналітичних методах антиаліайзингу [4] при визначенні інтенсивності світла враховують технічні характеристики елементів екранної системи координат. Інтенсивності кольору пікселя розраховують за формулою [23]:

$$I_A(P_x, P_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I_{\text{ідеал}}(x, y) \cdot F(x - P_x, y - P_y) dx dy, \quad (1.1)$$

де  $I_A$  - інтенсивність кольору пікселя з координатами  $(P_x, P_y)$ ;  $I_{\text{ідеал}}(x, y)$  – аналітична функція, яка задає інтенсивність кольору в кожній точці простору;  $F(x, y)$  - модель пікселя, яка визначає розподіл світла в межах пікселя. Функцію  $I_A(P_x, P_y)$  часто називають функцією фільтра [23].

В методах антиаліайзингу [23], [24-26], [28] найчастіше розглядається частковий випадок розрахунку інтегралу (1.1) для обмеженого класу примітивів.

Найпростішою з точки зору обчислень є модель [23], [30], [31], в якій піксель має форму квадрата. Для спрощення обчислень сторону квадрата приймають одиничного розміру. Функція, що описує розподіл світла в просторі для квадратної моделі, описується таким виразом:

$$F(x, y) = \begin{cases} 1, & \text{якщо } |x| \leq 0.5 \text{ та } |y| \leq 0.5; \\ 0, & \text{якщо } |x| > 0.5 \text{ та } |y| > 0.5. \end{cases}$$

Задача антиаліайзингу зводиться до обчислення площі покриття тієї частини квадрату, що потрапляє в зображення. Знаходження площі частини квадрату не вимагає значних обчислень. Це обумовлює найпоширеніше використання квадратної моделі пікселя.



Квадратна модель [23], [30], [31] пікселя не забезпечує високої точності при формуванні зображень, однак використовується в засобах динамічної графіки як первинне наближення.

Математична модель, яка розглядає піксель як круг одиничного радіуса, отримала назву кругової моделі [23], [32]. Для такої моделі при реалізації антиаліайзingu інтенсивність кольору пікселя встановлюється пропорційно до площі круга, яка відтинається траєкторією примітива.

Площа круга одиничного радіуса визначається за формулою:

$$S_k = \pi \cdot r^2 .$$

Як видно з рис. 1.1 кругова модель пікселя не забезпечує максимальне заощення екрану.

Площа ділянки при одиничному радіусі кругової моделі дорівнює  $S_k = 4 - \pi$ .

При великих розмірах екрану має місце суттєвий рівень нетехнологічного використання його площі, що негативно впливає на роздільну здатність зображення.

В засобах відображення часто використовують конусну модель пікселя [23], [33]. У цій моделі піксель розглядається як конус з одиничним об'ємом. Для цієї моделі інтенсивність кольору пікселя має найбільше значення в центрі та лінійно зменшується до 0 [33]. При цьому центри пікселя і основи конуса суміщають.

Вважають, що об'єм конуса дорівнює одиниці. Для цього підбирають висоту конуса та діаметр його основи.

Інтенсивність кольору пікселя встановлюється пропорційно об'єму, який траєкторія примітива відтинає від конуса, як показано на рис. 1.2.

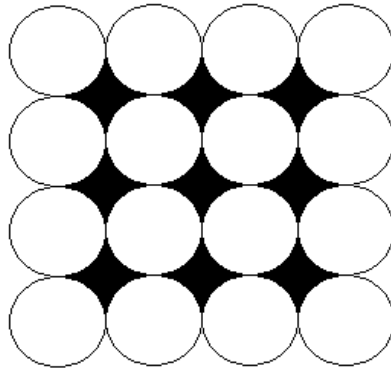


Рисунок 1.1 – Нетехнологічні зони при використанні кругової моделі пікселя

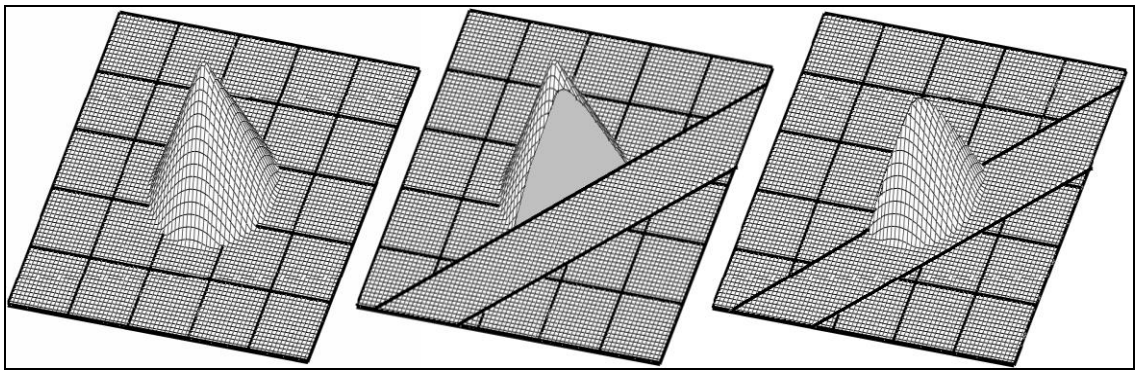


Рисунок 1. 2 – Перетин конуса траекторією прямої

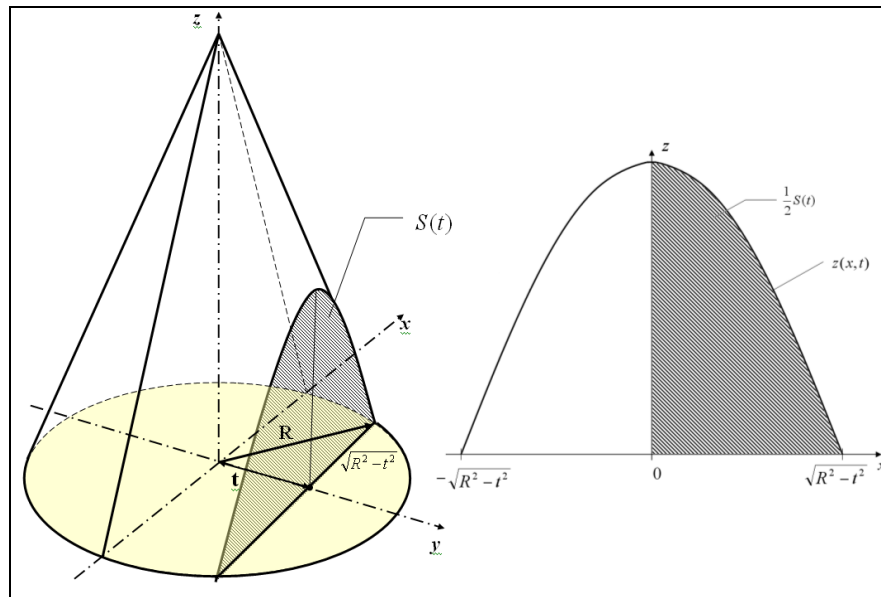


Рисунок 1. 3 – Перетин конуса вертикальною площиною

Інтенсивність кольору при використанні конусної моделі пікселя визначають за формулою [23], [33]  $I = F(d, t)$ , де  $t$  - товщина прямої.

Об'єм частини конуса знаходиться за формулою

$$F(d, t) = V(d + t) - V(d - t),$$

де  $V(d)$  - об'єм, який відтинається від конуса площиною, перпендикулярною до основи конуса.

Об'єм  $V(d)$  можна знайти за формулою:

$$V(d) = \int_0^d S(t) dt,$$

де  $S(t)$  - площа поперечного перетину конуса вертикальною площиною, яка віддалена на відстань  $t$  від центра конуса рис. 1.3;  $d$  - відстань між центром пікселя і вектором.

У засобах відображення на основі електронно-променевої трубки найадекватнішою вважається "гаусівська" модель пікселя (рис. 1.4) [26], [27], [33], [34]:

$$F(x, y) = \frac{1}{2\pi R^2} e^{-\frac{x^2 + y^2}{2R}},$$

де  $R = 1$  - радіус фільтра.

Як правило,  $R = 1$  гаусівська модель пікселя є найбільш фізично коректною.

Обчислення інтенсивності кольору при використанні гаусівської моделі пікселя полягає у визначенні об'єму фігури, що формується з перетину функції фільтрування та меж примітива. Враховуючи, що піксельна модель, заснована на гаусівській функції, вимагає значних обчислювальних ресурсів, її застосування обмежується випадками, коли до якості згладження

пред'являються високі вимоги. Графічно гаусівську модель пікселя можна представити як тіло обмежене площинами  $x\theta y$  і  $p(x, y)$  (рис. 1.4).

Інтенсивність кольору пікселя  $I$  для гаусівської моделі [33] визначається об'ємом тіла, обмеженого площиною  $x\theta y$ , поверхнею  $p(x, y)$  і вертикальною площиною, вздовж траєкторії примітива (рис. 1.5)  
 $I = F(d) = V(d) - V(d_1)$ .

Для рідкокристалічних дисплеїв (LCD) у якості  $F(x, y)$  використовують функцію Хемінга [31] (рис. 1.6).

$$F(x, y) = \begin{cases} \left( \cos\left(\frac{\pi \cdot x}{W}\right) \cos\left(\frac{\pi \cdot y}{W}\right) \right)^e, & \text{якщо } |x| \leq \frac{W}{2} \text{ та } |y| \leq \frac{W}{2}, \\ 0, & \text{якщо } |x| > \frac{W}{2} \text{ та } |y| > \frac{W}{2} \end{cases},$$

де  $W$ ,  $e$  - параметри для настроювання моделі під екран [23].

Для підвищення якості формування графічних сцен у засобах відображення використовують гексагональний растр [7], [12], [16], [20], [21], [35-45]. Базовим елементом формування гексагонального растра є правильний рівносторонній шестикутник – гексагон [51]. Гексагон, або рівносторонній шестикутник [6], [7], [51] – це двомірна геометрична фігура, що включає шість рівних сторін. Всі кути кути між сторонами, рівні  $120^\circ$  (рис. 1.7) На рис. 1.8 зображено розташування базового пікселя з сусідніми.

При формуванні точок на гексагональному растрі адресацію координат можна задати з кута екрана подібно квадратному растрі, або починати з центрального пікселя (рис. 1.9).

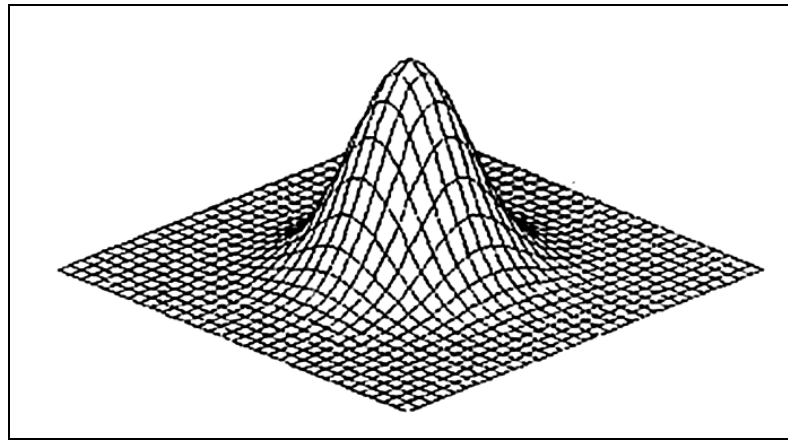


Рисунок 1.4 – Графічне зображення гаусівської моделі пікселя

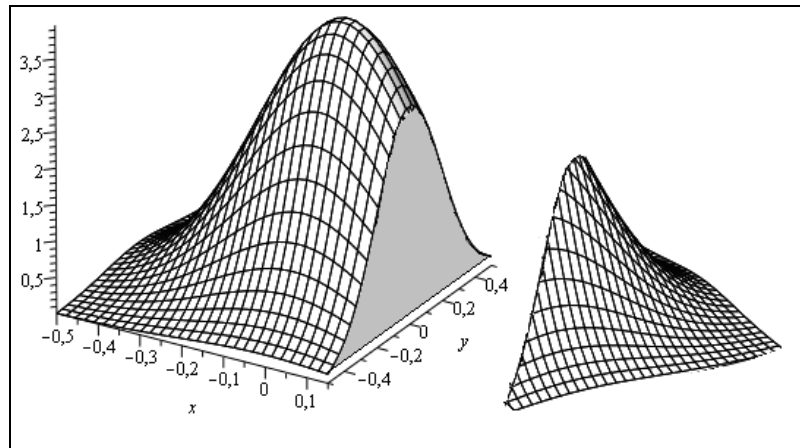


Рисунок 1.5 – Частина об'єму гаусівського пікселя

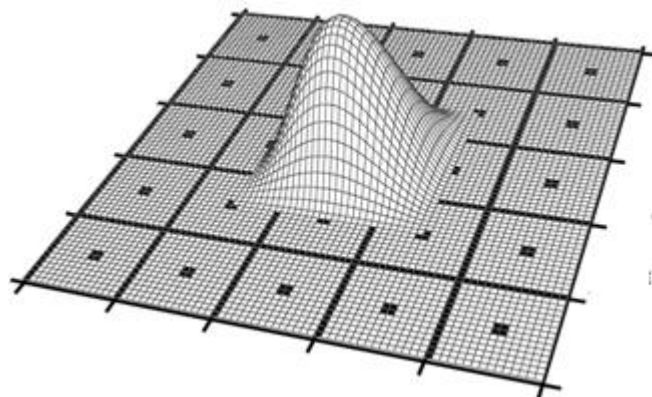


Рисунок 1.6 – Графічне зображення моделі пікселя у формі Хемінга

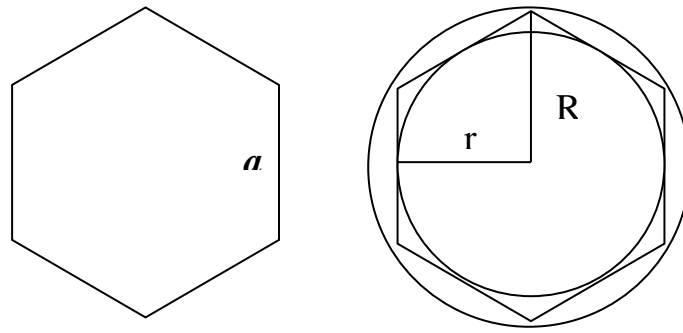


Рисунок 1.7 – Гексагон і вписане та описане кола

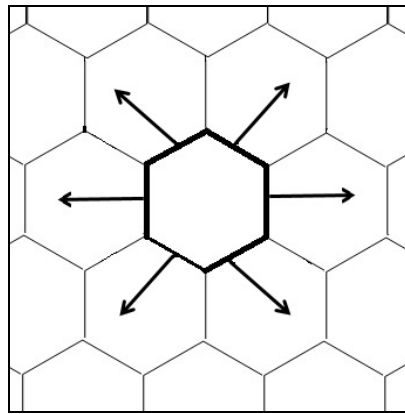


Рисунок 1.8 – Шестизв'язність гексагонального пікселя

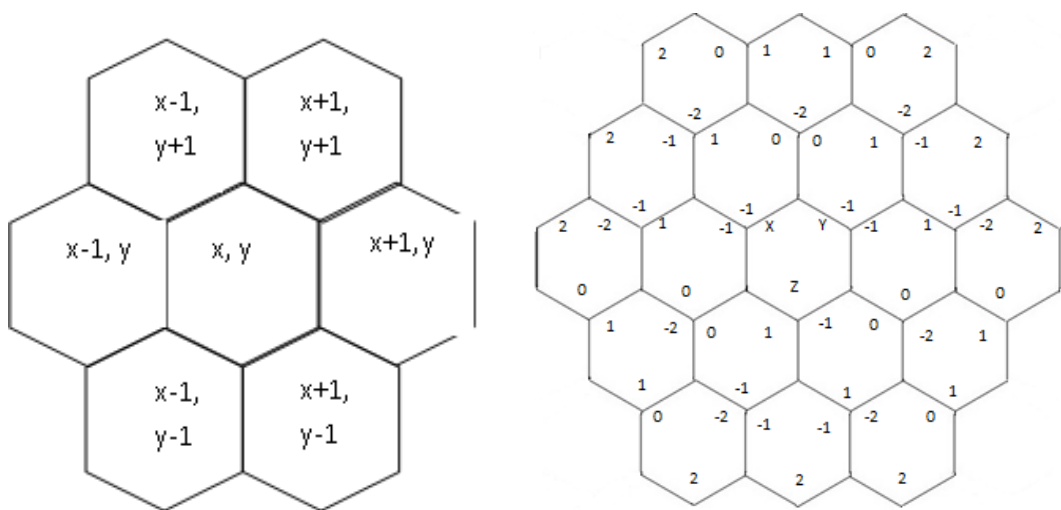


Рисунок 1.9 – Координати суміжних пікселів на гексагональному растрі

Сторони  $a$  правильного шестикутника рівні та дорівнюють радіусу кола, який описаний навколо шестикутника ( $a = R$ ). Усі кути гексагона

дорівнюють радіус вписаного кола  $r = \frac{\sqrt{3}}{2}a$ . Найдовша діагональ гексагону в два рази довша за його сторону.

Периметр гексагона та його площа розраховується відповідно за виразами:

$$P=6R = 4\sqrt{3}r, S = \frac{3\sqrt{3}}{2}R^2 = 2\sqrt{3}r^2.$$

При покритті площини гексагонами кожен піксель оточено шістьма сусідніми пікселями, що робить цю сітку шестиз'єднаною (див. рис. 1. 8).

Розрахуємо площу гексагона через радіус описаного кола, за умови, що  $R=1$  [51]. Оскільки у правильного шестикутника радіус описаного кола

дорівнює стороні, то  $S = \frac{3\sqrt{3}}{2}R^2 = 2,598$ . Якщо для круга з  $R=1$ , то

$S_k = \pi R^2 = 3,14$ . Знайдемо співвідношення площ фігур з площею круга.

У таблиці 1.1 наведено співвідношення [51] площ багатокутників до площі круга. З таблиці 1.1 видно, що з наведених фігур гексагон найбільше наближається до круга.

Таблиця 1.1 – Співвідношення площі багатокутників з площею круга одиничного радіуса

Геометрична форма	Площа	Співвідношення площі багатокутника по відношенню до кола
Круг	3,14	1
Трикутник	1,299	0,414
Квадрат	1,99	0,634
Гексагон	2,598	0,827

Піксель у формі гексагону задається сукупністю однакових субпікселів (рис.1.10).

Ці елементи зберігають шестикутну структуру. Відстань від центрального сегмента до його сусідів є однаковою.

Розглянемо формування сегменту субпікселів [51]. В ньому група субпікселів розташована на однаковій відстані від центрального субпікселя. Ці субпікселі утворюють фігуру, що перетинає коло, як показано на рис. 1.10. Лінійний підсегмент субпікселів – це набір субпікселів, центри яких розташовані на одній прямій лінії і входять до складу одного сегменту.

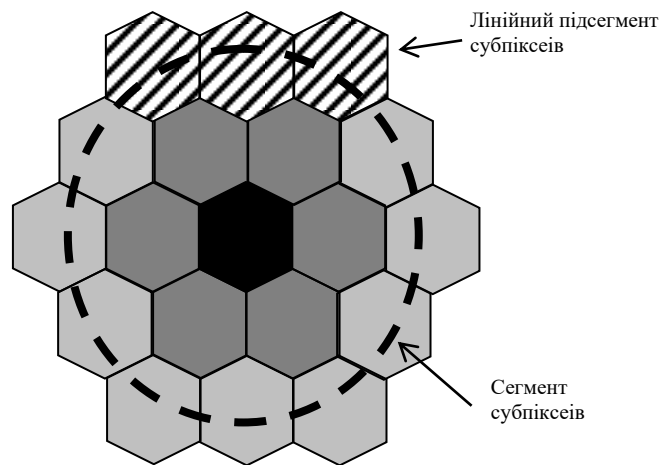


Рисунок 1.10 – Розбиття пікселя на субпікселі

Властивість гексагональних пікселів: при розбитті пікселя на субпікселі кількість субпікселів завжди буде непарною.

Характерною ознакою гексагональних пікселів є те, що при їх поділі на субпікселі загальна кількість субпікселів завжди стає непарною. Кожна пара сусідніх субпікселів з обраного лінійного підсегменту створює один субпіксель у наступному сегменті, внаслідок чого кількість новоутворених субпікселів зменшується на один від утворюючого.

Однак, крайні точки лінійного підсегменту формують лише по одному субпікселю. Враховуючи, що кількість лінійних підсегментів у заданому сегменті дорівнює шести, а їх вершини є спільними для двох сусідніх



підсегментів, то формула для обчислення кількості субпікселів у сегменті виглядає так:

$$(l+1)*6-6,$$

де  $l$  - число лінійного сегменту.

З цієї формули видно, що результат множення на 6 завжди буде парним, а віднімання 6 від парного числа також дасть парне число. Таким чином, сумарна кількість субпікселів у сегменті буде парною. Оскільки в центрі кожного пікселя знаходиться один центральний субпіксель, загальна кількість субпікселів виходить непарною.

Крім того, кількість субпікселів у кожному наступному сегменті зростає на 6 (рис. 1.11).

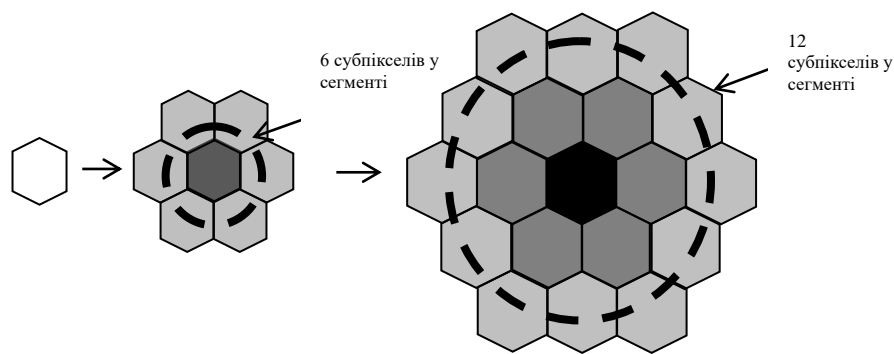


Рисунок 1.11 – Збільшення деталізації при розбитті пікселя на субпікселі

При поділі гексагонального пікселя на субпікселі, інтенсивність кольору кожного наступного сегмента субпікселів зменшується відповідно до їх віддаленості від центру. Це означає, що інтенсивність кольору може визначатися пропорційно до числа субпікселів, що попали в ділянку, обмежену полігоном. При цьому врахується їх віддаленість від центру.

При розробці алгоритмів формування графічних примітивів на гексагональному растрі використовують його властивості [51]: усі внутрішні кути рівні між собою; кожен внутрішній кут правильного шестикутника дорівнює  $120^\circ$ ; усі сторони рівні між собою; велика діагональ шестикутника служить діаметром кола, описаного навколо шестикутника. Вона дорівнює

довжині двох його сторін; менша діагональ шестикутника є більшою за сторону шестикутника; діагоналі шестикутника перетинаються в центрі і ділять шестикутник на шість рівносторонніх трикутників (висота кожного з цих трикутників дорівнює радіусу вписаного в шестикутник кола); трикутник, що складається з сторони шестикутника та його більшої та меншої діагоналей, є прямокутним, при цьому його гострі кути становлять  $30^\circ$  і  $60^\circ$ ; правильний шестикутник має здатність заповнювати площину без проміжків та накладань, що робить його використання доцільним для формування растрових зображень.

На рис. 1.12 показані співвідношення між кутами правильного шестикутника, що можуть бути застосовані у розробці алгоритмів для генерації графічних зображень на гексагональній сітці.

Передача кольорів у гексагональному растровому форматі має свої унікальні особливості. Поділ гексагонального пікселя на субпікселі надає можливість для більш деталізованої передачі кольору. Це, в свою чергу, дозволяє розміщувати елементи для відтворення основних кольорів таким чином, щоб значно підвищити якість зображення завдяки розширеному спектру кольоропередачі, як показано на рис. 1.13.

Проаналізуємо можливість апроксимації конусної моделі пікселя з урахуванням того, що основа пікселя представлена не кругом, а правильним шестикутником. Використання конусної моделі пікселя для антиаліайзингу передбачає визначення інтенсивності кольору пропорційно до об'єму, відсіченого від конуса вертикальною площиною.

Об'єм конуса (рис.1.14)  $V = \frac{1}{3}\pi R^2 h$  , де  $h$  – висота конуса. Прийmemo

об'єм конуса рівним одиниці. Висота конуса  $h = \frac{3}{\pi R^2}$  . Якщо радіус  $R$  покриває три субпікселі, то  $h = 0,106$ . Зрозуміло, що для центрального субпікселя  $R = 0,5$ . Тому об'єм складового конуса 1 (рис. 1.14). Об'єм конуса 2 (7 субпікселів в основі конуса) дорівнюватиме  $V_7 = 0,492$ . Приведемо

інтенсивність кольору субпікселя відповідно до об'ємів конусів. Для центрального субпікселя інтенсивність буде відповідати об'єму  $V_1 = 0,123$ .

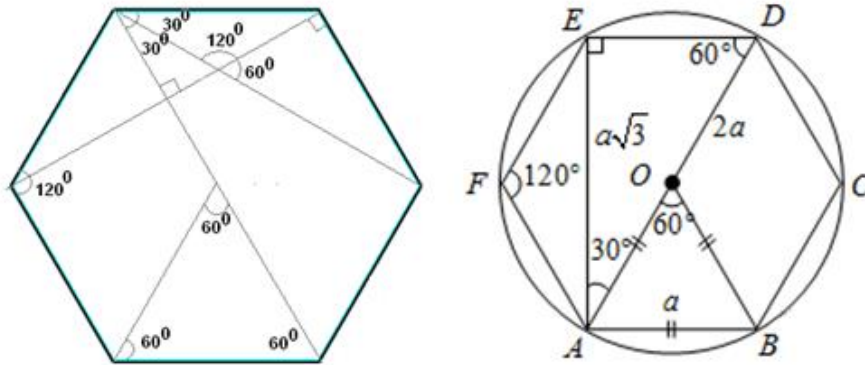


Рисунок 1.12 – Співвідношення між кутами гексагону

R- red	Червоний
G-green	Зелений
B- blue	Синій
C-cyan	Блакитний
M-magenta	Пурпуровий
Y-yellow	Жовтий

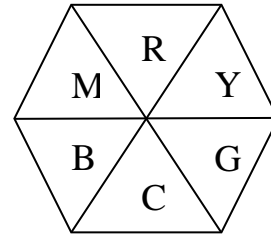


Рисунок 1.13 – Відтворення кольору в гексагональному пікселі

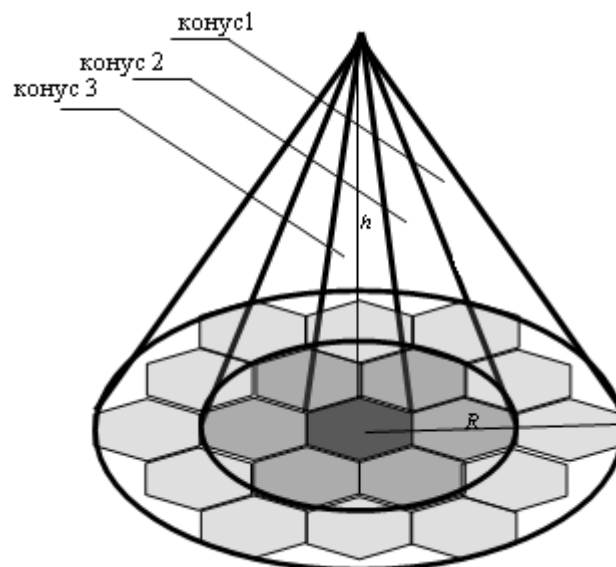


Рисунок 1.14 – Розташування конусів

Складову об'єму для субпікселів другого конуса можна визначити якщо від об'єму конуса 2 відняти об'єм конуса 1 і поділити на число субпікселів другого сегменту. Такі самі розрахунки проведемо для третього сегменту субпікселів. Матимемо такий розподіл інтенсивності як показано на рис. 1.15.

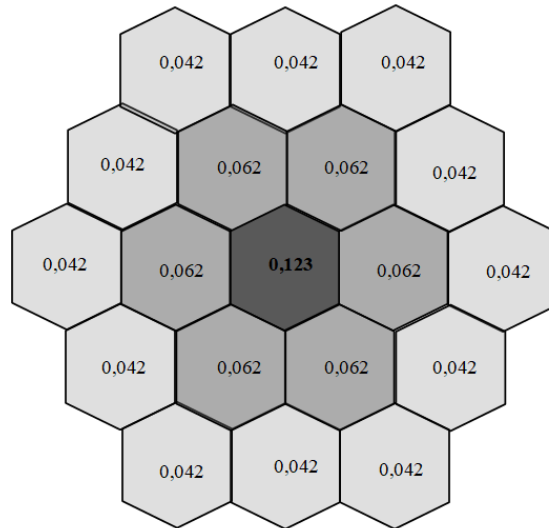


Рисунок 1.15 – Розподіл інтенсивності кольору

Наведений аналіз гексагональної моделі пікселя підтверджує перспективність його використання в системах формування графічної інформації.

Аналіз методів суперсемплінгу, які отримали широке застосування в системах формування реалістичних зображень, детально подано в [26], [44].

## 1.2 Обробка зображень на гексагональному растрі

Питаннями обробки гексагональних зображень активно займаються останні 40 років. Першу фундаментальну роботу зі зберігання гексагональних зображень було виконано Peter J. [57]. Мотивацією було знайти схеми зберігання зображень з використанням даних дерева та піраміди.

Переваги гексагональної решітки вибірки над квадратними були встановлені теоретично Д. Уайтхаусом [58]. Це було зроблено на основі

порівняння різних типів розбиття.

Р. Вітуллі [59] довів, що ефективність гексагональної дискретизації перевищує ефективність квадратної дискретизації, оскільки потрібно приблизно на 13% менше пікселів для отримання такої ж продуктивності, як і при квадратній дискретизації. Р. Мерсер [60] з метою економії обчислювальних ресурсів, що забезпечується гексагональною вибіркою, сформулював теорему гексагональної вибірки. Це було використано для отримання дискретного гексагонального перетворення Фур'є.

М. Голей [61] розробив паралельні обчислювальні системи з урахуванням гексагональних структур.

Стонтон [62] виконав роботи над практичною апаратною системою для обробки гексагональних зображень. Захоплювач кадрів був модифікований для виконання гексагональної вибірки. Отримані дані були оброблені за допомогою конвеєрної комп'ютерної системи. Було досліджено ряд алгоритмів, такі як обробка порогових значень і локальні оператори, а оброблені зображення відображалися на екрані комп'ютера. Гексагональні координати були представлені в системі координат, повернутої на кут. Порівняння проводилося за допомогою симулятора, який представляв шестикутний піксель як сукупність квадратних пікселів.

Активно проводяться дослідження щодо використання гексагональних структур для розпізнавання зображень, зокрема, запропоновані методи гексагонального оброблення зображень для розпізнавання обличчя [63-65]. Показано, що обробка гексагональних зображень набагато ефективніша, ніж при використанні існуючих методів.

Решітка вибірки, яка використовується для оцифрування даних безперервного зображення, є суттєвим визначальним фактором якості кінцевого цифрового зображення, а отже, ефективності його обробки. Для рівномірного вибору зображень довільного розміру, решітка повинна відповідати регулярній просторово повторюваній мозаїці. Цим вимогам [6] відповідає гексагональна структура.

Останнім часом відновився інтерес до використання гексагональних зображень, представлення архітектур для таких зображень і загальної обробки гексагональних зображень. Розроблені багатомасштабні гексагональні градієнтні оператори, на основі рамкових кінцевих елементів, для використання безпосередньо на гексагональних зображеннях.

Виявлення руху, виявлення країв і визначення орієнтації часто вимагають просторового обчислення різниці інтенсивності світла між сусідніми піксельними осередками [66]. Попереднє опрацювання зображення може підвищити ефективність обчислення, коли паралельне оброблення може бути досягнуто шляхом просторового розташування масиву пікселів. Геометричну оптимізацію масивів пікселів можна досягти, використовуючи гексагональні масиви замість прямолінійних масивів. Гексагональні масиви покращують щільність упаковки та зменшують складність просторового обчислення порівняно з квадратними та восьмикутними масивами. Вони також забезпечують геометричну симетрію для ефективного обчислення не лише на рівні суміжної сусідньої комірки, але й на рівні сусідньої комірки вищого порядку. Гексагональні масиви підвищують обчислювальну ефективність завдяки використанню симетричної конфігурації, яка дозволяє попередню обробку просторової інформації візуальної сцени за допомогою апаратних реалізацій, які повторюються у всіх сусідніх пікселях вищого порядку.

Гексагональні моделі використовують в системах захисту інформації. Так, зокрема, розроблено ефективний метод стеганографії [67] для приховування даних у зображеннях.

Біологічну оптичну систему [68] розглядають як каскад субфільтрів від фоторецепторів через гангліозні клітини в ямці до простих клітин у зоровій корі. Ця інформація надихнула багатьох дослідників вивчити біологічну сітківку, щоб дізнатися більше про можливості обробки інформації. Конуси та палички фоторецепторів у ямці людини мають форму шестикутника. Гексагональна сітка забезпечує вищу щільність упаковки, узгоджене

з'єднання сусідів і кращу кутову корекцію порівняно з прямолінійною квадратною сіткою.

Запропоновано алгоритм [68] комп'ютерного зору з використанням структури гексагональної сітки для програм, які узгоджені з зоровою системою людини. Він надає шестикутні змодельовані зображення для візуальної перевірки без використання гексагонального пристрою захоплення чи відображення. Результати моделювання показують, що запропонований алгоритм досягає вищого пікового співвідношення сигнал/шум - 98,45 і пропонує зображення високої роздільної здатності з меншою середньоквадратичною помилкою - 0,59.

Гексагональна структура [69] відповідає принципам моделюванню зорової кори головного мозку: рівновіддалені зв'язки між нейронами, клонування та поширення інформації на поверхні кори як інваріантний аналіз візуальних об'єктів, реалізує механізм конкурентній орієнтації нейронів.

Відмінною характеристикою топології нейронної мережі є гексагональне розташування збуджувальних зв'язків між нейронами, які дозволяють поширювати або клонувати інформацію на поверхні нейронного шару. Клонування інформації та модифікація ваги зв'язків між нейронами використовуються як базові принципи процесів навчання та розпізнавання. Гексагональні топологічні структури є перспективним напрямком для розпізнавання просторово-часових паттернів і вивчення механізмів пам'яті в мозку.

На основі багаторічного досвіду моделювання меж людського зорового сприйняття Оверінгтон [70] запропонував матричну модель людського зору. Етап вибірки передбачав просте переупорядкування даних вхідного зображення, зменшуючи кожні вісім рядків до семи для наближення ідеальної гексагональної вибірки. Дані відображалися на зміщеній «цегляній стіні». Операції із зображеннями низького рівня були протестовані в цій структурі, і модель виявилася сумісною з багатьма аспектами ранньої зорової системи людини.

Досліджуються різні аспекти гексагонального відбору проб, у тому числі еволюція гексагональних масивів біологічних сенсорів, вплив вибору сітки відбору проб і мозаїки на зовнішній вигляд зображення та світлозбірні властивості датчика. Обговорюється вплив сітки вибірки на структуру багатопроцесорних систем.

Розроблено процедури [71], для швидкого розпізнавання образів тривимірних об'єктів за допомогою паралельної обробки на гексагональній сітці. Основними етапами схеми є виявлення країв на основі різниць у рівнях сірого в околі, локальне потоншення країв, локальне виявлення особливостей і корекція ознак на основі порівняння особливостей у піксельному супероколі. Експериментальні випробування цих процедур показали певний успіх у розпізнаванні ключових особливостей скелетів об'єктів. Результати свідчать про те, що впровадження та подальший розвиток цих процедур на реальній гексагональній сітці фотодетекторів з підключеними блоками паралельної обробки були б дуже плідними.

Гексагональний растр [72] краще передає форму кривих, оскільки має кращу геометричну структуру.

Гексагональна сітка характеризується найнижчим співвідношенням периметра до площі серед усіх традиційних плоских мозаїк. На практиці це призводить до мінімізації ефектів країв при роботі з гексагональними сітками. Кожна клітина у шестикутній сітці має шість ідентичних сусідніх клітин. Відстань між центрами (центроїдами) цих клітин є однаковою для всіх сусідів.

У роботі [72] представлені результати порівняльного аналізу використання гексагональної та прямокутної сіток для дискретизації кривих Безьє. Було здійснено дискретизацію довільно заданих кубічних кривих Безьє на обох типах сіток. Виявлено, що дискретне подання кривих на гексагональній сітці точніше передає форму та розміри об'єктів у 80-85% випадків. Розроблено теоретичні основи для використання для формування сплайнів гексагональних структур [73], [74].



Сьогодні розроблено гексагональні підходи до машинного та глибокого навчання, які здебільшого пов'язані з розробкою гексагональних згорткових нейронних мереж і шестикутних згорткових шарів [75].

### **1.3 Використання гексагональних елементів у системах візуалізації**

Останні роки характеризуються особливою увагою до використання гексагональних структур для виготовлення екранів.

Технологія дисплея Micro-LED [76], [81] з використанням гексагональних елементів вважається наступним поколінням технології відображення, яка має характеристики високої мініатюризації, тонкої плівки та інтеграції, а також високу яскравість, контрастність, швидку швидкість реагування та тривалий терміну служби.

Починаючи з 2000 року рідкокристалічний дисплей (LCD) і технологія дисплеїв на органічних світлодіодах (OLED) широко використовуються і поступово домінують на ринку дисплеїв.

У даний час стандартними екранами, що використовуються, є рідкокристалічні дисплеї (LCDs). LCDs -екрани генерують світло за допомогою підсвічування, і світло модулюється рідкими кристалами в поєднанні з кольоровими фільтрами, утворюючи окремі субпікселі. Така конструкція досить неефективна, оскільки ефективно використовується лише 5-10% світла. Навпаки, OLED-дисплеї складаються з органічних субпікселів, які діють як власний випромінювач світла. Це дозволяє індивідуально контролювати яскравість, а екрани можна зробити гнучкими. Технологія OLED також усуває витік світла з елемента, що покращує загальну ефективність.

У даний час для вирощування шарів GaN в основному використовуються гетерогенні епітаксціальні підкладки (такі як кремній (Si), сапфір ( $\alpha$ -Al<sub>2</sub>O<sub>3</sub>), карбід кремнію (SiC) тощо. Щільність дислокацій гетероепітаксії можна зменшити шляхом додавання буферних шарів (таких

як буферний шар AlGa<sub>N</sub>/AlN, буферний шар суперґратки AlN/GaN, буферний шар Al<sub>x</sub>Ga<sub>1-x</sub>N тощо), латеральної епітаксії.

Матриця μLED, вирощена на напівполярній площині, дозволить отримати повноколірні дисплеї великої площі. Крім того, матриця μLED показала однорідність оптоелектронних характеристик елементів відображення.

Щоб охопити повноколірну гаму для дисплеїв, світлодіодні пікселі повинні містити червоні, сині та зелені окремі світлодіоди, розташовані в масиві [76]. Роздільна здатність дисплеїв залежить від розміру та кроку окремих світлодіодів.

Важлива властивість [76] нової технології полягає в тому, що пікселі μLED керуються індивідуально та можуть залишатися підсвіченими протягом усього кадру. Індивідуальне керування пікселями μLED досягається за допомогою схеми керування пікселями, побудованої з тонкоплівкових транзисторів (TFT) або комплементарних транзисторів метал-оксид-напівпровідник (CMOS).

Світлодіоди (світлодіоди) на основі GaN випромінюють світло у видимому діапазоні шляхом регулювання складу індію (In) [76].

Однією з важливих переваг OLED-моніторів [80] є їх гнучка структура, яка дозволяє створювати вигнуті екрани. Це відкриває нові можливості для створення інноваційних форм-факторів і застосувань.

Квантові точки (Quantum Dots) є ще однією важливою технологією, яка сприяє покращенню властивостей OLED-моніторів.

Квантові точки [77] – це нанометричні розміри полімерних матеріалів, які можуть конвертувати світло одного кольору в інший. Використання квантових точок допомагає підвищити яскравість і розширити колірну гаму OLED-панелей. Це робить зображення на моніторах ще більш реалістичними.

Однією з головних переваг цих технологій є їхнє низьке споживання енергії. OLED-монітори [76-85] і панелі на основі квантових точок ефективно використовують енергію.

Майбутнє цих технологій полягає в подальшому підвищенні ефективності та тривалості роботи світлодіодів. Тонкі та гнучкі OLED-панелі матимуть все більше застосувань у різних сферах, забезпечуючи більше інновацій та візуальних досвідів для користувачів.

Через різний опір електрода та пристроїв навколо контакту електрода під час ін'єкції струму світлодіода утворюється тепло, і температура пристрою постійно підвищується, що призводить до зменшення терміну служби.

Для більш високої надійності світлодіодів розроблено нову технологію [76] на основі InGaN/GaN з гексагональною плівкою нітриду бору (h-BN).

На рис. 1.16 наведено схему світловипромінюючого діода (LED) на основі InGaN/GaN з гексагональною плівкою нітриду бору (h-BN) [78].

Вирощування нанопірамід [79] із шестикратними напівполярними гранями на SiO<sub>x</sub> або SiN<sub>x</sub>, замаскованих с-площиною GaN, є альтернативним варіантом для покращення оптоелектронних характеристик світлодіодів на основі InGaN (рис. 1.17). Така конфігурація як на рис. 1.18 може бути використана для покращення продуктивності світлодіодів з високим вмістом компонентів In, оскільки, на додаток до нижчого поля поляризації цих кристалічних площин, тривимірні наноструктури забезпечують кращу релаксацію деформації для росту кристалів.

Зі збільшенням щільності пікселів і зменшенням розміру пікселя труднощі в досягненні високої продуктивності та високої точності інтеграції за допомогою процесів гібридної інтеграції різко зростають, що перешкоджає реалізації недорогого великомасштабного виробництва.

Останнім часом у індустрії дисплеїв спостерігається використання мікроструктурованих світлодіодів ( $\mu$ LED) розміром менше 50 мкм [76], [83].

У 2017 році SEA-LETI представив прототип дисплею з відеографічною матрицею з кроком пікселів 10 мкм на основі матриці Micro-LED 3.

У 2018 році Sony випустила гігантський зшитий 16K Micro-LED дисплей «CLEDIS». Корпорація AUO з Тайваню продемонструвала

12,1-дюймовий повноколірний дисплей Micro-LED з розміром пікселя менше 30 мкм, PPI 169 і роздільною здатністю 1920×720.

На InfoComm 2019 року Samsung анонсувала 292-дюймовий дисплей з великим екраном 8K [80]. Mojo Vision показала прототип мікродисплея з 14 000 пікселів на дюйм на основі 0,018-дюймової панелі. 2022 році компанія Seoul Viosys з Південної Кореї показала 54-дюймовий і 81,5-дюймовий екран Micro-LED з відстанню між пікселями 625 мкм і 937,5 мкм відповідно. Базуючись на структурі Micro-LED, її можна використовувати для виготовлення екранів 4K від 100 до 200 дюймів [81].

За даними [80] Omdia, Apple в 2021 році купила 172 мільйона OLED-панелей у Samsung, LG і BOE. Із 172 мільйонів панелей 106 мільйонів призначені для нової лінійки iPhone 13. Зокрема, BOE надасть 9 мільйонів OLED-екранів для Apple iPhone 12 Pro. Першим смартфоном, що використовує технологію гексагонального розташування кристалів BOE, стане Huawei P50 Pro+.

У 2023 році багато компаній Crystals [81] оголосили, що незабаром випустять свої продукти на основі гексагональних пікселів.

Наведений аналіз проказав перспективність використання гексагональних структур для відображення інформації, зокрема, для побудови моніторів.

Інтенсивні дослідження проводяться щодо використання гексагональних структур для запам'ятовування інформації [77], [78], [82], [83].

Досліджено гексагональний нітрид бору (h-BN) і його гетероструктури [77]. Цей квантовий матеріал може застосовуватись для електроніки, фотоніки, зондування та зберігання/перетворення енергії.

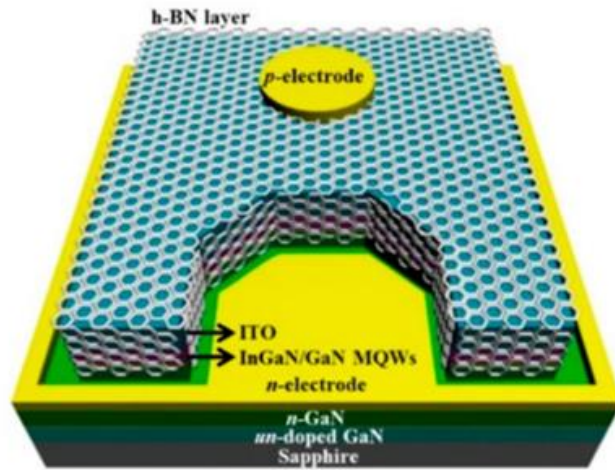


Рисунок 1. 16 – Світловипромінюючий діод (LED) на основі InGaN/GaN з гексагональною плівкою нітриду бору (h-BN) [78]

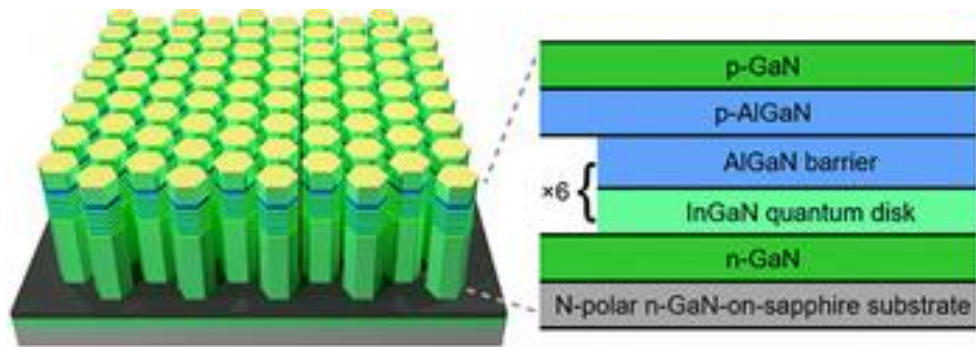


Рисунок 1.17 – Структура гексагонального стержня [79]

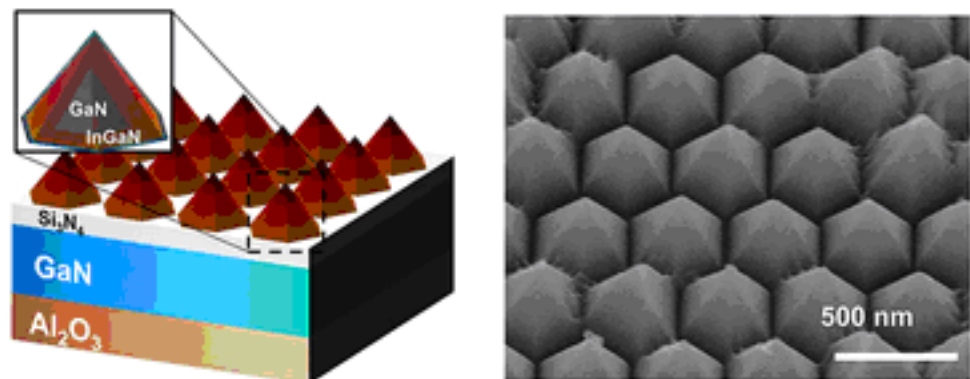


Рисунок 1.18 – Вигляд гексагональних індикаторних елементів [79]

Виявлено, що світло, випромінюване цими ізольованими комірками, можна використовувати для зберігання квантової інформації. Квантова пам'ять є основним структурним блоком, який необхідно для створення квантового Інтернету, де квантова інформація зберігається та надсилається через фотони.

Розроблено пам'ять типу «запис-багаторазове читання» (WORM), виготовленої з шарів квантових точок CdSe–ZnS quantum dots, розміщених між двома ізоляційними 2-вимірними гексагональними нітридами бору (hBN).

Створено платформу для зберігання квантових збуджень у формі спін-хвильових моделей в групі атомів, що містяться в оптичній порожнині. На рис.1.19 наведено структуру [82] блока пам'яті на h-BN ALD діелектричного матеріалу, отриманого атомно-шаровим осадженням IBSD методом іонного пучка. На рис. 1.20 MBE – основа, отримана методом молекулярно-променевої епітаксії, SVD- діелектричний шар. Досягнуто зменшення товщини до рівня одного шару, що обумовлює перспективи застосування для побудови великомасштабної надтонкої гнучкої пам'яті.

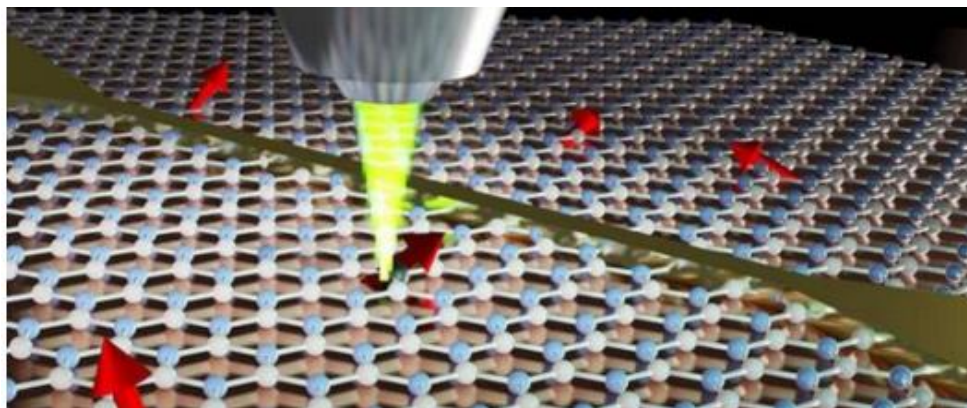


Рисунок 1.19 – Відображення ізольованих спінів на гексагональному нітриді бору під оптичним мікроскопом [82]

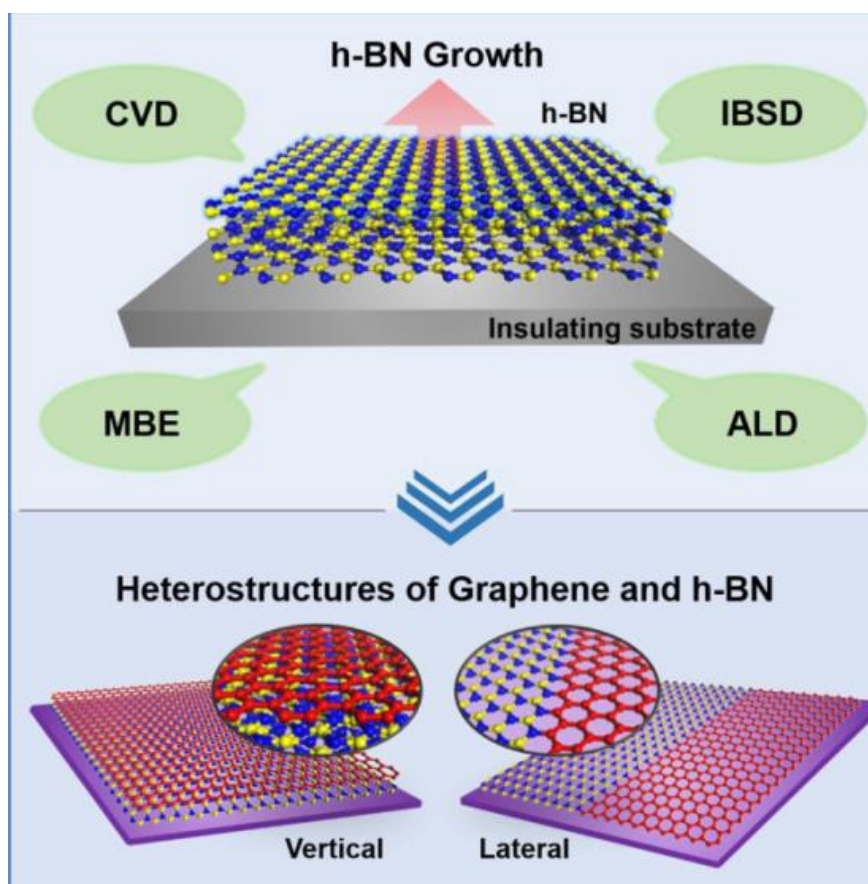


Рисунок 1.20 – Реалізація блока пам'яті на h-BN [82]

#### 1. 4 Аналіз методів побудови графічних примітивів

Процес генерації зображення растрового геометричного об'єкту називають растретизацією [1-4], [30]. Для відображення геометричний об'єкт задається в виді набору типових елементів. Растретизацію цих елементів необхідно виконати з високою швидкістю апаратними та програмними засобами графічного процесора. Найпоширенішими є відрізок прямої, коло, еліпс, парабола.

Відтворення основних графічних примітивів можна здійснювати методами [30] прямого розрахунку координат точок за математичним рівнянням. Такий спосіб дає хорошу точність растретизації, але при розрахунку використовуються операції множення і ділення над числами з плаваючою комою. Такі математичні операції порівняно з додаванням і відніманням потребують більшого часу, тому метод прямого розрахунку

координат застосовують лише для растрування відрізків прямої.

Для формування зображення відрізка прямої найчастіше використовують метод, заснований на розрахунку оцінювальної функції (ОФ) [1-5], [30], [87]; метод цифрового диференційного аналізатора (ЦДА) [1], [2], [86], [87]. Іноді використовують метод, оснований на використанні рівняння прямої [30], [87].

Сформуувати траєкторію відрізка прямої можна, використовуючи рівняння прямої:

$$Y = Y_n + \Delta Y \cdot \frac{X}{\Delta X} .$$

За умови, що  $\Delta X \geq \Delta Y$ , у формулу підставляють абсцису  $X$  і обчислюють відповідну координату  $Y$ , а при  $\Delta X < \Delta Y$  – навпаки. Точність відтворення зображення траєкторії відрізка прямої визначається округленням результату і може бути максимальною.

В методі використовуються операції множення та ділення, що обмежують його продуктивність.

У методі цифрового диференційного аналізатора [1-3], [30], [87] перед циклом інтерполяції і в подальшому використовують в режимі нагромаджувального додавання. Знак переносу при додаванні визначає тип крокового переміщення.

Для формування зображення траєкторії відрізка прямої найчастіше використовують метод оцінювальної функції [1-5], [30], [87], [90]. Це пояснюється простотою обчислювального процесу, оскільки в циклі інтерполяції використовують мікрооперації додавання та віднімання. При цьому досягається мінімальна похибка інтерполяції.

Характерна особливість метода полягає в тому, що при інтерполюванні використовується не значення оцінювальної функції, а її знак, який визначає тип крокового приросту.

На рис. 1.21 наведено принцип формування оцінювальної функції, який полягає у формуванні такої ОФ, яка має різний знак залежно від розміщення



стосовно ідеального відрізка прямої.

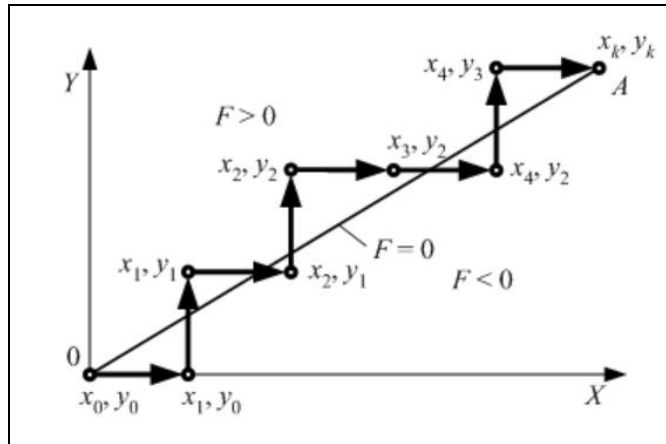


Рисунок 1.21 – Інтерполяція за методом оцінювальної функції

Для підвищення точності лінійного інтерполювання сусідні ортогональні прирости замінюють на діагональний, що покращує згладженість крокової траєкторії.

У методі [1-3], [30], [87] цифрового диференційного аналізатора часто використовується двійковий помножувач [30], [87] (інтегратор послідовного переносу), назва якого визначається виконанням останнім функції виду:

$$f_{вих} = f_{вх} \cdot \frac{УК}{2^n}$$
, де УК – значення керуючого коду,  $n$  – розрядність помножувача,  $f_{вх}$ ,  $f_{вих}$  – значення відповідно вхідної та вихідної частоти опорної імпульсної послідовності.

При використанні двійкових помножувачів має місце велика похибка інтерполяції і однаковий час формування відрізків прямих незалежно від значення координатних приростів відрізка прямої. Розроблено метод [87], заснований на скороченні циклу роботи помножувача, який усуває останній недолік. Однак метод має низьку точність інтерполяції.

Найпершими методами, які давали можливість відтворення зображення кола, були методи ЦДА, які використовували цифрові інтегратори.

В методі використовуються два цифрових інтегратора, охоплених перехресними зворотними зв'язками. При цьому досягалася висока

швидкодія, однак точність відтворення примітиву була низькою [87], що обмежує їх застосування, незважаючи на простоту реалізації інтерполятора. Важливою особливістю методів із застосуванням цифрових інтеграторів є постійність кругової швидкості руху точки по дузі кола, незалежно від радіуса кола. При цьому кола різних радіусів формуються за однаковий проміжок часу, що в окремих випадках призводить до потреби суттєвого підвищення продуктивності інтерполятора. А це в свою чергу призводить до його ускладнення.

Відомий метод кругової інтерполяції [30], [87], який передбачає застосування постійної пам'яті для апроксимації дуги кола за допомогою лінійної інтерполяції. Його недоліком є потреба у великих об'ємах пам'яті при великій кількості ділянок апроксимації. Похибка при обчисленнях за цим методом накопичується, оскільки дуга апроксимується ламаною лінією. Є потреба корекції накопиченої похибки.

Відомий метод [30] для інтерполяції кола в полярних координатах - метод площ. Він не отримав широкого поширення оскільки вимагає завдання кола в полярних координатах і характеризується низькою швидкістю.

Найбільш поширені методи кругової інтерполяції мають за основу визначення оцінювальної функції [1]-[6], [87], [89]. Рівняння оцінювальної функції ( $OF_i$ ) для кола в загальному вигляді буде:

$$OF_i = (x_i^2 + y_i^2) - R^2,$$

де  $R$  – радіус кола, а  $x$ ,  $y$  - координати центра кола. З формули слідує, що різниця квадратів матиме від'ємне значення для всіх точок в середині кола, додатне значення для точок, що лежать за колом і дорівнюватиме нулю для точок на колі. Така властивість оцінювальної функції дає можливість по знаку  $OF_i$  з урахуванням розташування точки відносно системи координат і напрямом руху, формувати в дискретному координатному просторі одиничний приріст і наблизити траєкторію точки кола до реальної (рис. 1.25). Якщо виконувати послідовні кроки у відповідних ділянках растру, щоб при

цьому змінювався знак оцінювальної функції (з додатного на від'ємний і навпаки), то можна досягти наближення траєкторії інтерполяції до кола, яке відтворюється на растрі.

Найбільш відомий і простий метод генерації кола це алгоритм Брезенхема [1-5], [30], [87]-[89]. В алгоритмі Брезенхема генерується тільки одна восьма частина кола, інші частини отримують шляхом послідовного симетричного відображення згенерованої частини (рис. 1.22).

Більш ефективним є метод Обідника [87], який використовує в обчислювальному процесі більш прості формули.

Як видно з рис. 1.23, на початку генерують перший октант (дугу в проміжку від  $0^\circ$  до  $45^\circ$ ) напрямком генерації – проти часової стрілки. Наступний крок – це “дзеркальне” відображення другого октанту відносно прямої  $y=x$ , і так генерується перший квадрант. Перший квадрант можна відобразити відносно прямої  $x=0$  і отримати відповідну частину кола в другому квадранті. Таким чином, верхнє півколо відображене відносно прямої  $y=0$  завершує формування кола на растрі згідно алгоритму. Простота апаратної реалізації та оперування з цілими числами є перевагами алгоритму Брезенхема. Однак формули для інтерполяції є надлишковими.

Методи формування еліпсів, парабол, гіпербол наведено в [1-6], [88], [89]. В переважній більшості випадків використовується метод оцінювальної функції та метод Ву.

Широке впровадження гексагональних растрів для побудови систем візуалізації обумовили розробку для них методів формування графічних примітивів.

У роботі [90] виконано експериментальні дослідження, які показали, що відрізок прямої, згенерований на гексагональному растрі за допомогою алгоритму Брезенхема на сітці з перемещуванням, є кращим, ніж звичайна реалізація квадратної сітки.

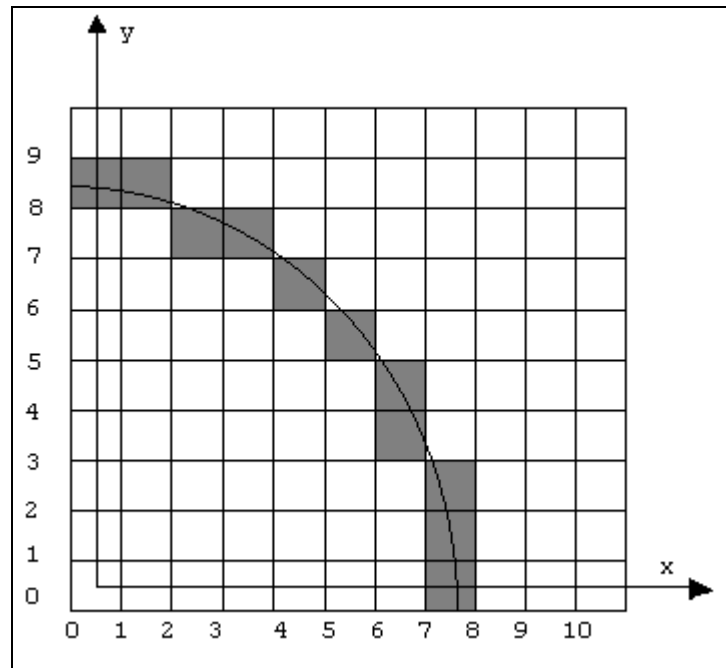


Рисунок 1.22 – Наближена траєкторія кола на растрі

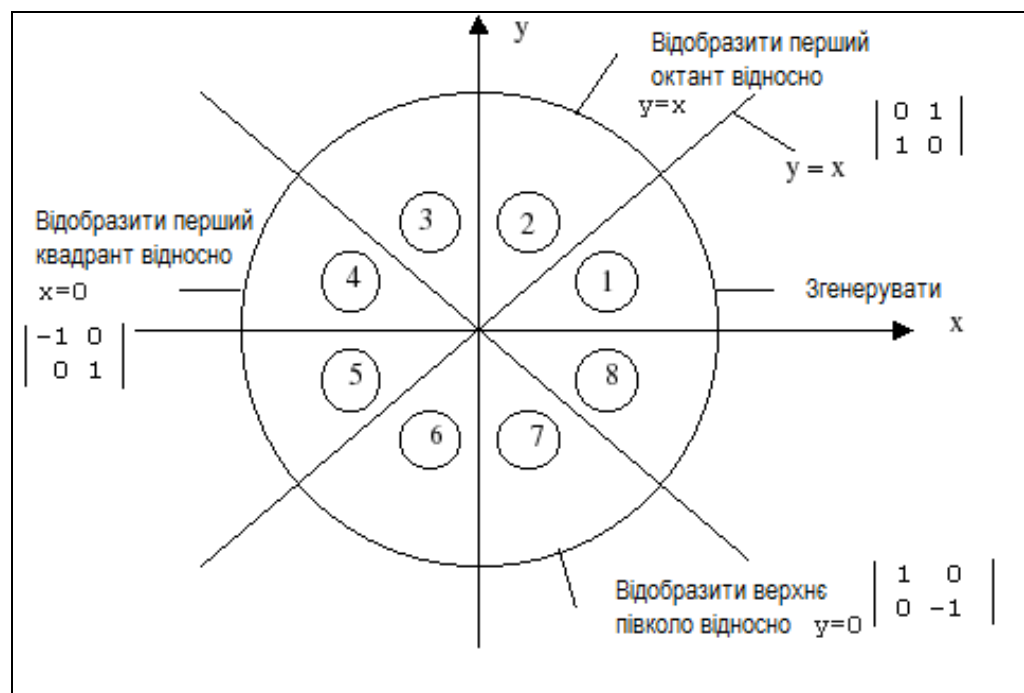


Рисунок 1.23 – Генерація кола за алгоритмом Брезенхема

Розроблено метод [91] лінійної інтерполяції на гексагональному растрі, який базується на формулах Брезенхема, які є надлишковими та передбачають збільшення розрядності для обчислювального процесу.

Один з напрямків [93] досліджень направлений на підвищення точності лінійної інтерполяції, який передбачає виконання додаткових обчислень, що зменшує продуктивність формування векторів. Слід зазначити, що зменшити похибку інтерполяції можливо за рахунок симетричності діагональних кроків в межах цифрових сегментів, з яких складається крокова траєкторія.

Розроблено метод [94] формування траєкторії кола на гексагональному растрі. Метод використовує для кругової інтерполяції формули Брезенхема з адаптацією на гексагональний растр. На жаль, покроковий характер формування траєкторії кола обмежує швидкодію формування графічного примітива.

Запропоновано метод [95] формування траєкторій на гексагональному растрі зображень конічних перетинів для окреслення еліпсів, кіл. Базовий алгоритм вимагає виконання трьох операцій додавання у циклі інтерполяції, хоча для виявлення зміни загального напрямку між двома суміжними секторами потрібен додатковий тест. Великий обсяг обчислень виконується для встановлення типу крокових переміщень залежно від ділянки формування траєкторії.

Розроблено метод [96] кругової інтерполяції на основі визначення кутів. Метод має високу точність, однак потребує для динамічної графіки підвищення швидкодії.

Для формування еліпсів [97], гіпербол [98] на гексагональному растрі найбільшого поширення отримав метод оцінювальних функцій. Це стосується як прямокутного та гексагонального растрів.

Проведений аналіз показує перспективність розробки методів інтерполювання для високопродуктивного формування графічних примітивів.

## 1.5 Висновки до розділу 1

На сучасному етапі розвитку комп'ютерної графіки необхідне підвищення реалістичності сформованих зображень і зменшення часу їх генерації.

Для збільшення роздільної здатності екранів використовують гексагональний растр. При використанні гексагональної моделі пікселя збільшується кількість точок на екрані, що передбачає використання для формування примітивів більшої кількості пікселів, що впливає на швидкодію формування примітивів. Методи інтерполяції на гексагональному растрі більш складні порівняно з використанням квадратного растру. Підвищення продуктивності методів і засобів формування зображень на гексагональному растрі важливо при генерації динамічних графічних зображень і в інтерактивному режимі, коли передбачається, що користувач у реальному часі може впливати на процес формування зображення. Висока швидкодія формування графічних сцен необхідна для гексагональних ігор, які отримали значне поширення. Таким чином, розробка високопродуктивних методів і засобів формування графічних примітивів на гексагональному растрі є актуальною задачею, яка має важливе практичне значення. Метою роботи є підвищення продуктивності та реалістичності формування графічних зображень на гексагональному растрі. Основними задачами дослідження є:

- аналіз сучасних методів і засобів генерації графічних зображень на гексагональному растрі;
- встановити нові аналітичні залежності між параметрами пікселів;
- визначити особливості формування крокових приростів при формування графічних примітивів;
- розробити нові:
  - методи прискореного формування векторів на гексагональному растрі;
  - високопродуктивні методи формування кіл;
  - методи прискореного формування еліпсів;

- методи контурного антиаліазингу зображень графічних примітивів;
- визначити особливості суперсемплінгу графічних зображень, які використовують гексагональну модель пікселя;
- розробити:
  - конвертор зображень з прямокутного в гексагональний растр;
  - графічний редактор для формування графічних зображень на гексагональному растрі;
  - програмні та апаратні засоби для формування графічних зображень на гексагональному растрі;
- провести експериментальні дослідження розроблених моделей і методів.

Результати досліджень цього розділу наведено в публікаціях: [7], [12], [16], [26], [32], [33], [35], [36], [44], [47], [49], [51], [99-103].

## РОЗДІЛ 2

### МЕТОДИ ФОРМУВАННЯ ВЕКТОРІВ НА ГЕКСАГОНАЛЬНОМУ РАСТРІ

#### 2.1 Особливості формування векторів на гексагональному растрі

Відрізки прямих відносять до найпоширеніших примітивів [87], тому продуктивності формування векторів приділяють особливу увагу.

Лінійна інтерполяція за “прямим” методом [30], [87] використовує рівняння відрізка прямої:  $y = (\Delta y / \Delta x) \cdot x$ , де  $\Delta x$  і  $\Delta y$  – відповідно прирости координат відрізка прямої. Оскільки, при виконанні по координаті  $x$  крокового приросту  $x_i = x_i + 1$ , то можна розрахувати відповідні ординати:

$$y_{i+1} = (\Delta y / \Delta x) \cdot x_{i+1} = (\Delta y / \Delta x)(x_i + 1) = (\Delta y / \Delta x)x_i + \Delta y / \Delta x = y_i + \Delta y / \Delta x.$$

Метод забезпечує максимальну точність формування векторів, але використовує «довгі» операції ділення для знаходження  $\Delta y / \Delta x$ .

Розглянемо реалізацію «прямого» методу при використанні гексагонального растру. На рис. 2.1 наведено приклад формування відрізка прямої на гексагональному растрі.

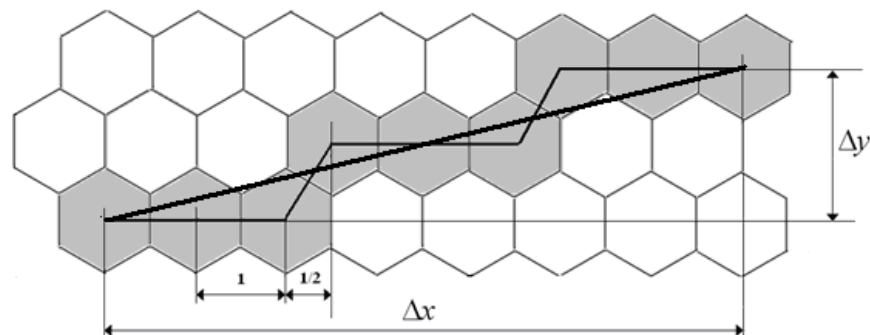


Рисунок 2.1 – Формування відрізка прямої на гексагональному растрі

При використанні гексагонального растру приріст  $\Delta x$  дорівнює кількості горизонтальних переміщень і складових переміщень діагональних кроків, які дорівнюють  $\frac{1}{2}$ .



Визначимо значення ординатного приросту.

Радіус описаного кола дорівнює  $R = \frac{2}{\sqrt{3}}r$ . Оскільки крок

дискретизації вибираємо рівним одиниці (рис. 2.2), то  $r = \frac{1}{2}$ .

Тому  $R = \frac{1}{\sqrt{3}}$ ,  $t = R = \frac{1}{\sqrt{3}}$ .

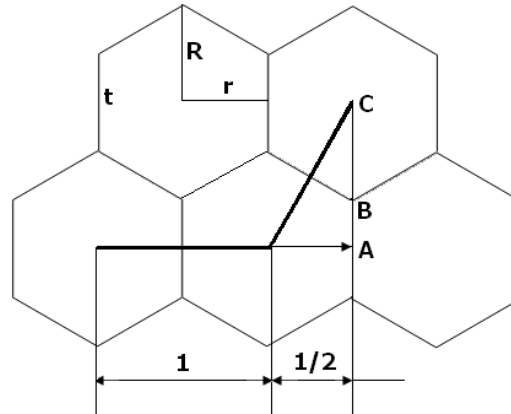


Рисунок 2.2 – Формування крокових переміщень

Знайдемо ординатний приріст діагонального кроку:

$$AC = BA + BC = R + \frac{t}{2} = \frac{1}{\sqrt{3}} + \frac{1}{2\sqrt{3}} = \frac{3}{2\sqrt{3}} = \frac{\sqrt{3}}{2}.$$

Тому приріст  $\Delta y$  дорівнює:

$$z \cdot AC = z \cdot \frac{\sqrt{3}}{2},$$

де  $z$  - кількість діагональних кроків.

Тангенс нахилу відрізка прямої при формуванні відрізка на гексагональному растрі дорівнює:

$$\frac{\Delta y}{\Delta x} = \frac{z \cdot \frac{\sqrt{3}}{2}}{\Delta x} = \frac{z\sqrt{3}}{2\Delta x}.$$

Формула для визначення ординати пікселя крокової траєкторії має такий вигляд:

$$y_{i+1} = (\Delta y / \Delta x) \cdot x_{i+1} = \frac{z\sqrt{3} \cdot x_{i+1}}{2\Delta x}.$$

У випадку формування діагонального кроку відстань між центрами пікселів дорівнює 0,5, тому можливо виконувати інтерполяцію половинними приростами з подальшим об'єднанням половинних горизонтальних кроків.

Можливо виконання і повних горизонтальних приростів з корекцією останніх при виконанні діагонального кроку (див. рис. 2.1).

Точність інтерполювання за «прямим» методом [30], [87] залежить від вибраного методу округлення та може досягти максимальної точності.

Для формування відрізків прямих найчастіше використовують метод оцінювальної функції (*OF*) [1-5], [30], [87]. В цьому випадку для визначення точок траєкторії використовують мікрооперації зсуву, додавання та віднімання. Метод характеризується максимальною точністю лінійного інтерполювання. В цьому випадку абсолютна похибка не перевищує пів кроку дискретизації для прямокутного растру.

За методом, розробленим для формування відрізків прямих, формується спеціальна оцінювальна функція, яка має додатне значення вище прямої та від'ємне - нижче прямої (рис. 2.3). Коли значення цієї оцінювальної функції дорівнює нулю, поточна точка траєкторії вважається частиною відрізка прямої. Визначення точок траєкторії відрізка прямої під час генерації крокової траєкторії орієнтоване на зміні знаку цієї оцінювальної функції на протилежний.

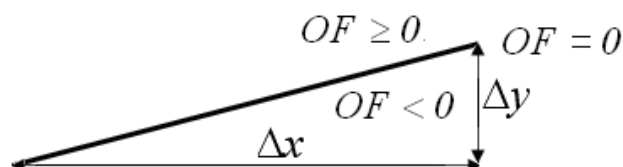


Рисунок 2.3 – Формування оцінювальної функції

Формула для розрахунку оцінювальної функції [1-5], [30], [87] має

такий вигляд:

$$OF_{i+1} = y_i \cdot \Delta x - \Delta y \cdot x_i. \quad (2.1)$$

Метод оцінювальної функції характеризується простотою обчислювального процесу та високою точністю формування крокової траєкторії, а тому використовується в переважній кількості випадків.

При формуванні графічних сцен відрізки прямих мають найбільше поширення, що обумовлює актуальність розробки методів підвищення продуктивності методів лінійної інтерполяції та їх засобів.

При генерації крокової траєкторії векторів за методом оцінювальної функції з нульовим початковим значенням, має місце відносно велика похибка [87]. Це обумовлено тим, що знак  $OF_{i+1}$  змінює своє значення в кінцевій точці цифрового сегменту, довжина якого дорівнює  $\frac{БП}{МП}$  (рис. 2.4), де  $БП$ ,  $МП$  - відповідно більший і менший прирости з  $\Delta x$  і  $\Delta y$ .

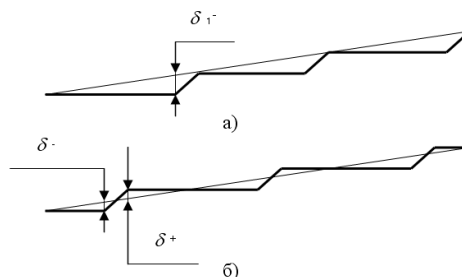


Рисунок 2.4 – Симетрування похибки в межах цифрового сегменту

Похибку можна зменшити, якщо розташувати діагональний крок у середині сегменту (як зображено на рис. 2.4, б). Це досягається шляхом встановлення ненульового значення оцінювальної функції.

Для розміщення діагонального кроку в середині сегменту необхідно задати початкове значення  $OF$ , яке відповідає значенню оцінювальної функції при виконанні  $БП/2$  горизонтальних приростів. Враховуючи, що для основи цифрового сегмента  $\Delta y = 0$ , знайдемо значення  $OF$  при умовному

зміщенні початку вектора на  $K = \frac{БП}{МП} : 2$  горизонтальних кроків.

$$OF_{i+1} = y_i \Delta x - (x_i - \frac{\Delta x}{2}) \Delta y = y_i \Delta x - x_i \Delta y + \frac{\Delta x}{2} = OF_\delta + \frac{\Delta x}{2} = \frac{\Delta x}{2}.$$

В останній формулі  $OF_\delta$  - початкове значення оцінювальної функції, яке дорівнює нулю. Враховуючи, що для першого октанта  $\Delta x = БП$ , то

$OF_0 = \left\lfloor \frac{БП}{2} \right\rfloor$ . Це обумовлено тим, що при непарному  $БП$  діагональний крок

при симетруванні похибки може бути розташовано в  $\frac{БП - 1}{2}$  такті.

Розглянемо особливості виконання горизонтального та діагонального кроків при використанні гексагонального растру (рис. 2.2).

На рис. 2.5 показано приклад генерації вектора, де точка  $B$  розташована на однаковій відстані від вузлів сітки. Таким чином, вибір між точками  $C$  і  $D$  для включення в крокову траєкторію є еквівалентним. У випадках, коли ця умова не виконується, слід вибирати ту точку, яка розташована ближче до відрізка прямої.

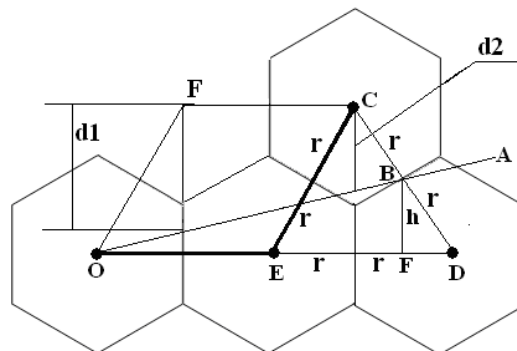


Рисунок 2.5 – Визначення ординарної похибки

Оцінимо максимальні ординатні похибки  $d1$  і  $d2$ . Для цього проаналізуємо  $\Delta ECD$ . У трикутнику всі сторони рівні, тому  $\angle CDE = 60^\circ$ . З трикутника  $BDF$  знаходимо висоту:

$$h = BD \cdot \sin 60^\circ = r \cdot \sin 60^\circ = \frac{1}{2} \cdot \sin 60^\circ = \frac{\sqrt{3}}{4} = 0,433.$$

З рис. 2.5 видно, що  $h < d/2$ .  $d/2 < R + \frac{t}{2} = \frac{3R}{2}$ , що менше висоти діагонального кроку.

Отримані співвідношення можна використати для розробки методів інтерполяції на гексагональному растрі.

## 2.2 Формування відрізків прямих на гексагональному растрі

Розглянемо формування на гексагональному растрі векторів за методом оцінювальної функції.

Формула оцінювальної функції для формування відрізка прямої, для якого відомі прирости  $\Delta$  по осях координат  $X$ ,  $Y$ , має такий вигляд:

$$OF_i = \frac{y_i}{x_i} - \frac{\Delta y}{\Delta x} = \frac{y_i \Delta x - x_i \Delta y}{x_i \Delta x}. \quad (2.2)$$

Щоб визначити, по якій з осей  $OX$  чи  $OY$  виконати наступний крок інтерполяції, знаходять знак  $OF_i$  (рис. 2.6).

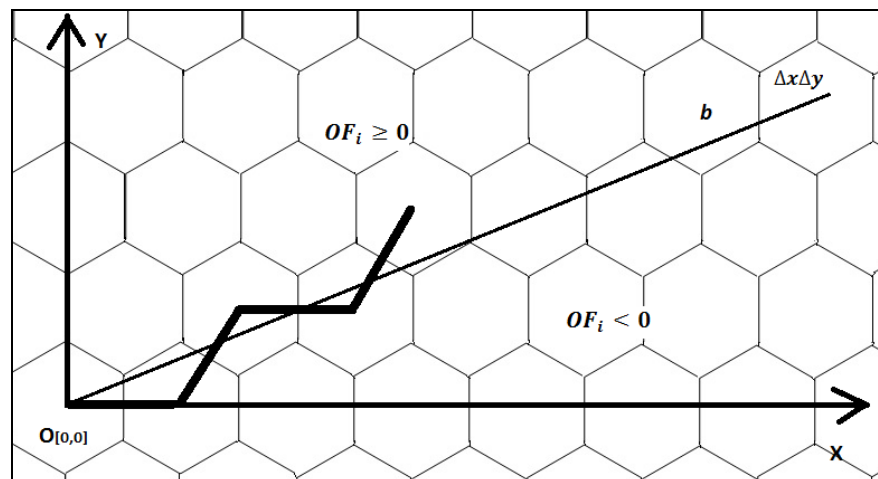


Рисунок 2.6 – Формування відрізка прямої кроковими приростами

За умови, що точка траєкторії знаходиться вище вектора  $b$ , оцінювальна функція  $OF_i \geq 0$  і наступний крок треба виконувати вздовж осі

$OX$ . Якщо точка траєкторії знаходиться нижче вектора  $b$ , то  $OF_i \leq 0$  і наступний крок треба реалізувати по обох осях одночасно (діагональний крок).

Оскільки в формулі 2.2 знаменник дробу  $x_i \Delta x$  не впливає на знак оцінювальної функції, то  $OF_i = y_i \Delta x - x_i \Delta y$ .

Знайдемо значення оцінювальної функції при виконанні горизонтального переміщення по осі  $OX$ , тобто  $x_{i+1} = x_i + 1$ .

$$OF_{i+1} = y_i \Delta x - \Delta y (x_i + 1) = y_i \Delta x - x_i \Delta y - \Delta y = OF_i - \Delta y.$$

Аналогічно знайдемо нове значення  $OF_{i+1}$  при виконанні діагонального кроку. В цьому випадку координат по осі абсцис збільшиться на  $\frac{1}{2}$ , а по осі

ординат на  $\frac{3}{2\sqrt{3}}$  (рис. 2.7).

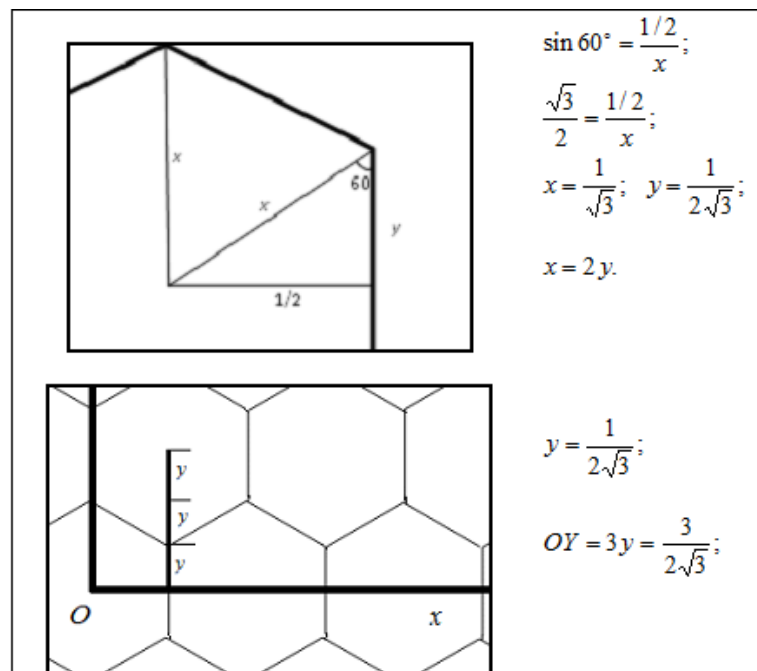


Рисунок 2.7 – Розрахунок значення координати по осі  $OY$

Як показано на рис. 2.8, розрахунок координат ведеться з урахуванням того, що відстань між центрами сусідніх пікселів дорівнює одиниці. При цьому, під час діагонального переміщення, зміни координат відбуваються

так: координата по осі ОХ збільшується на половину одиниці, тоді як зміна координати по осі ОУ на  $\frac{3}{2\sqrt{3}}$ .

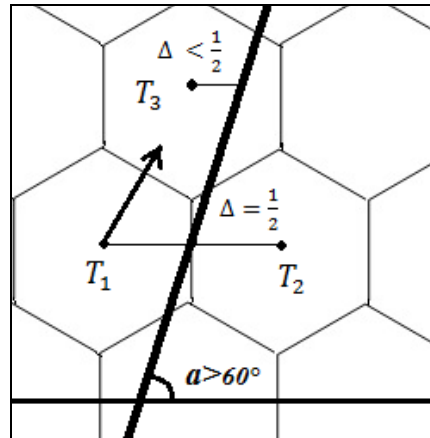


Рисунок 2.8 – Визначення діагонального кроку

Враховуючи це, нове значення оцінювальної функції визначають за виразом:

$$\begin{aligned} OF_{i+1} &= \left(y_i + \frac{3}{2\sqrt{3}}\right)\Delta x - \left(x_i + \frac{1}{2}\right)\Delta y = \\ &= \Delta x y_i + \Delta y x_i - \frac{\Delta y}{2} = OF_i + \frac{3\Delta x}{2\sqrt{3}} - \frac{\Delta y}{2}. \end{aligned}$$

Знайдені формули справедливі для інтерполяції відрізків прямих з кутами нахилу від  $0^\circ$  до  $60^\circ$  по відношенню до осі ОХ.

Формули, які були наведені, справедливі для інтерполяції векторів з нахилом від  $0^\circ$  до  $60^\circ$  відносно осі ОХ. У випадку генерації векторів з нахилом між  $60^\circ$  та  $90^\circ$ , елементарні інтерполяційні кроки вздовж осі ОХ не формуються. Для наведеної ділянки кроки виконуються одночасно по обох осях, тобто у діагональному напрямку. Розглядаємо елементарне переміщення для лінійного вектору, нахил якого до осі ОХ перевищує  $60^\circ$  (див. рис. 2.8).

Для точки  $T_1$  найближчими сусідніми пікселями будуть пікселі з

центрами  $T_2$  і  $T_3$ . Максимальне відхилення траєкторії від відрізка прямої у випадку вибору сусіднього пікселя з центром  $T_2$  буде більшим, ніж у випадку вибору сусіднього пікселя з центром  $T_3$ . Для кращого наближення траєкторії до відрізка прямої доцільною буде реалізація діагонального кроку. Тому, для векторів з кутами нахилу від  $60^\circ$  до  $90^\circ$  кроки переміщення будуть тільки діагональні. Знайдемо вирази для визначення  $OF_{i+1}$  при формуванні векторів з кутами від  $60^\circ$  до  $90^\circ$  (рис. 2.9).

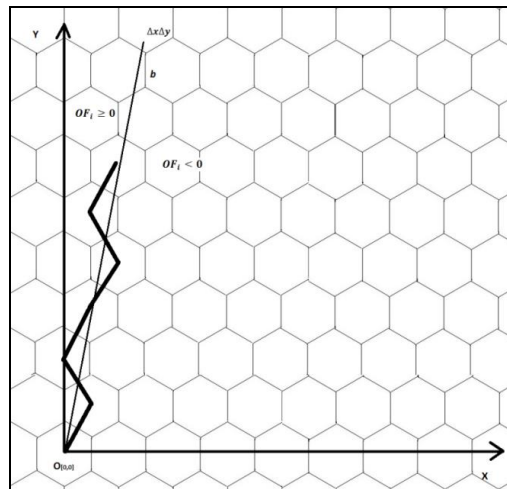


Рисунок 2.9 – Покрокове формування відрізка прямої для кутів  $60^\circ \div 90^\circ$

Для  $OF_i \geq 0$  координата по ОХ буде змінюватись за формулою:

$$x_{i+1} = x_i + \frac{1}{2}, \text{ а по } OY - y_{i+1} = y_i + \frac{3}{2\sqrt{3}};$$

$$\begin{aligned} OF_{i+1} &= \Delta x \left( y_i + \frac{3}{2\sqrt{3}} \right) - \Delta y \left( x_i + \frac{1}{2} \right) = \\ &= \Delta x y_i + \frac{3\Delta x}{2\sqrt{3}} - \Delta y x_i - \frac{\Delta y}{2} = OF_i + \frac{3\Delta x}{2\sqrt{3}} - \frac{\Delta y}{2}. \end{aligned}$$

$$\text{Для } OF_i < 0 \quad x_{i+1} = x_i - \frac{1}{2}, \text{ а по } OY \quad y_{i+1} = y_i + \frac{3}{2\sqrt{3}}.$$

Тому:



$$\begin{aligned}
 OF_{i+1} &= \Delta x \left( y_i + \frac{3}{2\sqrt{3}} \right) - \Delta y \left( x_i - \frac{1}{2} \right) = \\
 &= \Delta x y_i + \frac{3\Delta x}{2\sqrt{3}} - \Delta y x_i + \frac{\Delta y}{2} = OF_i + \frac{3\Delta x}{2\sqrt{3}} + \frac{\Delta y}{2}
 \end{aligned}$$

Для спрощення обчислювального процесу представимо значення  $\frac{3}{2\sqrt{3}}$

сумою операндів, знаменники яких є степені двійки:

$$\frac{3\Delta x}{2\sqrt{3}} \approx \left( \frac{1}{2} + \frac{1}{4} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{256} + \frac{1}{512} \right) \Delta x.$$

Такий підхід дозволить використовувати в розрахунках мікрооперації додавання та зсуву в сторону молодших бітів. Встановлено, що відносна помилка обчислень не перевищить значення  $\delta=0,09142\%$

### 2.3 Формування відрізків прямих комбінованими приростами на гексагональному растрі

Підвищити продуктивність формування векторів на гексагональному растрі можливо при одночасному визначенні в кожному інтерполяційному такті відразу двох точок траєкторії, а не лише одну. Визначимо нахил вектора, який складається з горизонтальних та суміщених з ним діагональних кроків:

$$\operatorname{arctg}(AC : 1\frac{1}{2}) = \operatorname{arctg} \frac{\sqrt{3}}{3} = 30^{\circ}.$$

Нахил вектора, який включає тільки тільки діагональні кроки, дорівнює

$$\operatorname{arctg}(AC : \frac{1}{2}) = \operatorname{arctg} \frac{2\sqrt{3}}{2} = 60^{\circ}.$$

Вказані межі виступають як крайні для секторів (рис. 2.10). Отже, на секторі Д1 неможливе формування двох поруч розташованих діагональних кроків за умови забезпечення найвищої точності інтерполяції. Для сектора Д2 подібне обмеження є характерним для двох сусідніх горизонтальних кроків.

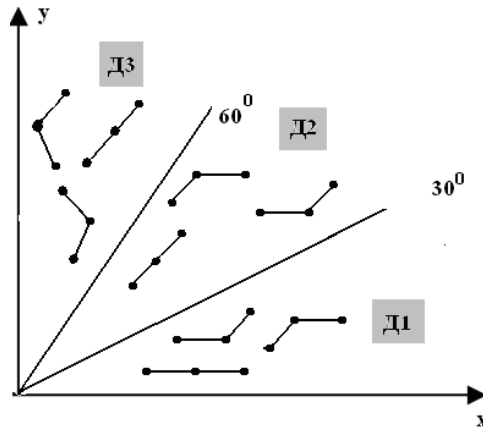


Рисунок 2.10 – Типи допустимих сполучень крокових переміщень

Розглянемо формування відрізків прямих з діапазону від  $60^\circ$  до  $90^\circ$  проти годинникової стрілки. Для цього випадку формування комбінованих горизонтальних і вертикальних крокових приростів неможливо. На рис. 2.11 зображено ліве та праве діагональне переміщення (рис. 2.11).

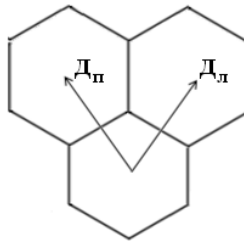


Рисунок 2.11 – Ліве та праве діагональне крокове переміщення

Для діагонального кроку:

$$x = x + \frac{1}{2}, y = y + \frac{\sqrt{3}}{2}.$$

Для діагонального кроку  $Дп$  можна записати:

$$x = x - \frac{1}{2}, y = y + \frac{\sqrt{3}}{2}.$$

Доведемо, що для відрізків прямих у секторі від  $60^\circ$  до  $90^\circ$  (сегмент Д3) неможливо сформувати два діагональні кроки  $Дп$  для відтворення

відрізків з мінімальною похибкою для гексагонального растру.

На рис. 2.12 зображено приклад формування діагональних крокових переміщень.

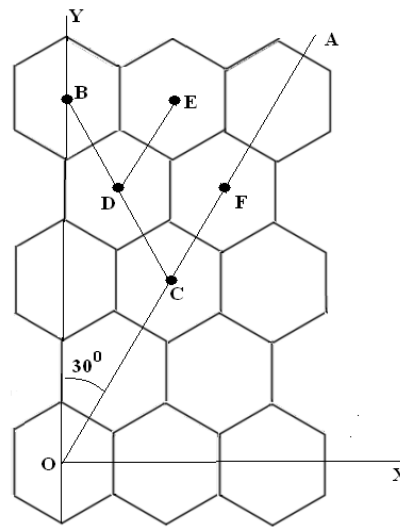


Рисунок 2.12 – Приклад реалізації діагональних переміщень

Розглянемо випадок проходження вектора через точку  $C$ , за умови, що кут  $XOA$  дорівнює  $30^\circ$ . Це граничний випадок для аналізованої ділянки. Розглянемо точку  $C$ , яка розташована на однаковій відстані від точок  $D$  і  $F$ . У цьому випадку можна сформулювати еквівалентні діагональні переміщення  $DC$  і  $CF$ . Якщо вибрано переміщення  $DC$ , то серед наступних можливих переміщень  $DE$  і  $DE$  необхідно виконати  $DE$ , оскільки його розташовано ближче до вектора. У іншому випадку за вектором буде сформовано дві точки, що віддаляються від вектора.

Це неприпустимо згідно методу  $OF$ , який забезпечує наближення до вектора. При наближенні променя  $OA$  до осі  $OY$  така ситуація погіршується. Таким чином, при генерації векторів з ділянки від  $60^\circ$  до  $90^\circ$  формування двох послідовних діагональних переміщень недопустимо.

На рис. 2.13 показано приклад, коли вектор  $OA$  проходить через середину двох поруч розташованих бокових сторін шестикутника. Зрозуміло, що в цьому випадку комбінації діагональних переміщень  $(OB, DB)$  і  $(OC, CD)$  рівнозначні, тому що  $CT=TB$ . Якщо нахил сегмента  $OA$  зменшується, точка  $B$  рухається ближче до вектора, тоді як точка  $C$  віддаляється. У цьому

випадку максимальне відхилення вектора  $OA$  від точки  $C$  не перевищить  $CB=1$ .

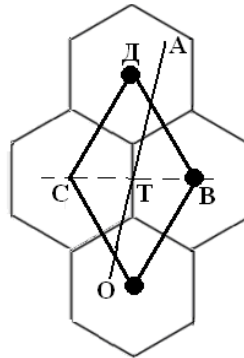


Рисунок 2.13 – Приклад формування діагональних кроків

З рис. 2.13 видно, що максимальне ординатне відхилення не перевищує  $R + \frac{t}{2} = \frac{3R}{2}$ , а саме висоти діагонального кроку.

Визначемо умови розміщення векторів в секторах  $D1, D2, D3$ .

Як було показано раніше, коли вектор нахилений до осі  $OX$  під кутом  $30^\circ$ , то його тангенс кута приймає значення  $\frac{\sqrt{3}}{3}$ .

Для всіх векторів з кутом нахилу до  $30^\circ$  до осі  $OX$ :

$$\frac{\Delta y}{\Delta x} \leq \frac{\sqrt{3}}{3}.$$

Праву частину останньої нерівності запишемо так:

$$\frac{\sqrt{3}}{3} \approx \frac{1}{2} + \frac{1}{16}.$$

У даному випадку рівень відносної похибки становить 2,5 відсотка.

Враховуючи зазначене наближення, запишемо нерівність наступним чином:

$$\frac{\Delta y}{\Delta x} \leq \frac{9}{16}.$$

Щоб спростити реалізацію, запишемо нерівність так:

$$16 \cdot \Delta y \leq 8 \cdot \Delta x + \Delta x.$$

У наведеній нерівності множники представлені степенями двійки, що дозволяє виконати мікрооперацію множення за допомогою зсуву та додавання.

Коли кут нахилу відрізка прямої складає з віссю  $OX$   $60^\circ$ , то його тангенс дорівнює  $\sqrt{3}$ . Для спрощення розрахунків  $\sqrt{3}$  задамо сумою ступенів двійки:  $\sqrt{3} \approx 1 + \frac{1}{2} + \frac{1}{4}$ . У такому випадку відносна похибка обчислень становить всього 1,03 %.

У випадку, коли кут нахилу вектора до осі  $OX$  складає  $60^\circ$ :

$$\frac{\Delta y}{\Delta x} \leq 1 + \frac{1}{2} + \frac{1}{4}.$$

Цю нерівність можна представити так:

$$4 \cdot \Delta y \leq 7 \cdot \Delta x.$$

Щоб спростити обчислення, останню нерівність можна представити так:

$$4 \cdot \Delta y \leq 8 \cdot \Delta x - \Delta x.$$

В даному випадку для проведення порівняння застосовуються мікрооперації зсуву та додавання.

Якщо вектор не розміщено на ділянках  $D1$  і  $D2$ , то він належить ділянці  $D3$ .

Перед початком формування векторів слід аналізувати зазначені умови під час циклу підготовки.

Розглянемо формування оцінювальної функції для сектора  $D1$ . При виконанні двох горизонтальних кроків:

$$OF_{i+1} = y_i \Delta x - \Delta y (x_i + 2) = y_i \Delta x - x_i \Delta y - 2 \Delta y = OF_i - 2 \Delta y.$$

При формуванні горизонтального та діагонального кроків:

$$\begin{aligned}
OF_{i+1} &= (y_i + \frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i + \frac{1}{2} + 1) = y_i\Delta x - x_i\Delta y + \frac{\sqrt{3}}{2}\Delta x - \frac{3}{2}\Delta y = OF_i + \frac{\sqrt{3}}{2}\Delta x - \frac{3}{2}\Delta y \\
&= (\frac{1}{2} + \frac{1}{16} + \frac{1}{64})\Delta x - (1 + \frac{1}{2})\Delta y = \frac{37}{64}\Delta x - \Delta y - \frac{1}{2}\Delta y = \frac{32\Delta x + 4\Delta x + \Delta x}{64} - \frac{2\Delta y + \Delta y}{2}.
\end{aligned}$$

Розглянемо формування  $OF_{i+1}$  для ділянки  $D2$ .

У випадку виконання двох діагональних кроків:

$$\begin{aligned}
OF_{i+1} &= (y_i + \frac{2\sqrt{3}}{2})\Delta x - \Delta y(x_i + 1) = y_i\Delta x - x_i\Delta y + \sqrt{3}\Delta x - \Delta y = OF_i + \sqrt{3}\Delta x - \Delta y = \\
&= OF_i + \frac{8-1}{4}\Delta x - \Delta y = OF_i + \frac{8\Delta x - \Delta x}{4} - \Delta y.
\end{aligned}$$

При генерації векторів в секторі  $D3$  використовуються комбінації двох діагональних переміщень і сполучення лівого та правого діагональних кроків. В цьому випадку:

$$OF_{i+1} = (y_i + \frac{2\sqrt{3}}{2})\Delta x - x_i\Delta y = y_i\Delta x - x_i\Delta y + \sqrt{3}\Delta x = OF_i + \sqrt{3}\Delta x.$$

Розглянемо закінчення процесу інтерполяції векторів, який відбувається після виконання всіх крокових приростів. Проаналізуємо випадок формування векторів елементарними приростами. Для сектора  $D1$  формування вектора закінчується після видачі вихідному блоку всіх кроків вздовж провідної координати.

За умови, що більший приріст по провідній координаті має значення  $\Delta x = v$ , то після виконання горизонтального переміщення (рис. 2.14, а), цей операнд зменшують на 1, а після виконання діагонального – на 0,5. При досягненні нульового значення формування відрізка прямої закінчують.

При лінійній інтерполяції використовують для віднімання від  $v$  2 або 1,5 відповідно при поєднанні двох горизонтальних переміщень або горизонтального і діагонального переміщення відповідно (див. рис. 2.14, а).

Інтерполяцію припиняють, коли значення операнда  $v$  досягає нуля.

У випадку непарної кількості кроків (коли результат віднімання є від'ємним), останній крок в подвійному переміщенні блокується.

Для сектора  $D2$  рис. 2.14, б, як і для  $D1$   $\Delta x \geq \Delta y$ . При лінійній інтерполяції для цього випадку використовуються такі комбінації крокових переміщень: два діагональні кроки; горизонтальний і діагональні кроки.

При виконанні двох діагональних кроків  $v$  зменшують на одиницю.

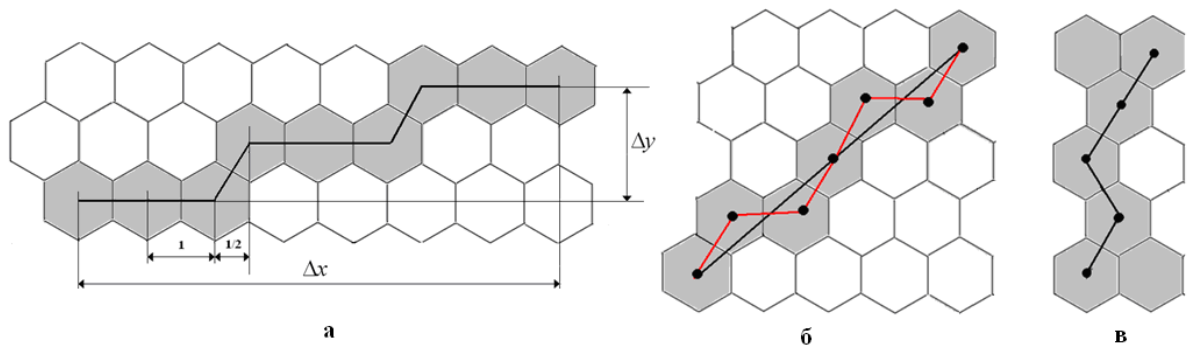


Рисунок 2.14 – Формування векторів в секторах  $D1$ ,  $D2$ ,  $D3$

Для сектора  $D3$   $\Delta y > \Delta x$ , тому  $v = \Delta y$ .

Одним із шляхів підвищення швидкодії формування векторів на гексагональному растрі полягає в незалежній генерації парних та непарних точок траєкторії відрізка прямої. У такому разі крокову траєкторію формують два лінійних інтерполятори. Ці пристрої можна реалізувати ядрами відеокарти.

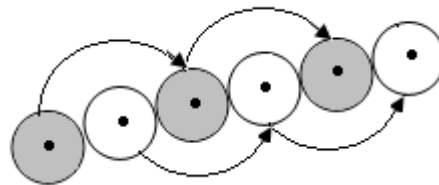


Рисунок 2.15 – Визначення парних і непарних точок траєкторії відрізка прямої

При визначенні непарних точок траєкторії відрізка прямої відповідний інтерполятор використовує перед інтерполяцією таке значення -  $OF_0 = \lfloor BP/2 \rfloor$ . Для інтерполятора, який відповідає за парні точки, початкове значення функції  $OF_0$  визначається як сума  $\lfloor BP/2 \rfloor$  і значення оцінювальної функції у першій точці траєкторії. Якщо  $BP$  відрізка прямої





похибку всередині цифрового сегменту.

5. Для високопродуктивної генерації крокових траєкторій векторів рекомендується використовувати дві окремі оцінювальні функції: одну для генерації парних точок траєкторії, а іншу - для непарних точок.

Результати досліджень цього розділу наведено в публікаціях: [35], [37], [41], [51], [48], [103-105].

## РОЗДІЛ 3

### МЕТОДИ ФОРМУВАННЯ КІЛ НА ГЕКСАГОНАЛЬНОМУ РАСТРІ

#### 3.1 Визначення типів крокових приростів при побудові кола на гексагональному растрі

Алгоритми оцінювальної функції для побудови кола основані на аналізі знаку оцінювальної функції. При круговій інтерполяції на прямокутному растрі максимальна похибка не перевищує пів кроку дискретизації (рис. 3.1).

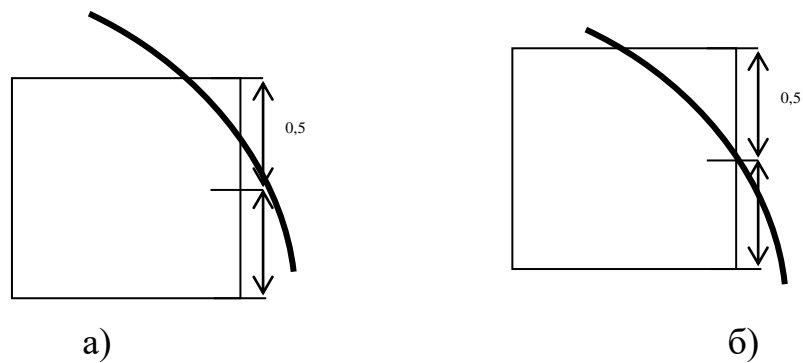


Рисунок 3.1 – Максимальна похибка 0,5 кроку при інтерполяції кола

Типи крокових приростів при формуванні кола на прямокутному растрі з максимальною точністю інтерполяції зображено на рис. 3.2.

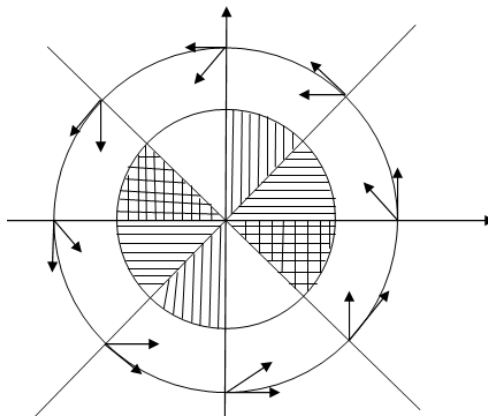


Рисунок 3.2 – Типи кроків для формування кола

Доведено [87], що для генерації кола з максимальною точністю використовують такі елементарні переміщення: горизонтальні, вертикальні, діагональні. Якщо застосовувати лише горизонтальні та вертикальні кроки,

максимальна абсолютна похибка може досягти кроку дискретизації.

При використанні гексагонального растру відстані між горизонтальними та вертикальними пікселями різні [35] (рис. 3.3).

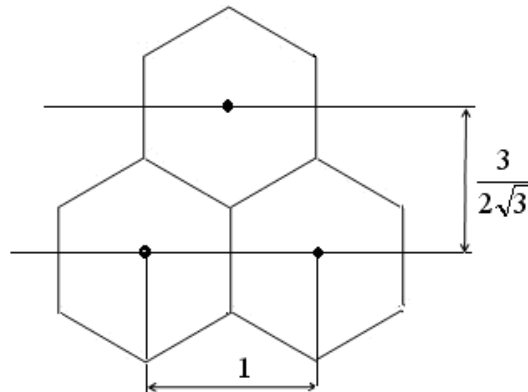


Рисунок 3.3 – Відстані між пікселями на гексагональному растрі

Встановимо типи елементарних кроків залежно від ділянок формування крокової траєкторії на екрані.

Нехай потрібно відобразити на гексагональному растрі коло радіусом  $R$  з центром  $O(x_0, y_0)$ . Розглянемо спочатку ділянку кола, що відповідає дузі

$\theta \in \left[0, \frac{\pi}{6}\right]$ , де  $\theta$  – кут, утворений радіус-вектором кола з додатним напрямком

осі  $x$  декартової системи координат. У подальшому напрямком вектора визначає кут  $\theta$  між цим вектором і додатним напрямком осі  $x$ . Для даної

ділянки кола вектор дотичної  $\vec{t}$  до кола утворює кути  $\theta \in \left[\frac{\pi}{2}, \frac{2\pi}{3}\right]$

(рис. 3. 4).

Вузлом назвемо центр гексагонального пікселя. Слід зазначити, що сусідні вузли розташовані на відстані, що дорівнює 1. Відстанню від кола до пікселя назвемо найкоротшу відстань від вузла до траєкторії. Відстань є додатною, за умови, що вузол знаходиться праворуч від дотичної до лінії і від'ємною, якщо ліворуч.

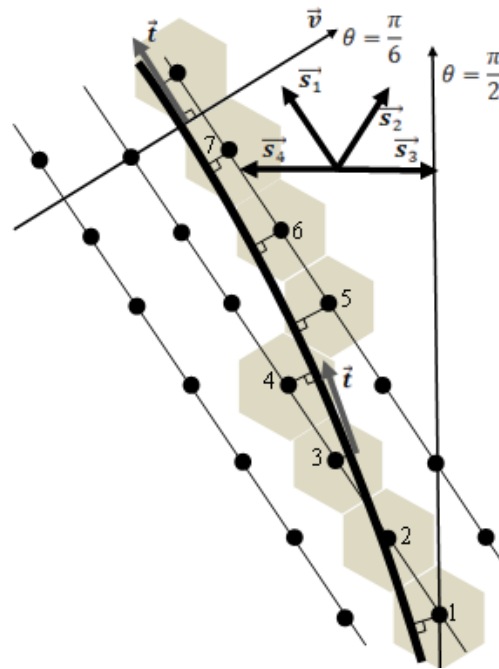


Рисунок 3.4 – Дуга кола на гексагональному рістрі

При відображенні кола на гексагональному растрі формуються лише ті пікселі, для яких відстань до траєкторії найменша. Якщо крива неперервна, то пікселі генеруються неперервно, тобто, для кожного пікселя обов'язково має бути сформований лише один з шести сусідніх.

Доведемо, що при виборі будь-якого пікселя в гексагональній мережі (як стартової точки), формувати пікселі, що відображають задану дугу кола  $\theta \in \left[0, \frac{\pi}{6}\right]$ , можна при русі від стартової точки у напрямку годинникової стрілки, переходячи від пікселя до сусіднього пікселя з використанням лише переходів у напрямках  $\vec{s}_1$  (що відповідає  $\theta = \frac{2\pi}{3}$ ) і  $\vec{s}_2$  (що відповідає  $\theta = \frac{\pi}{3}$ ), причому перехід у напрямку  $\vec{s}_2$  не може відбуватися двічі підряд.

**Доведення.** Дотична  $\vec{t}$  до дуги кола  $\theta \in \left[0, \frac{\pi}{6}\right]$  завжди утворює гострий кут з напрямком.

1. За умови, що вузол розміщено з правого боку від кола (додатна

відстань), то при переході до сусіднього вузла у напрямку вектора  $\vec{S}_1$ , відстань від вузла до траєкторії буде зменшуватись до тих пір, доки пряма, що проходить через зазначені вузли, не перетне дугу кола. Це має місце для послідовностей вузлів 1, 2 і 5, 6, 7 (див. рис. 3.4). При русі у всіх інших напрямках  $\vec{S}_2, \vec{S}_3, \vec{S}_4$  відстань від вузла до дуги кола буде збільшуватися. В цьому випадку ці переміщення мають більші кути з напрямком  $\vec{t}$ .

Розглянемо рис. 3.4. Рух у напрямках  $\vec{S}_2, \vec{S}_3$  зі стартового вузла (0) віддаляється від лінії (враховано взаємне розміщення та можливі кути між векторами  $\vec{t}, \vec{S}_2, \vec{S}_3$ ).

Для руху у напрямку  $\vec{S}_4$  можливі такі випадки: а) виконано переміщення у вузол (0) з вузла (-2) (напрямок  $\vec{S}_1$ ), тоді подальший рух у цьому напрямку до вузла (1) буде тільки зменшувати відстань до дуги; б) виконано переміщення у вузол (0) з вузла (-1) (напрямок  $\vec{S}_2$ ). Як було показано раніше, відстань до дуги від вузла (0) менша за відстань від вузла (4).

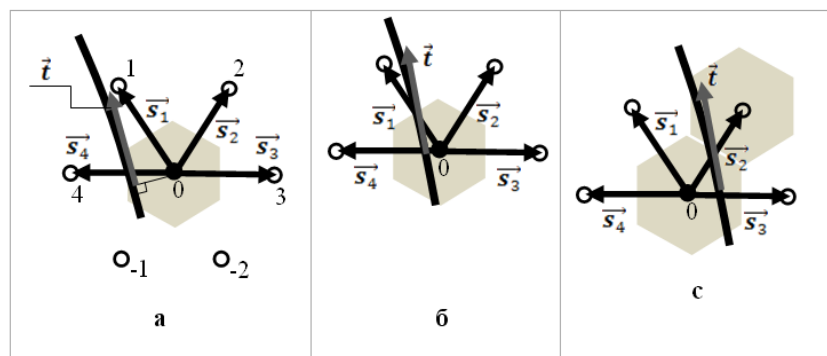


Рисунок 3.5 – Вибір крокових переміщень

Якщо дуга кола перетинає відрізок, що з'єднує сусідні вузли у напрямку  $\vec{S}_1$  (рис. 3.5, б), то вузол, що знаходився зліва від дуги, перейде (в

напрямку  $\vec{S}_1$ ) у вузол, що буде розташований справа від дуги, причому відстань до дуги буде завжди менше за 0.5. Те, що відстань мінімізується у напрямку  $\vec{S}_1$  випливає з того, що вектори  $\vec{S}_2, \vec{S}_3, \vec{S}_4$  утворюють з напрямком дотичної  $\vec{t}$  більші кути ( послідовність вузлів 2, 3).

2. За умови, що вузол знаходиться зліва від кривої, то можливі такі випадки:

а) при переміщенні в напрямку  $\vec{S}_1$  новий вузол залишиться найближчим до дуги (див. послідовність вузлів 3, 4.). Переміщення в напрямках  $\vec{S}_3, \vec{S}_4$  неможливі, так як вони утворюють більші кути з дотичною  $\vec{t}$ .

б) якщо при переході у напрямку  $\vec{S}_1$  вузол не буде найближчим до дуги кола, то перехід у напрямку  $\vec{S}_2$  дасть найближчий вузол, так як відстань від такого вузла до кола буде завжди меншою за 0.5 (див. послідовність вузлів 3, 5). При цих умовах перехід  $\vec{S}_4$  забезпечить більшу похибку, а перехід до сусіда праворуч (напрямок  $\vec{S}_3$ ) дає відстань, що перевищує граничне значення 0.5. Так, у найкращому випадку, значення близькі до граничного 0.5 при переході праворуч можливі лише, коли дотична  $\vec{t}$  до дуги утворює кут  $\theta = \frac{\pi}{2}$ , і дуга проходить по дотичній через праву границю гексагону. Але і тоді дуга не перетинає правий сусідній піксель, тому відстань від правого вузла більше за 0.5 і не є найменшою.

Варто відзначити, що два послідовні переходи в напрямку  $\vec{S}_2$  є неможливими. Це пов'язано з тим, що перший перехід здійснюється, як у випадку, зображеному на рис. 3.5б. Коли дуга розташована праворуч від вузла, то другий перехід лише збільшить відстань від дуги на величину, яка

перевищує допустимий максимум у 0.5.

Розглянемо тепер дугу кола, що відповідає  $\theta \in \left[-\frac{\pi}{6}, 0\right]$ . Вона симетрична дузі  $\theta \in \left[0, \frac{\pi}{6}\right]$ , тому у при русі кола проти годинникової стрілки можливі напрямки переходу  $\overrightarrow{S_1}$  і  $\overrightarrow{S_2}$  зберігаються.

Об'єднаємо отримані результати. Для дуги кола  $\theta \in \left[-\frac{\pi}{6}, \frac{\pi}{6}\right]$  напрямками побудови кола є виключно напрямки переходу  $\overrightarrow{S_1}$ ,  $\overrightarrow{S_2}$ , причому два переходи у напрямку  $\overrightarrow{S_2}$  виконати послідовно неможливо.

Виконаємо поворот дуги кола  $\theta \in \left[-\frac{\pi}{6}, \frac{\pi}{6}\right]$  на кут  $\frac{\pi}{3}$  в напрямку годинникової стрілки. Як результат отримаємо дугу  $\theta \in \left[\frac{\pi}{6}, \frac{\pi}{2}\right]$ , причому напрямки переходу  $\overrightarrow{S_1}$ ,  $\overrightarrow{S_2}$  перейдуть, відповідно, у напрямки, що відповідають кутам  $\theta = \frac{2\pi}{3}$ ,  $\theta = \pi$ .

Скористаємося симетрією кола відносно напрямків, які паралельні осям декартової координатної системи.

На основі попередніх досліджень можна зробити висновок, що в кожному з секторів кола  $\theta \in \left[-\frac{\pi}{6} + \frac{\pi}{3}n, \frac{\pi}{6} + \frac{\pi}{3}n\right]$ ,  $n = 0..5$ , де  $n = 0..5$ , існують лише два можливі напрямки переходу. При цьому, у кожному з цих напрямків виконати два послідовні переходи є неможливим.

Проведені дослідження про можливі типи крокових приростів дозволяє підвищити продуктивність наявних методів [35], [94-96] кругової інтерполяції на гексагональному растрі.

### 3.2 Формування кіл за методом оцінювальної функції

При інтерполяції кола на гексагональному растрі є відмінності у формуванні крокових приростів порівняно з прямокутним растром.

Розглянемо процес генерації кола з центром у точці початку координат. Будемо інтерполювати коло проти годинникової стрілки, починаючи з точки на колі з координатами  $y=0, x=R$ , де  $R$  – радіус кола. Припустимо, що центр кола та початкова точка інтерполяції розміщені точно у центрах відповідних пікселів растру (див. рис. 3.6).

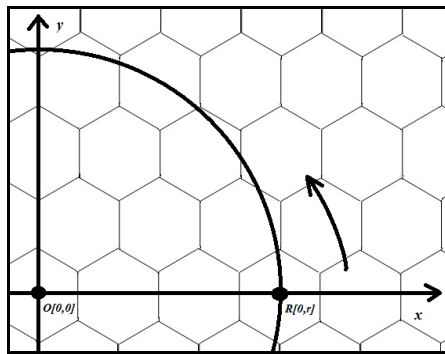


Рисунок 3.6 – Початкова точка і напрям інтерполювання кола в першому квадранті

За умови, що  $y=0$  і  $x=R$  (рис. 3.6), при формуванні кола в напрямку проти годинникової стрілки, значення  $x$  буде монотонно зменшуватися відносно зростання аргументу  $y$ .

Для будь-якої точки на колі, при інтерполяції проти годинникової стрілки, існує тільки три можливі переходи, що найкраще наближають коло (рис. 3.7).

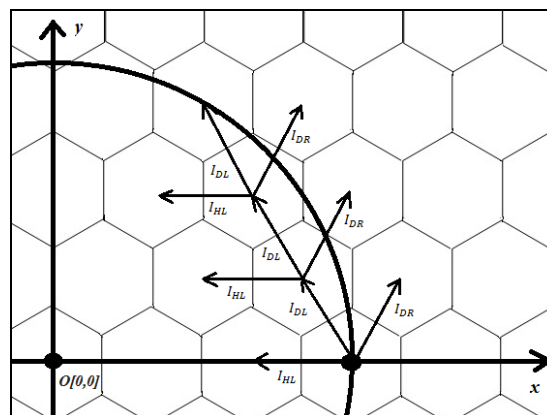


Рисунок 3.7 – Кроки інтерполяції



При лінійному інтерполюванні можливими кроками інтерполяції є горизонтальний крок вліво, діагональний крок вліво та діагональний крок вправо.

На рис. 3.7 зображено кроки інтерполяції. Крок  $I_{HL}$  - горизонтальний вліво, крок  $I_{DL}$  - діагональний вліво, крок  $I_{DR}$  - діагональний вправо.

При формуванні послідовності кроків для траєкторії кола у першому октанті, у секторі між  $0^\circ$  та  $30^\circ$ , використання діагональних кроків забезпечує найменшу помилку [38].

При формуванні горизонтального переміщення ординатна похибка інтерполяції буде перевищувати половини кроку дискретизації.

Для формування крокової траєкторії в першому октанті (сектор від  $0^\circ$  до  $30^\circ$ ) згенеруємо ділянку кола, що відповідає дузі  $\theta \in \left[0, \frac{\pi}{6}\right]$  де  $\theta$  – кут, між радіусом-вектора кола та додатним напрямком осі абсцис. У цьому випадку для відтворення крокової траєкторії формують виключно діагональні кроки інтерполяції: крок  $I_{DL}$  - діагонально вліво, крок  $I_{DR}$  - діагонально вправо (рис. 3.8).

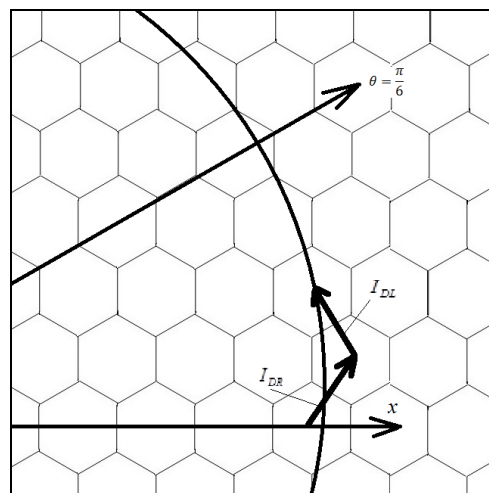


Рисунок 3.8 – Інтерполяція дуги кола в секторі від  $0^\circ$  до  $30^\circ$

При круговій інтерполяції доцільно вибрати той піксель, для якого квадрат відстані між пікселем і точкою на колі буде мінімальним:

$$I_{DL} = \left| \left( x_i - \frac{1}{2} \right)^2 + \left( y_i + \frac{3}{2\sqrt{3}} \right)^2 - R^2 \right|$$

$$I_{DR} = \left| \left( x_i + \frac{1}{2} \right)^2 + \left( y_i + \frac{3}{2\sqrt{3}} \right)^2 - R^2 \right|$$

Як відомо, оцінювальна функції  $OF_i$  для кругової інтерполяції в точці з координатами  $x_i, y_i$  має такий вид:

$$OF_i = (x_i^2 + y_i^2) - R^2.$$

Знайдемо значення оцінювальної функції при виконанні  $i+1$ -го кроку інтерполяції для різних переміщень:

$$\begin{aligned} OF_{DL(i+1)} &= \left( x_i - \frac{1}{2} \right)^2 + \left( y_i + \frac{3}{2\sqrt{3}} \right)^2 - R^2 = x_i^2 + y_i^2 - R^2 - x_i + \frac{3}{\sqrt{3}} y_i + 1 = \\ &= OF_i - x_i + \sqrt{3} y_i + 1, \end{aligned}$$

$$\begin{aligned} OF_{DR(i+1)} &= \left( x_i + \frac{1}{2} \right)^2 + \left( y_i + \frac{3}{2\sqrt{3}} \right)^2 - R^2 = x_i^2 + y_i^2 - R^2 + x_i + \sqrt{3} y_i + 1 = \\ &= OF_i + x_i + \sqrt{3} y_i + 1. \end{aligned}$$

Проаналізуємо отримані значення оцінювальної функції:

$$OF_{DL(i+1)} = OF_i - x_i + \sqrt{3} y_i + 1, \quad OF_{DR(i+1)} = OF_i + x_i + \sqrt{3} y_i + 1.$$

Залежно від результату аналізу, вибираємо такий крок інтерполяції, який відповідає найменшому значенню  $OF_i$ . Присвоюємо  $OF_i$  таке значення, яке відповідає вибраному кроку. Переходимо до наступного кроку.

Для формування траєкторії в першому октанті в секторі від  $30^\circ$  до  $90^\circ$  відтворимо ділянку кола, що відповідає дузі  $\theta \in \left[ \frac{\pi}{6}, \frac{\pi}{2} \right]$ . У першому октанті в секторі від  $30^\circ$  до  $90^\circ$  при формуванні крокової траєкторії найменша похибка

має місце при використанні  $I_{DL}$  - діагонального лівого і  $I_{HL}$  - горизонтального лівого кроків (рис. 3.9).

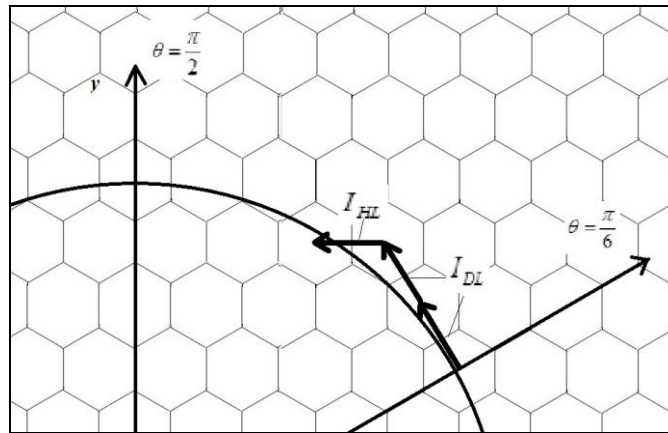


Рисунок 3.9 – Інтерполяція дуги кола в секторі від  $30^\circ$  до  $90^\circ$

Знайдемо значення оцінювальної функції на  $i+1$ -му кроці інтерполяції

для першого октанту дуги кола  $\theta \in \left[ \frac{\pi}{6}, \frac{\pi}{2} \right]$ :

$$\begin{aligned} OF_{DL(i+1)} &= \left( x_i - \frac{1}{2} \right)^2 + \left( y_i + \frac{3}{2\sqrt{3}} \right)^2 - R^2 = \\ &= x_i^2 + y_i^2 - R^2 - x_i + \frac{3}{\sqrt{3}} y_i + 1 = OF_i - x_i + \sqrt{3} y_i + 1. \end{aligned}$$

$$\begin{aligned} OF_{HL(i+1)} &= (x_i - 1)^2 + (y_i^2) - R^2 = x_i^2 - 2x_i + 1 + y_i^2 - R^2 = \\ &= x_i^2 + y_i^2 - R^2 - 2x_i + 1 = OF_i - 2x_i + 1. \end{aligned}$$

Отримаємо такі значення оцінювальної функції:

$$OF_{DL(i+1)} = OF_i - x_i + \sqrt{3} y_i + 1, \quad OF_{HL(i+1)} = OF_i - 2x_i + 1.$$

Розглянемо відтворення крокової траєкторії в другому октанті в секторі від  $90^\circ$  до  $150^\circ$ . Це ділянка кола відповідає дузі  $\theta \in \left[ \frac{\pi}{2}, \frac{5\pi}{6} \right]$ . Якщо траєкторія кола формується в першому октанті в секторі від  $90^\circ$  до  $150^\circ$ , то найменша похибка кругової інтерполяція має місце при використанні  $I_{DL}$  - діагонального

лівого і  $I_{HL}$  - горизонтального лівого кроків (рис. 3.10).

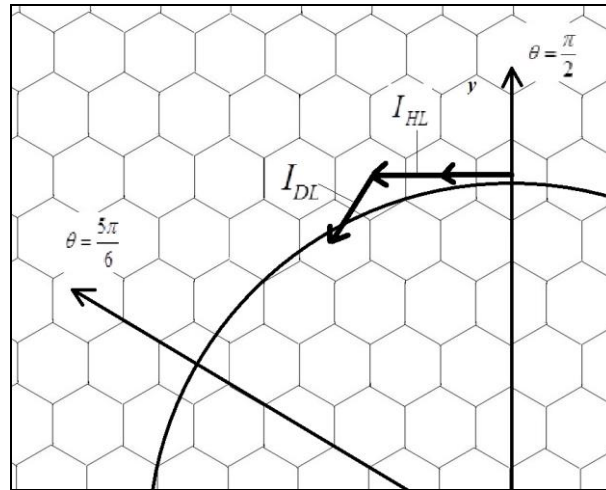


Рисунок 3.10 – Формування крокової траєкторії кола в секторі від  $90^\circ$  до  $150^\circ$

Запишемо вирази для розрахунку квадрату відстані між пікселем і точкою на колі:

$$I_{DL} = \left| \left( x_i - \frac{1}{2} \right)^2 + \left( y_i - \frac{3}{2\sqrt{3}} \right)^2 - R^2 \right|, \quad I_{HL} = \left| (x_i - 1)^2 + (y_i)^2 - R^2 \right|.$$

Формули для визначення  $OF_{(i+1)}$  на  $i+1$ -му кроці кругової інтерполяції для другого октанту дуги кола ( $\theta \in \left[ \frac{\pi}{2}, \frac{5\pi}{6} \right]$ ) мають такий вигляд:

$$\begin{aligned} OF_{DL(i+1)} &= \left( x_i - \frac{1}{2} \right)^2 + \left( y_i - \frac{3}{2\sqrt{3}} \right)^2 - R^2 = \\ &= x_i^2 + y_i^2 - R^2 - x_i - \frac{3}{\sqrt{3}} y_i + 1 = OF_i - x_i - \sqrt{3} y_i + 1. \end{aligned}$$

$$\begin{aligned} OF_{HL(i+1)} &= (x_i - 1)^2 + (y_i^2) - R^2 = \\ &= x_i^2 - 2x_i + 1 + y_i^2 - R^2 = x_i^2 + y_i^2 - R^2 - 2x_i + 1 = OF_i - 2x_i + 1. \end{aligned}$$

Таким чином, отримано такі значення оцінювальної функції:

$$OF_{DL(i+1)} = OF_i - x_i - \sqrt{3} y_i + 1. \quad OF_{HL(i+1)} = OF_i - 2x_i + 1.$$

Для відтворення траєкторії кола в другому та третьому октанті в

секторі від  $150^\circ$  до  $210^\circ$  сформуємо ділянку кола, що відповідає дузі  $\theta \in \left[ \frac{5\pi}{6}, \frac{7\pi}{6} \right]$ .

На ділянці від  $150^\circ$  до  $210^\circ$  найменша похибка кругової інтерполяції має місце при використанні  $I_{DL}$  - діагонального лівого і  $I_{DR}$  - діагонального правого переміщень (рис. 3.11).

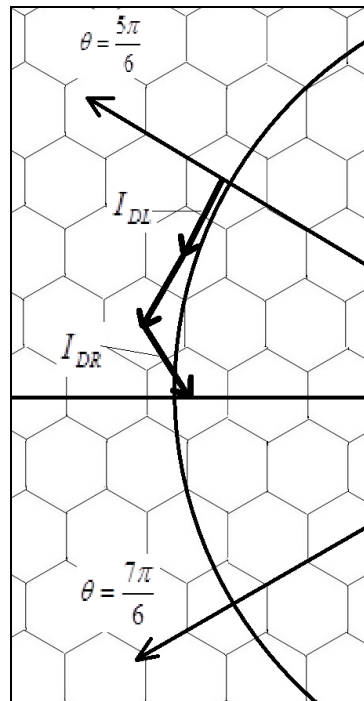


Рисунок 3.11 – Інтерполяція дуги кола в секторі від  $150^\circ$  до  $210^\circ$

Знайдемо значення оцінювальної функції на  $i+1$  - му кроці інтерполяції:

$$\begin{aligned} OF_{DL(i+1)} &= \left( x_i - \frac{1}{2} \right)^2 + \left( y_i - \frac{3}{2\sqrt{3}} \right)^2 - R^2 = \\ &= x_i^2 + y_i^2 - R^2 - x_i - \frac{3}{\sqrt{3}} y_i + 1 = OF_i - x_i - \sqrt{3} y_i + 1. \end{aligned}$$

$$\begin{aligned} OF_{DR(i+1)} &= \left( x_i + \frac{1}{2} \right)^2 + \left( y_i - \frac{3}{2\sqrt{3}} \right)^2 - R^2 = \\ &= x_i^2 + y_i^2 - R^2 + x_i - \sqrt{3} y_i + 1 = OF_i + x_i - \sqrt{3} y_i + 1. \end{aligned}$$

Отримаємо такі значення оцінювальної функції:

$$OF_{DL(i+1)} = OF_i - x_i - \sqrt{3}y_i + 1, \quad OF_{DR(i+1)} = OF_i + x_i - \sqrt{3}y_i + 1.$$

У третьому октанті в секторі від  $210^\circ$  до  $270^\circ$  сформуємо ділянку кола, що відповідає дузі  $\theta \in \left[ \frac{7\pi}{6}, \frac{3\pi}{2} \right]$ . Найменша похибка інтерполяції має місце при використанні  $I_{DR}$  - діагонального правого і  $I_{HR}$  - горизонтального правого кроків (рис. 3.12).

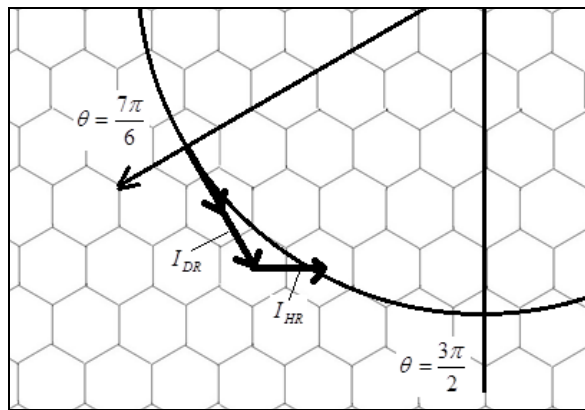


Рисунок 3.12 – Формування крокової траєкторії дуги кола в секторі від  $210^\circ$  до  $270^\circ$

Знайдемо значення оцінювальної функції на  $i+1$  - му кроці інтерполяції:

$$\begin{aligned} OF_{DR(i+1)} &= \left(x_i + \frac{1}{2}\right)^2 + \left(y_i - \frac{3}{2\sqrt{3}}\right)^2 - R^2 = \\ &= x_i^2 + y_i^2 - R^2 + x_i - \sqrt{3}y_i + 1 = OF_i + x_i - \sqrt{3}y_i + 1, \end{aligned}$$

$$\begin{aligned} OF_{HR(i+1)} &= (x_i + 1)^2 + (y_i)^2 - R^2 = x_i^2 + y_i^2 - R^2 + 2x_i + 1 = \\ &= OF_i + 2x_i + 1. \end{aligned}$$

Отримуємо такі значення оцінювальних функцій:

$$OF_{DR(i+1)} = OF_i + x_i - \sqrt{3}y_i + 1, \quad OF_{HR(i+1)} = OF_i + 2x_i + 1.$$

Інтерполяція дуги кола в четвертому октанті відповідає дузі

$\theta \in \left[ \frac{3\pi}{2}, \frac{11\pi}{6} \right]$ . Найкраще наближення буде при формуванні  $I_{HR}$  - горизонтального правого і  $I_{DR}$  - діагонального правого кроків (рис. 3.13).

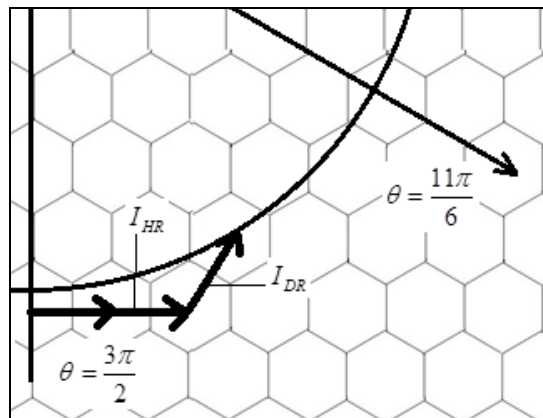


Рисунок 3.13 – Формування крокової траєкторії в секторі від  $270^\circ$  до  $330^\circ$

Знайдемо значення оцінювальних функцій на  $i+1$ -му кроці інтерполяції:

$$\begin{aligned} OF_{HR(i+1)} &= (x_i + 1)^2 + (y_i)^2 - R^2 = x_i^2 + y_i^2 - R^2 + 2x_i + 1 = \\ &= OF_i + 2x_i + 1, \end{aligned}$$

$$\begin{aligned} OF_{DR(i+1)} &= \left(x_i + \frac{1}{2}\right)^2 + \left(y_i + \frac{\sqrt{3}}{2}\right)^2 - R^2 = x_i^2 + y_i^2 - R^2 + x_i + \sqrt{3}y_i + 1 = \\ &= OF_i + x_i + \sqrt{3}y_i + 1. \end{aligned}$$

Отже:  $OF_{DR(i+1)} = OF_i + x_i + \sqrt{3}y_i + 1$ ,  $OF_{HR(i+1)} = OF_i + 2x_i + 1$ .

Розглянемо ділянку кола, що відповідає дузі  $\theta \in \left[ \frac{11\pi}{6}, 2\pi \right]$ . Найменша похибка інтерполяції буде при використанні  $I_{DR}$  - діагонального правого і  $I_{DL}$  - діагонального лівого кроків (рис. 3.14).

Для цієї частини дуги знайдемо значення оцінювальних функцій на  $i+1$ -му кроці інтерполяції:

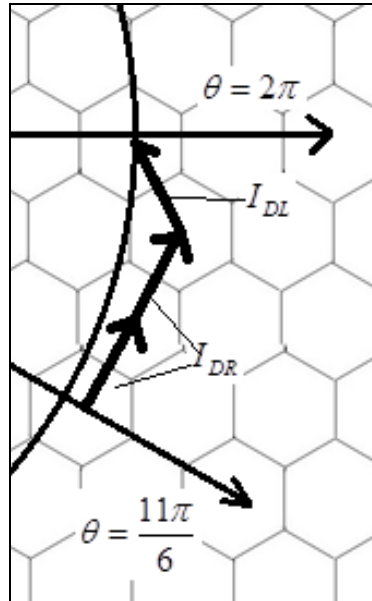


Рисунок 3.14 – Формування крокової траєкторії дуги кола в секторі від  $330^\circ$  до  $360^\circ$

$$\begin{aligned} OF_{DR(i+1)} &= \left(x_i + \frac{1}{2}\right)^2 + \left(y_i + \frac{3}{2\sqrt{3}}\right)^2 - R^2 = x_i^2 + y_i^2 - R^2 + x_i + \sqrt{3}y_i + 1 = \\ &= OF_i + x_i + \sqrt{3}y_i + 1. \end{aligned}$$

$$OF_{DL(i+1)} = \left(x_i - \frac{1}{2}\right)^2 + \left(y_i + \frac{3}{2\sqrt{3}}\right)^2 - R^2 = OF_i - x_i + \sqrt{3}y_i + 1.$$

Отримаємо:

$$OF_{DR(i+1)} = OF_i + x_i + \sqrt{3}y_i + 1, \quad OF_{DL(i+1)} = OF_i - x_i + \sqrt{3}y_i + 1.$$

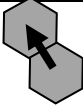
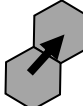
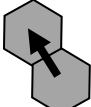



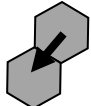
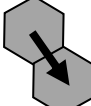
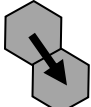

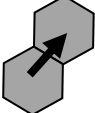

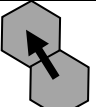
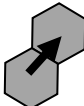
Зведемо отримані типи крокових приростів і формули значень оцінювальної функції до таблиці 3.1.

Отримані формули для розрахунку оцінювальних функцій в якості базової мікрооперації використовують нагромаджувальне додавання. Це спрощує апаратну реалізацію кругової інтерполяції на гексагональному растрі.

Лістинг програми для формування кола на гексаноевому растрі наведено в додатку.



Таблиця 3.1 – Кроки інтерполяції та значення оцінювальної функції

Ділянка інтерполяції	Типи крокових приростів	Формули розрахунку оцінювальних функцій
$\theta \in \left[0, \frac{\pi}{6}\right]$		$OF_{DL(i+1)} = OF_i - x_i + \sqrt{3}y_i + 1$
		$OF_{DR(i+1)} = OF_i + x_i + \sqrt{3}y_i + 1$
$\theta \in \left[\frac{\pi}{6}, \frac{\pi}{2}\right]$		$OF_{DL(i+1)} = OF_i - x_i + \sqrt{3}y_i + 1$
		$OF_{HL(i+1)} = OF_i - 2x_i + 1$
$\theta \in \left[\frac{\pi}{2}, \frac{5\pi}{6}\right]$		$OF_{DL(i+1)} = OF_i - x_i - \sqrt{3}y_i + 1$
		$OF_{HL(i+1)} = OF_i - 2x_i + 1$
$\theta \in \left[\frac{5\pi}{6}, \frac{7\pi}{6}\right]$		$OF_{DL(i+1)} = OF_i - x_i - \sqrt{3}y_i + 1$
		$OF_{DR(i+1)} = OF_i + x_i - \sqrt{3}y_i + 1$
$\theta \in \left[\frac{7\pi}{6}, \frac{3\pi}{2}\right]$		$OF_{DR(i+1)} = OF_i + x_i - \sqrt{3}y_i + 1$
		$OF_{HR(i+1)} = OF_i + 2x_i + 1$
$\theta \in \left[\frac{3\pi}{2}, \frac{11\pi}{6}\right]$		$OF_{DR(i+1)} = OF_i + x_i + \sqrt{3}y_i + 1$
		$OF_{HR(i+1)} = OF_i + 2x_i + 1$
$\theta \in \left[\frac{11\pi}{6}, 2\pi\right]$		$OF_{DL(i+1)} = OF_i - x_i + \sqrt{3}y_i + 1$
		$OF_{DR(i+1)} = OF_i + x_i + \sqrt{3}y_i + 1$

### 3.3 Метод прискореного формування кіл на гексагональному растрі

Використання гексагональної моделі пікселя призводить до збільшення кількості точок на екрані. Це вимагає використання більшої кількості крокових переміщень для кругової інтерполяції, що впливає на продуктивність формування графічного примітива.

Кругова інтерполяція на гексагональному растрі складніша порівняно з використанням прямокутного растру. Це пояснюється різними ординатними розмірами гексагонального пікселя. Слід зазначити, що збільшення роздільної здатності екрану дещо послаблює вимоги до точності інтерполяції.

У зв'язку з цим, актуальними питаннями є підвищення продуктивності колової інтерполяції. Одним з підходів може бути формування крокової траєкторії кола не за допомогою елементарних крокових приростів, а шляхом використання цифрових сегментів, що охоплюють декілька кроків.

Також пропонується використовувати стохастичний розподіл крокових переміщень, який залежить від конкретної ділянки формування траєкторії. Це дозволить підвищити ефективність колової інтерполяції, зменшивши обчислення, необхідні для прогнозування наступного кроку. При використанні гексагонального растру і формуванні подвійних комбінованих приростів можливі їх такі типи (рис. 3.15).

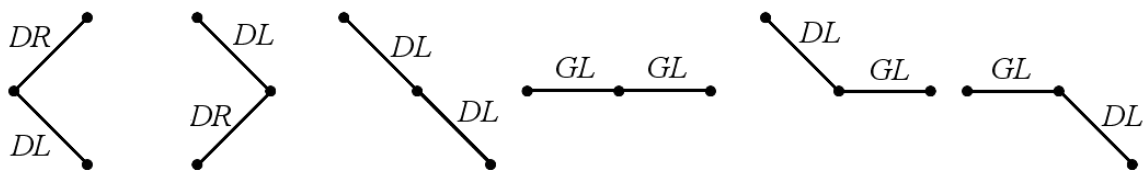


Рисунок 3.15 – Типи можливих кроків

У подальшому будемо використовувати такі позначення елементарних кроків: *DL* – діагональний крок ліворуч, *DR* – діагональний крок праворуч, *GL* – горизонтальний крок ліворуч.

Була розроблено комп'ютерну програму, яка аналізує відсоткове

співвідношення комбінованих кроків у різних секторах формування крокової траєкторії кола.

Дослідження показали, що доцільно розбивати коло на сектори розміром у  $15^\circ$ . У такому випадку для кожного сектора використовуються три типи цифрових сегментів. Наприклад, для сектору від 0 до 15 градусів на гексагональному растрі можуть формуватися такі типи комбінованих кроків: діагональний ліворуч  $DL$  і діагональний праворуч  $DR$ , діагональний праворуч  $DR$  і діагональний ліворуч  $GL$ , діагональний ліворуч  $GL$  і діагональний ліворуч  $DR$ . На рис. 3.16 зображено можливі типи комбінованих приростів для різних секторів формування кола.

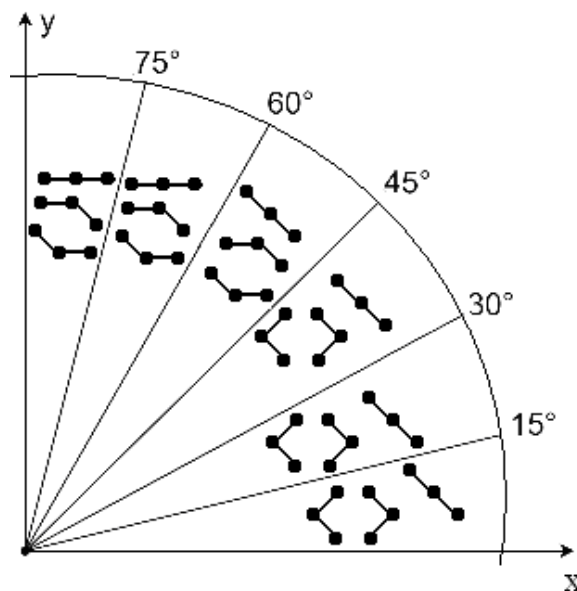








Рисунок 3.16 – Типи подвійних кроків на кожному із секторів кола

У таблиці 3.2 наведено питомий розподіл крокових приростів у кожному секторі. З таблиці видно, що при формуванні крокової траєкторії кола використовується три типи комбінованих кроків.

Три типи крокових переміщень у кожному секторі не дозволяє застосувати двійкову зміну, тому деякі пари кроків можна об'єднати в один тип приросту.

Таблиця 3.2 – Розподіл подвійних кроків

		Сектор кола					
		0°-15°	15°-30°	30°-45°	45°-60°	60°-75°	75°-90°
Тип подвійного кроку		-	-	-	-	22,7%	71,8 %
		-	-	14,60%	36,50%	36,5 %	14,6%
		-	-	13,60%	40,80%	40,8%	13,6%
		22,70%	71,80%	71,80%	22,70%	-	-
		36,50%	14,60%	-	-	-	-
		40,80%	13,60%	-	-	-	-

З рис. 3.17 видно, що формування крокових переміщень (1, 2) і (2, 2) забезпечує перехід до однієї і тієї ж кінцевої точки. Тому їх можна замінити одним типом крокового приросту. При визначенні типу переміщення краще віддавати перевагу тому сегменту, який має вищу імовірність виникнення.

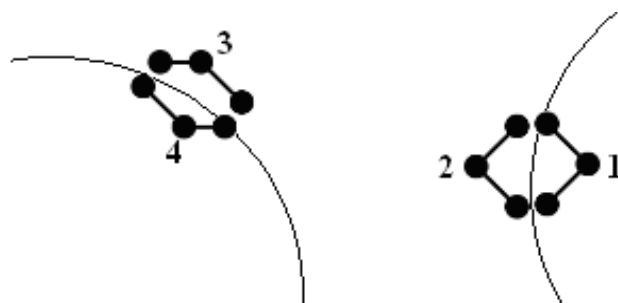


Рисунок 3.17 – Типи однотипних комбінованих приростів

На рис. 3.18 наведено приклад формування дуги комбінованими приростами.

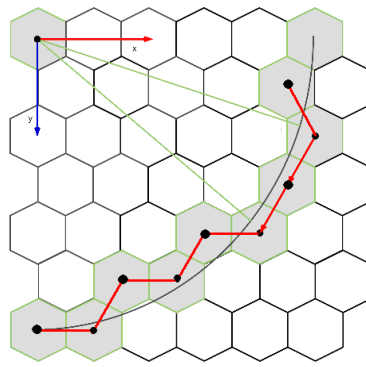


Рисунок 3.18 – Приклад формування дуги кола комбінованими приростами

У таблиці 3.3 наведено значення ймовірності появи подвійних крокових приростів з урахуванням комбінування їх однотипності.

Таблиця 3.3 – Розподіл комбінованих подвійних кроків

		Розподіл подвійних кроків			
		Прогнозований тип кроку	Відсоток	Тип кроку	Відсоток
Сектор кола	0°-15°		77,3%		22,70%
	15°-30°		71,80%		28,2%
	30°-45°		71,80%		28,2%
	45°-60°		77,3%		22,70%
	60°-75°		77,3%		22,70%
	75°-90°		71,80%		28,2%

Отримані результати створюють основу для розробки ряду ефективних методів кругової інтерполяції.

При цьому для прогнозування позиції наступної точки траєкторії вибирається подвійний крок, який має найбільшу ймовірність появи. У випадку, коли прогноз виявився невдалим (про це свідчить знак оцінювальної функції), розраховується скориговане значення  $OF_i$ .

Оцінювальну функцію ( $OF_i$ ) будемо визначати [1-5], [30], [87] за формулою:

$$OF_i = (x_i^2 + y_i^2) - R^2.$$

За умови формування кола в секторі від  $0^\circ$  до  $15^\circ$  проти годинникової стрілки найбільш ймовірним є приріст, який включає діагональне ліворуч і діагональне праворуч переміщення.

Визначимо для нього наступне значення оцінювальної функції:

$$\begin{aligned} OF_{DL,DR} &= (x_i^2 + y_i^2) - R^2 = (x + 0)^2 + \left(y + 2 \cdot \frac{\sqrt{3}}{2}\right)^2 - R^2 = \\ &= x^2 + y^2 - R^2 + 2\sqrt{3} \cdot y + 3 = OF_i + 2\sqrt{3}y + 3. \end{aligned} \quad (3.1)$$

У випадку формування двох діагональних кроків  $DL, DL$  значення  $OF_i$  визначають за виразом:

$$\begin{aligned} OF_{DL,DL} &= x_i^2 + y_i^2 - R^2 = (x - 1)^2 + \left(y + 2 \cdot \frac{\sqrt{3}}{2}\right)^2 - R^2 = \\ &= x^2 + y^2 - R^2 - 2 \cdot x + 1 + 2\sqrt{3}y + 3 = OF_i - 2 \cdot x + 2\sqrt{3}y + 4. \end{aligned} \quad (3.2)$$

Оскільки прогнозування потенційного типу комбінованого приросту відбувається для напрямків  $DL, DR$ , то при необхідності формування послідовних кроків  $DL, DL$  потрібно внести корективи в значення оцінювальної функції. Аналізуючи два останні вирази, можна зробити висновок, що для цього до значення оцінювальної функції  $OF_{DL,DL}$  потрібно додати  $2 \cdot x - 1$ . У такому випадку, при одночасному прогнозуванні наступного комбінованого приросту  $DL, DL$  скориговане значення оцінювальної функції буде рівним:

$$\begin{aligned} OF_K &= OF_i - 2 \cdot x + 2\sqrt{3}y + 4 + (2x - 1) + \\ &+ (-2 \cdot x + 2\sqrt{3}y + 4) = OF_i + 4\sqrt{3}y + 7. \end{aligned}$$

Як правило, значення  $OF_i$  формується в нагромажувальному суматорі.

Тому остання дія передбачає встановлення нагромажувальний суматор в новий стан, який відповідає комбінованому кроці  $DL, DL$ .

Для зменшення обсягу обчислень пропонується провести корекцію оцінювальної функції одночасно з новим прогнозом, що дозволяє не виконувати установку нагромажувального суматора в новий стан. У такому випадку, нове значення  $OF_k$  буде визначено згідно за такою формулою:

$$\begin{aligned} OF_k &= (OF_i + 2\sqrt{3}y + 3) - (2x + 1) + (2\sqrt{3}y + 3) = \\ &= OF_i + 4\sqrt{3}y - 2x + 7. \end{aligned}$$

Розглянемо формування кола в секторі, для якого кут складає  $\Theta \in \left[ \frac{\pi}{12}, \frac{\pi}{6} \right]$ . Із таблиці 3.3 видно, що в цьому випадку найбільш поширеними є комбіновані прирости, які включають два діагональні кроки  $DL, DL$ . Для цього випадку необхідно виконувати прогноз на формування комбінованого приросту  $DL, DL$ . При цьому  $OF_i$  розраховують за виразом (3.2). Для кроків  $DL, DR$  оцінювальну функцію розраховують за формулою (3.1). У випадку, коли прогноз на реалізацію кроку  $DL, DL$  не виправдався, то необхідно скорегувати значення  $OF_i$ . Порівняння виразів (3.1) і (3.2) показало, що вони відрізняються доданком  $2x - 1$ . Тому необхідно для корекції  $OF_i$  і прогнозу наступного приросту знайти такий вираз:

$$\begin{aligned} OF_k &= OF_i - 2 \cdot x + 2\sqrt{3}y + 4 + (2 \cdot x - 1) + (-2 \cdot x + 2\sqrt{3}y + 4) = \\ &= OF_i + 4\sqrt{3}y + 7. \end{aligned}$$

При генерації крокової траєкторії кола в секторі від 30 до 45 градусів комбінований приріст  $DL, DL$  має ймовірність появи 71,8%, тому прогнозування з використанням  $OF$  доцільно проводити саме на такий цифровий сегмент. При цьому  $OF_{DL,DL}$  розраховують за виразом (3.2).

Розрахуємо оцінювальної функції для приросту  $GL, DL$  :

$$\begin{aligned}
 OF_{GL,DL} &= (x_i^2 + y_i^2) - R^2 = \left(x - \frac{3}{2}\right)^2 + \left(y + \frac{\sqrt{3}}{2}\right)^2 - R^2 = \\
 &= x^2 + y^2 - R^2 - 3x + \frac{9}{4} + \sqrt{3}y + \frac{3}{4} = -2x + \sqrt{3}y + 3.
 \end{aligned}$$

Останні два вирази відрізняються один від одного доданком  $-\sqrt{3}y - 1$ .

Розрахуємо скореговане значення  $OF_k$ :

$$\begin{aligned}
 OF_k &= OF_i - 2 \cdot x + 2\sqrt{3}y + 4 + (-\sqrt{3}y - 1) + (-2 \cdot x + 2\sqrt{3}y + 4) = \\
 &= OF_i - 4 \cdot x + 3\sqrt{3}y + 7.
 \end{aligned}$$

У секторі від  $45^\circ$  до  $60^\circ$  найбільш вірогідним є комбінований крок  $DL, GL$ . Таких кроків у секторі - 77,3%. Знайдемо для даного випадку значення оцінювальної функції:

$$\begin{aligned}
 OF_{DL,GL} &= (x_i^2 + y_i^2) - R^2 = \left(x - \frac{3}{2}\right)^2 + \left(y + \frac{\sqrt{3}}{2}\right)^2 = \\
 &= x^2 + y^2 - R - 3x + \frac{9}{4} + \sqrt{3}y + \frac{3}{4} = OF_i - 3x + \sqrt{3}y + 3.
 \end{aligned}$$

Приріст  $DL, DL$  має меншу ймовірність. Для нього оцінювальна функція визначається за виразом (3.2). Формули для  $DL, GL$  і  $DL, DL$  відрізняються на  $x - \sqrt{3}y + 1$ .

З урахуванням прогнозованого розрахунку на крок  $DL, GL$  знаходимо нове значення  $OF_i$ :

$$OF_k = OF_i - 3x + \sqrt{3}y + 3 + (x - \sqrt{3}y + 1) + (-3x + \sqrt{3}y + 3) = -5x + 3\sqrt{3}y + 7$$

Для сектора від  $60^\circ$  до  $75^\circ$  комбіноване переміщення  $DL, GL$  має ймовірність появи 77,30%, тому прогнозне обчислення доцільно виконувати для цього сегменту.

Менш ймовірним є комбіноване переміщення  $GL, GL$ . Запишемо формулу для цього випадку:

$$OF_{GL,GL} = (x_i^2 + y_i^2) - R^2 = (x - 2)^2 + y^2 - R^2 = OF_i - 2x + 4.$$



У випадку невдалого прогнозу та суміщення розрахунку найімовірнішого комбінованого приросту значення оцінювальної функції визначаємо за формулою:

$$\begin{aligned} OF_{GL,GL} &= (x_i^2 + y_i^2) - R^2 = (x-2)^2 + y^2 - R^2 = \\ &= OF_i - 2x + 4 + -3x + \sqrt{3}y + 3 = OF_i - 5x + 2\sqrt{3}y + 7. \end{aligned}$$

Для сектора від  $75^\circ$  до  $90^\circ$  найбільш поширеними є комбіновані кроки  $GL, GL$  (71,8%), а менш поширеними  $DL, GL$  (28,2%). Розрахуємо значення скорегованої оцінювальної функції, за умови невірного прогнозу у напрямку  $GL, GL$  :

$$OF_{GL,GL} = (x_i^2 + y_i^2) - R^2 = (x-2)^2 + y^2 - R^2 = OF_i - 2x + 4.$$

Різниця між значеннями оцінювальних функцій для  $GL, GL$  і  $DL, GL$  складає  $-x + \sqrt{3}y - 1$ . Тому скорегована оцінювальна функція буде мати такий вигляд:

$$OF_k = OF_i - 5x + \sqrt{3}y + 7$$

На рис. 3.19 наведено приклад формування дуги кола на гексагональному растрі.

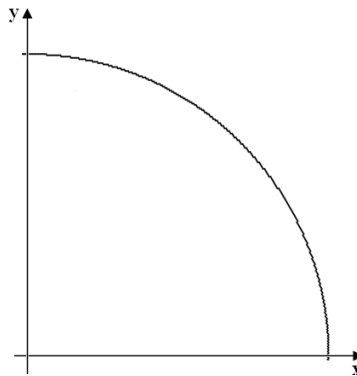


Рисунок 3.19 – Формування дуги кола на гексагональному растрі

При формуванні кола, як на прямокутному, так і гексагональному растрах необхідно проводити аналіз переходів меж секторів. У розробленому методі їх більше, що обумовлює менший, ніж у два рази виграш у продуктивності колової інтерполяції. Формування крокової

траєкторії кола на гексагональному растрі подвійними приростами дає можливість підвищити продуктивність колової інтерполяції в середньому в 1,7 разів.

### 3.4 Особливості формування еліпсів на гексагональному растрі

Розглянемо питання формування крокової траєкторії еліпса на гексагональному растрі з центром  $O(x_0, y_0)$ , що збігається з центром деякої комірки гексагональної сітки та півосями  $a, b$ , паралельними координатним осям декартової системи координат. Розглянемо спочатку дугу еліпса з початковою точкою  $(x_0 + a, y_0)$ , що відповідає  $\theta \in [\pi/2, 2\pi/3]$ , де  $\theta$  – кут, утворений вектором дотичної  $\vec{t}$  до еліпса з додатним напрямком осі  $x$  декартової системи координат (рис. 3.20). У подальшому напрямком будь-якого вектора визначає кут  $\theta$  – кут між цим вектором і напрямком осі  $x$ .

Центром гексагональної клітини будемо називати вузол, при цьому відстань між сусідніми вузлами є одиницею. Найкоротшу відстань від еліпса до комірки визначаємо як мінімальну відстань від вузла до лінії, виходячи з припущення, що лінія не має ширини. Для генерації траєкторії еліпса згідно методу оцінювальної функції, вибираємо тільки ті пікселі, які найближчі до лінії. Якщо лінія неперервна, то комірки формуються неперервно, для кожної сформованої комірки обов'язково формується лише одна з шести сусідніх.

Доведено [106], що коли дотична  $\vec{t}$  до лінії утворює кути  $\theta \in [\pi/2, 2\pi/3]$ , то при обході проти годинникової стрілки вибирають комірки гексагональної сітки, що відображають задану дугу. Рух виконують від початкової точки ( $\theta = \pi/2$ ) у напрямку проти годинникової стрілки, переміщуючись від комірки до наступної комірки з використанням лише переходів у напрямках  $\vec{S}_1$  (що відповідає  $\theta = 2\pi/3$ ) і  $\vec{S}_2$  (що відповідає  $\theta = \pi/3$ ) як на рис. 3.21, причому перехід у напрямку  $\vec{S}_2$  не може відбуватися двічі підряд.

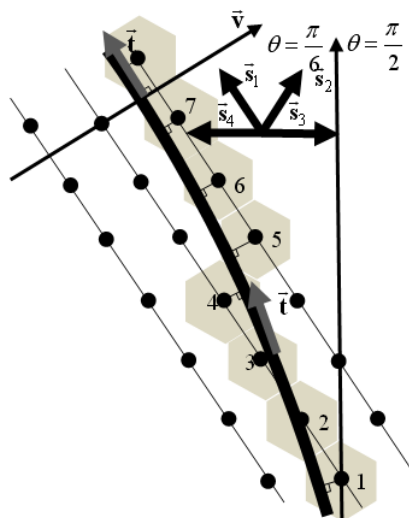


Рисунок 3.20 – Оптимальні напрямки формування комірок сітки, що відповідають дузі еліпса

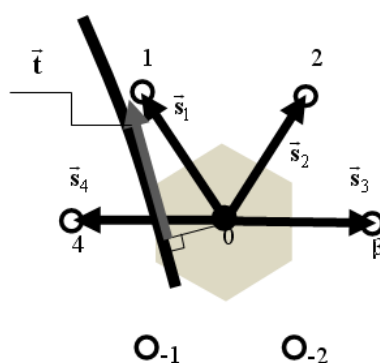


Рисунок 3.21 – Можливі напрямки формування комірок

Перемістимо центр декартової системи координат у точку  $O(x_0, y_0)$ . Параметричне рівняння еліпса, центр якого знаходиться у початку координат, а півосі  $a$  і  $b$ , паралельні відповідно осі абсцис та осі ординат, може бути виражене таким чином:  $x(\varphi) = a \cos(\varphi)$ ,  $y(\varphi) = b \sin(\varphi)$  де  $\varphi$  - це кут між радіус-вектором точки на еліпсі та додатнім напрямком осі абсцис.

У подальшому будемо враховувати, що вектори  $\vec{v}$  і  $\vec{t}$  для точки еліпса, не ортогональні. Визначимо кут  $\varphi(\theta) = \varphi_\theta$  точки еліпса, у якій дотична  $\vec{t}$

нахилена під кутом  $\theta$  з умови:  $\frac{dy}{dx}\varphi(\theta) = \operatorname{tg} \theta$ . Звідки отримуємо

$$-\frac{b}{a}\operatorname{ctg}\varphi_\theta = \operatorname{tg} \theta \text{ і для кутів } \varphi_\theta \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right), \theta \in (0, \pi) \text{ маємо } \varphi_\theta = \operatorname{arctg}\left(-\frac{b}{a}\operatorname{ctg}\theta\right).$$

Враховуючи отриману формулу можна записати, що  $\varphi\left(\frac{2\pi}{3}\right) = \operatorname{arctg}\left(\frac{b}{a\sqrt{3}}\right)$  і, очевидно,  $\varphi\left(\frac{\pi}{2}\right) = 0$ . Таким чином, у той час, коли радіус-вектор  $\vec{v}$  точки еліпса рухається по дузі  $\varphi \in \left[0, \operatorname{arctg}\left(\frac{b}{a\sqrt{3}}\right)\right]$  (обхід проти годинникової стрілки), дотична  $\vec{t}$  до лінії утворює кути  $\theta \in \left[\pi/2, 2\pi/3\right]$ .

Згідно з дослідженням [106], для визначення послідовності комірок, які утворюють заданий сегмент еліпса, можна рухатися від вихідної точки ( $\theta = \pi/2$  еквівалентно  $\varphi = 0$ ) у напрямку, проти годинникової стрілки шляхом послідовного переходу від однієї комірки до іншої, використовуючи тільки переходи в напрямках  $\vec{s}_1$  і  $\vec{s}_2$ , причому перехід у напрямку  $\vec{s}_2$  не може виконуватися двічі підряд.

Враховуючи симетрію еліпса відносно осі абсцис  $\varphi\left(\frac{\pi}{3}\right) = -\operatorname{arctg}\left(\frac{b}{a\sqrt{3}}\right)$  можемо узагальнити правило з комірок на дугу  $\varphi \in \left[-\operatorname{arctg}\left(\frac{b}{a\sqrt{3}}\right), \operatorname{arctg}\left(\frac{b}{a\sqrt{3}}\right)\right]$ .

Якщо радіус-вектор  $\vec{v}$  точки еліпса рухається по дузі  $\varphi \in \left[-\operatorname{arctg}\left(\frac{b}{a\sqrt{3}}\right), \operatorname{arctg}\left(\frac{b}{a\sqrt{3}}\right)\right]$  (обхід проти годинникової стрілки), то дотична  $\vec{t}$  до лінії утворює кути  $\theta \in \left[\pi/2, 2\pi/3\right]$  і вибрати комірки гексагональної сітки, що відображають задану дугу, можна рухаючись від початкової точки ( $\theta = \pi/3$  еквівалентно  $\varphi = -\operatorname{arctg}\left(\frac{b}{a\sqrt{3}}\right)$ ) у напрямку проти годинникової стрілки з використанням лише переходів у напрямках  $\vec{s}_1$  (що

відповідає  $\theta = 2\pi/3$ ) і  $\vec{s}_2$  (що відповідає  $\theta = \pi/3$ ). Причому, перехід у напрямку  $\vec{s}_1$  не може відбуватися двічі підряд при  $\theta \in [\pi/3, \pi/2]$ , а перехід у напрямку  $\vec{s}_2$  не може відбуватися двічі підряд при  $\theta \in [\pi/2, 2\pi/3]$ .

Розглянемо випадок, коли дотична  $\vec{t}$  до дуги еліпса утворює кути  $\theta \in [2\pi/3, \pi]$  (обхід проти годинникової стрілки). З наведених раніше міркувань і геометрії еліпса випливає, що радіус вектор  $\vec{v}$  точки еліпса, який відповідає цій дузі, утворює кути  $\varphi \in [\arctg(b/a\sqrt{3}), \pi/2]$  і, як було доведено раніше, формування (обхід проти годинникової стрілки) вимагає лише двох переходів у напрямках  $\vec{s}_1$  ( $\theta = 2\pi/3$ ) і  $\vec{s}_4$  ( $\theta = \pi$ ). Причому перехід у напрямку  $\vec{s}_4$  не може відбуватися двічі підряд при  $\theta \in [2\pi/3, 5\pi/6]$ , а перехід у напрямку  $\vec{s}_1$  не може відбуватися двічі підряд при  $\theta \in [5\pi/6, \pi]$ .

Врахуємо симетрію еліпса відносно осей декартової системи координат. З властивостей симетрії проведених досліджень випливає, що у кожному з секторів еліпса (рис. 3.22), у яких вектор дотичної  $\vec{t}$  утворює кути  $\theta \in \left[ \frac{\pi}{3} + \frac{\pi}{3}n, \frac{2\pi}{3} + \frac{\pi}{3}n \right]$ ,  $n = 0..5$ , а це еквівалентно тому, що радіус вектор  $\vec{v}$  точки еліпса розбиває еліпс на сектори критичними точками:

$$\varphi_\theta \in \left\{ -\arctg\left(\frac{b}{a\sqrt{3}}\right), \arctg\left(\frac{b}{a\sqrt{3}}\right), \frac{\pi}{2}, \pi - \arctg\left(\frac{b}{a\sqrt{3}}\right), \frac{3\pi}{2}, 2\pi - \arctg\left(\frac{b}{a\sqrt{3}}\right) \right\},$$

існує лише по два напрямки переходу.

Якщо поділити кожен з сегментів  $\theta \in \left[ \frac{\pi}{3} + \frac{\pi}{3}n, \frac{2\pi}{3} + \frac{\pi}{3}n \right]$  на дві рівні частини, то на кожній з отриманих ділянок неможливо здійснити два послідовні переходи в одному з двох можливих напрямків.

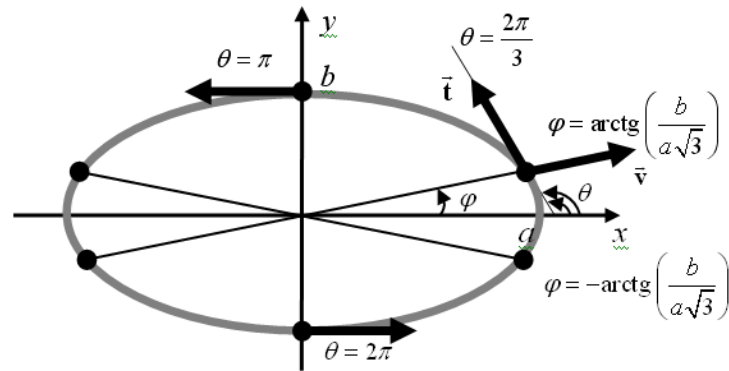


Рисунок 3.22 – Поділ еліпса на сегменти

Для спрощення програмної реалізації алгоритму важливо забезпечити швидке визначення значень трансцендентної функції.  $\text{arctg}(z)$ ,  $z \in \mathbb{R}$ .

Відомо [107], [108], що:

$$\text{arctg}(z) \approx \frac{z}{1 + 0.28125z^2}.$$

Має місце похибка  $5 \cdot 10^{-3}$ , при  $-1 \leq z \leq 1$ ;

Інша модель [2]:

$$\text{arctg}(z) \approx \frac{\pi}{4}x + 0.273x(1 - |x|).$$

Має місце похибка  $4 \cdot 10^{-3}$ , при  $-1 \leq z \leq 1$ .

Для значень  $z$  поза межами проміжку  $-1 \leq z \leq 1$  використовується тотожність -  $\text{arctg}(z) + \text{arctg}(1/z) = \pi/2$ .

Розглянемо особливості формування еліпсів, повернутих на заданий кут. Використаємо результати попереднього випадку, де задачу розв'язано у частковому випадку, коли одна з осей еліпса розташована горизонтально.

Систему гексагональних координат будемо для зручності співставляти з системою декартових координат  $\{O;(x, y)\}$  із центром  $O(x_0, y_0)$ , що співпадає з центром еліпса і є центром деякої комірки гексагональної сітки. Позначимо за  $\theta$  кут, утворений вектором дотичної  $\vec{t}$  до еліпса і додатним

напрямок горизонтальної осі  $x$  декартової системи координат (рис. 3.22).

У роботі [39] доведено, що у випадку, коли дотична  $\vec{t}$  до лінії утворює кути  $\theta \in [\pi/2, 2\pi/3]$  (обхід проти годинникової стрілки), то формувати комірки гексагональної сітки, що відображають відповідну ланку лінії, можна рухаючись від початкової точки  $\theta = \pi/2$  у напрямку проти годинникової стрілки, переходячи від комірки до сусідньої комірки з використанням лише переходів у напрямках  $\vec{s}_1$  (що відповідає  $\theta = 2\pi/3$ ) і  $\vec{s}_2$  (що відповідає  $\theta = \pi/3$ ) (рис. 3.21), причому перехід у напрямку  $\vec{s}_2$  не може відбуватися двічі підряд.

Продовжуючи такі міркування та розбивши всі можливі значення  $\theta \in [0, 2\pi]$  на певні проміжки, можна мінімізувати кількість переходів при формуванні комірок, що відображають лінію на кожному проміжку, з шести до двох. З метою оптимізації будемо використовувати також симетрію еліпса.

Отже, завдання полягає в тому, щоб визначити межі сегментів еліпса, в кожному з яких потрібні переходи тільки в двох напрямках. Параметричне рівняння еліпса з напівосями  $a$ ,  $b$ , паралельними координатним осям, і центром у точці перетину координат, представляється у такому вигляді:

$$X(\varphi) = \begin{bmatrix} x(\varphi) \\ y(\varphi) \end{bmatrix} = \begin{bmatrix} a \cos \varphi \\ b \sin \varphi \end{bmatrix},$$

де  $\varphi$  – кут утворений радіус-вектором  $\vec{v}$  точки еліпса та додатним напрямком осі абсцис. Зазначимо, що вектори  $\vec{v}$  і  $\vec{t}$  для точки еліпса не є ортогональними. Здійснимо поворот еліпса на кут  $\omega$ . Не втрачаючи загальності, нехай  $\omega \in (-\pi/2, \pi/2)$ . Матриця, що здійснює поворот має вид:

$$A(\omega) = \begin{bmatrix} \cos \omega & -\sin \omega \\ \sin \omega & \cos \omega \end{bmatrix}.$$

Тоді декартові координати точки  $X_R(\varphi, \omega) = (x(\varphi, \omega), y(\varphi, \omega))$  еліпса після повороту:

$$X_R(\varphi, \omega) = \begin{bmatrix} x(\varphi, \omega) \\ y(\varphi, \omega) \end{bmatrix} = A(\omega) \cdot X(\varphi). \quad (3.3)$$

Знайдемо кут  $\varphi(\theta, \omega)$  точки еліпса, у якій дотична  $\vec{t}$  нахилена під кутом  $\theta$  з умови:  $\frac{dy(\varphi, \omega)}{dx(\varphi, \omega)} = \operatorname{tg}\theta$ , або  $\frac{dy(\varphi, \omega)/d\varphi}{dx(\varphi, \omega)/d\varphi} = \operatorname{tg}\theta$ . З останнього виразу

$$\text{знаходимо } \operatorname{tg}\varphi(\theta, \omega) = -\frac{b}{a} \operatorname{ctg}(\theta - \omega).$$

Враховуючи область визначення та область значень функцій і допустимі значення параметрів, що входять в останнє рівняння, отримаємо:

$$\varphi\left(\theta = \frac{\pi}{2}, \omega\right) = \operatorname{arctg}\left(-\frac{b}{a} \operatorname{ctg}\left(\frac{\pi}{2} - \omega\right)\right), \quad \text{if } -\frac{\pi}{2} < \omega < \frac{\pi}{2}; \quad (3.4)$$

$$\varphi\left(\theta = \frac{2\pi}{3}, \omega\right) = \begin{cases} \operatorname{arctg}\left(-\frac{b}{a} \operatorname{ctg}\left(\frac{2\pi}{3} - \omega\right)\right) + \pi, & \text{if } -\frac{\pi}{2} < \omega < -\frac{\pi}{3}, \\ \frac{\pi}{2}, & \text{if } \omega = -\frac{\pi}{3}, \\ \operatorname{arctg}\left(-\frac{b}{a} \operatorname{ctg}\left(\frac{2\pi}{3} - \omega\right)\right), & \text{if } -\frac{\pi}{3} < \omega < \frac{\pi}{2}; \end{cases} \quad (3.5)$$

$$\varphi\left(\theta = \frac{\pi}{3}, \omega\right) = \begin{cases} \operatorname{arctg}\left(-\frac{b}{a} \operatorname{ctg}\left(\frac{\pi}{3} - \omega\right)\right), & \text{if } -\frac{\pi}{2} < \omega < \frac{\pi}{3}, \\ -\frac{\pi}{2}, & \text{if } \omega = \frac{\pi}{3}, \\ \operatorname{arctg}\left(-\frac{b}{a} \operatorname{ctg}\left(\frac{\pi}{3} - \omega\right)\right) - \pi, & \text{if } \frac{\pi}{3} < \omega < \frac{\pi}{2}; \end{cases} \quad (3.6)$$

$$\varphi(\theta = \pi, \omega) = \begin{cases} \operatorname{arctg}\left(-\frac{b}{a} \operatorname{ctg}(\pi - \omega)\right) + \pi, & \text{if } -\frac{\pi}{2} < \omega < 0, \\ \frac{\pi}{2}, & \text{if } \omega = 0, \\ \operatorname{arctg}\left(-\frac{b}{a} \operatorname{ctg}(\pi - \omega)\right), & \text{if } 0 < \omega < \frac{\pi}{2}. \end{cases} \quad (3.7)$$

Вирази (3.4) – (3.7) є ключовими для наступних обчислень. Зазначимо, що  $\varphi$  — це кут, що відповідає певній точці на еліпсі без виконання повороту на кут  $\omega$ . Для визначення декартових координат цієї точки після повороту,



слід використовувати формулу (3.3). Кут, під яким спостерігається точка після повороту, становить  $\varphi + \omega$ .

З наведеного вище випливає, що у той час коли радіус-вектор  $\vec{v}$  точки еліпса рухається по дузі  $\varphi \in \left[ \varphi\left(\theta = \frac{\pi}{2}, \omega\right), \varphi\left(\theta = \frac{2\pi}{3}, \omega\right) \right]$  (обхід проти годинникової стрілки), дотична  $\vec{t}$  до лінії утворює кути  $\theta \in [\pi/2, 2\pi/3]$ . Отже, можна формувати пікеселі, рухаючись від початкової точки  $\theta = \pi/2$  у напрямку проти годинникової стрілки, переходячи від комірки до комірки з використанням лише переходів у напрямках  $\vec{s}_1$  і  $\vec{s}_2$ , причому перехід у напрямку  $\vec{s}_2$  не може відбуватися двічі підряд.

Як було показано раніше, якщо дотична  $\vec{t}$  до лінії утворює кути  $\theta \in [\pi/3, 2\pi/3]$ , то формувати комірки гексагональної сітки, що відображають задану дугу, можна рухаючись від початкової точки  $\theta = \pi/3$  у напрямку проти годинникової стрілки з використанням переходів у напрямках  $\vec{s}_1$  (що відповідає  $\theta = 2\pi/3$ ) і  $\vec{s}_2$  (що відповідає  $\theta = \pi/3$ ). Причому перехід у напрямку  $\vec{s}_1$  не може відбуватися двічі підряд при  $\theta \in [\pi/3, \pi/2]$ , а перехід у напрямку  $\vec{s}_2$  не може відбуватися двічі підряд при  $\theta \in [\pi/2, 2\pi/3]$ . Отримані вище формули дають змогу визначити кути та декартові координати точок, що є кінцевими для дуг, що відповідають  $\theta \in [\pi/3, \pi/2]$  і  $\theta \in [\pi/2, 2\pi/3]$ .

Розглянемо дугу еліпса, до якої дотична  $\vec{t}$  утворює кути  $\theta \in [2\pi/3, \pi]$  (обхід проти годинникової стрілки). Для радіус вектора  $\vec{v}$  точки еліпса, що відповідає цій дузі, кути  $\varphi$ , визначаються формулами  $\varphi(\theta = 2\pi/3, \omega)$  і  $\varphi(\theta = \pi, \omega)$ . Як було показано раніше, формування (обхід проти годинникової стрілки) вимагає лише двох переходів у напрямках  $\vec{s}_1$  ( $\theta = 2\pi/3$ ) і  $\vec{s}_4$  ( $\theta = \pi$ ). Причому перехід у напрямку  $\vec{s}_4$  не може відбуватися двічі підряд при  $\theta \in [2\pi/3, 5\pi/6]$ , а перехід у напрямку  $\vec{s}_1$  не може відбуватися двічі підряд при  $\theta \in [5\pi/6, \pi]$ .

Використаємо тепер центральну симетрію повернутого еліпса та гексагональної сітки. З властивостей симетрії еліпса видно, що:  $\varphi(\theta = 4\pi/3, \omega) = \varphi(\theta = \pi/3, \omega) + \pi$ . З попередніх досліджень випливає, що на проміжку  $\theta \in [\pi, 4\pi/3]$  переходи можливі лише у напрямках  $\vec{s}_4$  і  $\vec{s}_5$ .

Для побудови еліпса на ділянці  $\theta \in [4\pi/3, 2\pi + \pi/3]$  достатньо попередньо побудувати еліпс на ділянці  $\theta \in [\pi/3, 4\pi/3]$  і врахувати, що якщо сформована точка, що має координати  $(x(\varphi, \omega), y(\varphi, \omega))$ , то має бути сформована також точка, що має координати  $(-x(\varphi, \omega), -y(\varphi, \omega))$  у системі декартових координат з центром, що збігається з центром еліпса.

Рисунок 3.23 ілюструє специфіку досліджуваної задачі.

Проведені дослідження можна узагальнити на випадок відображення на гексагональній сітці будь якої кривої, кутові коефіцієнти дотичних до якої обчислюються за аналітичними формулами.

Якщо еліпс, що відображається на гексагональній сітці має достатньо великі розміри (у пікселях), то незважаючи на локальне підвищення обчислювальної складності запропонованого алгоритму за рахунок додаткових обчислень, він стає ефективним порівняно з алгоритмом Брезенхема — обчислювальна складність алгоритму асимптотично зменшується утричі.

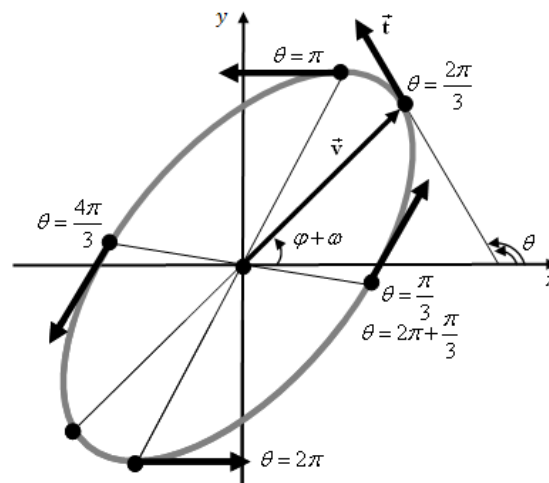


Рисунок 3.23 - Еліпс, повернутий на кут

Отримані співвідношення дають можливість спростити алгоритми еліптичної інтерполяції за рахунок зменшення обчислень на аналіз типів крокових переміщень.

Розглянемо, для прикладу, визначення оцінювальної функції при формуванні еліпса на гексагональному растрі.

Оцінювальна функція для еліпса має загальний вигляд [1], [39], [40]:

$$OF_{i,j} = Ax^2 + Bxy + Cy^2 - 1$$

На гексагональному растрі приріст по осі дорівнює  $\frac{1}{2}$  а по  $y$  -  $\frac{3}{2\sqrt{3}}$ .

Після одиничного кроку по осі  $y$  оцінювальна функція матиме такий вигляд:

$$\begin{aligned} OF_{i,j+1} &= Ax^2 + Bx\left(y + \frac{3}{2\sqrt{3}}\right) + C\left(y + \frac{3}{2\sqrt{3}}\right)^2 - 1 = \\ &= Ax^2 + Bxy + B\frac{3}{2\sqrt{3}}x + Cy^2 + C\frac{3}{\sqrt{3}}y - \frac{1}{4} = OF_{i,j} + B\frac{3}{2\sqrt{3}}x + C\sqrt{3}y - \frac{1}{4}. \end{aligned}$$

Після одиничного кроку по осі  $x$  оцінювальна функція матиме такий вигляд:

$$\begin{aligned} OF_{i+1,j} &= A\left(x + \frac{1}{2}\right)^2 + B\left(x + \frac{1}{2}\right)y + Cy^2 - 1 = \\ &= Ax^2 + Ax + \frac{1}{4} + Bxy + B\frac{1}{2}y + Cy^2 - 1 = OF_{i,j} + Ax + B\frac{1}{2}y - \frac{3}{4}. \end{aligned}$$

Аналогічно можна отримати формули для інших переміщень.

### 3.5 Висновки до розділу 3

1. Розглянуто особливості формування кіл і еліпсів на гексагональному растрі.

2. Встановлено типи крокових переміщень для кіл і еліпсів залежно від ділянки формування крокової траєкторії, що спрощує методи формування еліпсів за рахунок влучення надлишкових обчислень.

3. Модифіковано метод оцінювальної функції для формування кіл на гексагональному растрі.

4. Вперше запропоновано метод кругової інтерполяції на гексагональному растрі, особливість якого полягає у використанні апріорного визначеного стохастичного розподілу крокових приростів залежно від ділянки формування крокової траєкторії, що дало можливість в 1,7 зменшити швидкодію інтерполяції за рахунок прогнозування найбільш вірогідної комбінації кроків.

Результати досліджень цього розділу наведено в публікаціях: [9], [38], [39], [40] [42], [41], [46], [106], [109-113].

## РОЗДІЛ 4

### МЕТОДИ АНТИАЛІАЙЗИНГУ ЗОБРАЖЕНЬ ГРАФІЧНИХ ПРИМІТИВІВ

Основна причина появи ступінчатого ефекту (аліайзингу) – недостатня роздільна здатність пристроїв відображення. Вважається, що ефект аліайзингу для людей із рівнем зору, який вищий середнього, невидимий для форматів вище WHSXGA (26 214 400 пікселів). На жаль, досягти такої роздільної здатності надзвичайно проблематично.

Тому розробка методів усунення ступінчатого ефекту при формування графічних зображень є актуальною.

#### **4.1 Суперсемплінг графічних зображень, які використовують гексагональну модель пікселя**

Один з найбільш розповсюджених методів усунення аліайзингу - це метод, описаний у джерелах [3], [22], [26], [47], [98], [100]. Він оснований на підвищенні частоти дискретизації. Його суть полягає у виконанні обчислень зображення сцени з вищою роздільною здатністю, ніж потрібно для пристрою відображення, і подальшому зменшенні роздільної здатності шляхом усереднення перед виведенням на екран. Метод має просту апаратну реалізацію. Він реалізований у більшості існуючих системах комп'ютерної графіки.

Для методу характерна низька швидкодія, оскільки при збільшенні кількості пікселів в  $n$  разів, кількість розрахунків на один піксель, зростає в  $n^2$  разів. Цей метод вимагає додаткової пам'яті та підвищення пропускну здатності шини пам'яті.

Розглянемо питання реалізації суперсемплінгу при відображенні інформації на гексагональному растрі. На рисунку 4.1 показано тайл, що містить сім пікселів, кожен з яких має інтенсивність кольору  $I_i$ .

При виконанні суперсемплінгу інтенсивність результуючого пікселя

буде дорівнювати  $I = \frac{\sum I_i}{7}$ .

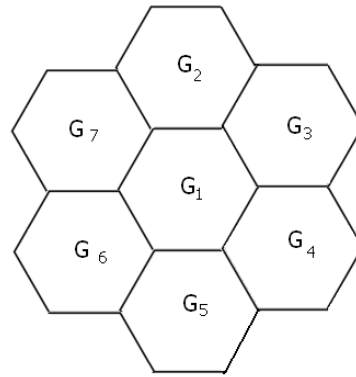


Рисунок 4.1 – Тайл, який включає 7 гексагональних пікселів

Значення інтенсивності результуючого імпульсу можна визначити за формулою:

$$I = \frac{\sum I_i \cdot W_i}{7},$$

де  $W_i$  - ваги пікселів.

Наступний тайл з рефлексійною симетрією має 19 пікселів (рис. 4.2).

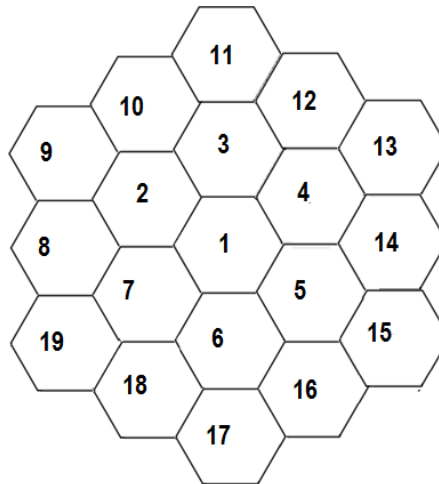


Рисунок 4.2 – Тайл, який включає 19 гексагональних пікселів

Інтенсивність кольору результуючого пікселя після виконання суперсемплінгу буде дорівнювати:

$$I = \frac{\sum I_i}{19}$$

Для покращення якості згладжування зображень необхідно надавати пріоритет пікселям в тайлі. В цьому контексті, найбільшу вагу отримує центральний піксель, а загальна сума ваг усіх пікселів повинна складати 100%.

На рис. 4.3 продемонстровано розподіл ваг пікселів всередині тайлу. Інтенсивність кінцевого імпульсу можна обчислити використовуючи такий вираз:

$$I = \frac{\sum I_i \cdot W_i}{7}$$

де  $W_i$  - ваги пікселів.

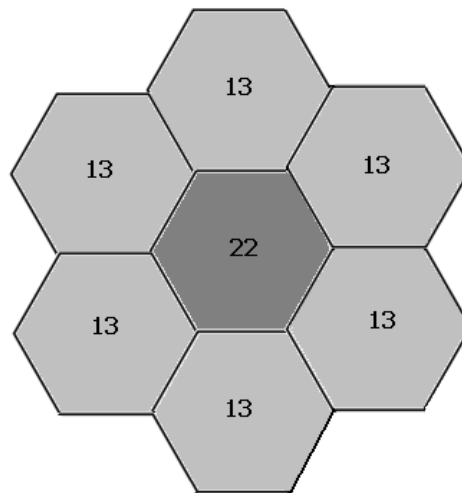


Рисунок 4.3 – Розподіл інтенсивностей кольору серед 7 гексагональних пікселів

На рис. 4.4 наведено розподіл ваг пікселів всередині тайлу, який включає 19 пікселів. При визначенні ваг пікселів доцільно використовувати функцію Гауса.

Якість згладжування визначається кількістю додаткових вибірок та їх розміщенням усередині пікселя. Чим більше кількість точок вибірки, тим вища якість зображення, але це збільшує потребу в додатковій пам'яті та

зменшує продуктивність антиаліазингу.

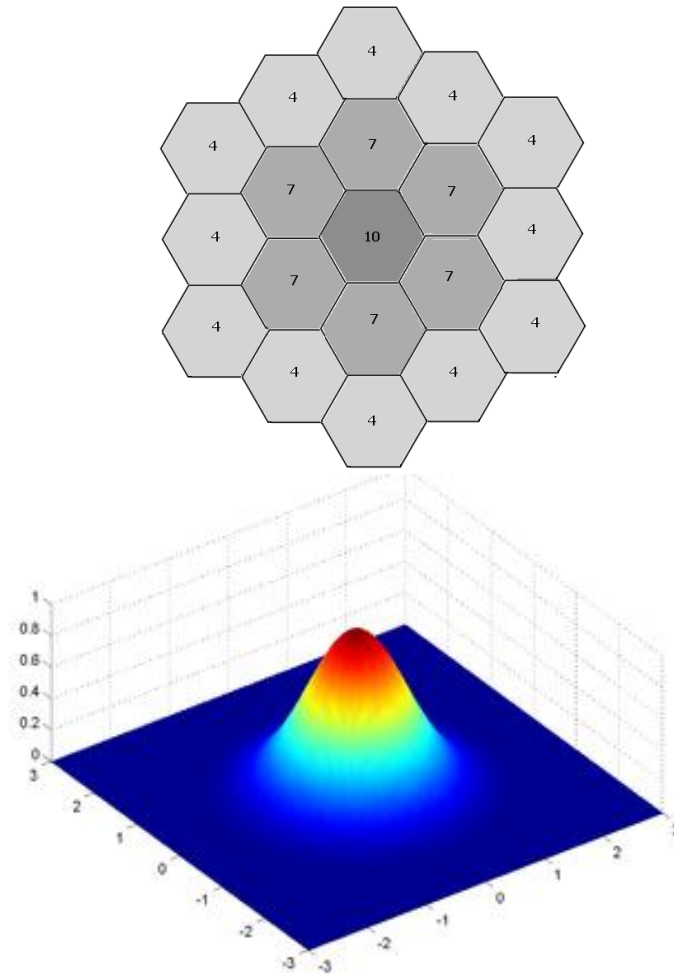


Рисунок 4.4 – Розподіл інтенсивностей кольору серед 19 гексагональних пікселів

М. Діпе і Е. Уолт запропонували [114] стохастично розташувати додаткові точки вибірки. Цей метод ґрунтується на ідеї, що для людського ока випадкове розташування підвбірок сприймається як "шум". Таким чином, один вид артефакту замінюється іншим, до якого людське око є менш чутливим.

Недолік цього методу полягає у високих обчислювальних витратах, які виникають через необхідність розміщення щонайменше 16 випадково розташованих підвбірок на кожен піксель, аби досягти бажаного рівня "неперервного шуму". Якщо кількість підвбірок зменшується, метод



залишається ефективним, але шум стає більш вираженим для людського ока. На рис. 4.4 зображений тайл із 19 пікселями, що робить можливим для реалізація методу М. Діпе і Е. Уолта.

Зменшити рівень аліайзингу можна за рахунок накладання пікселів (А, В) (рис. 4.5) у різних тайлах, або при використанні субпікселів (рис. 4.6).

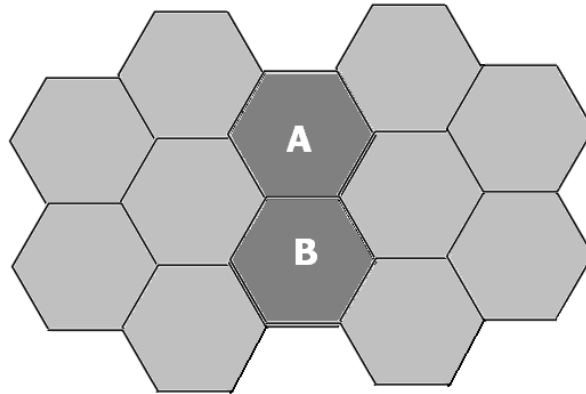


Рисунок 4.5 – Сумісне використання пікселів (А, В) у різних тайлах

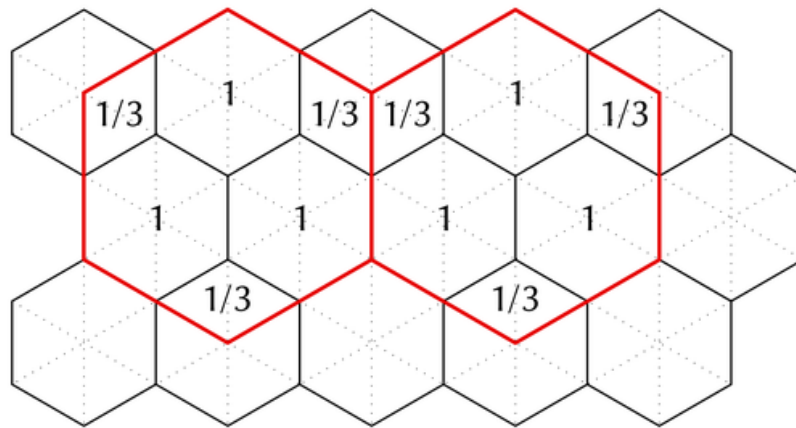


Рисунок 4.6 – Накладання субпікселів у різних тайлах

Розглянуті варіанти виконання суперсрмплінгу на гексагональному растрі можна використати в системах високореалістичної кінцевої візуалізації.

## 4.2 Метод антиаліазингу з визначенням оцінювальних функцій для субпікселів

Згідно з методами крайового антиаліазингу [1-3], [114] для зменшення ступінчастості меж графічних примітивів інтенсивність  $I$  кольору пікселя встановлюють пропорційно до площі  $S$  покриття пікселя об'єктом (рис. 4.7).

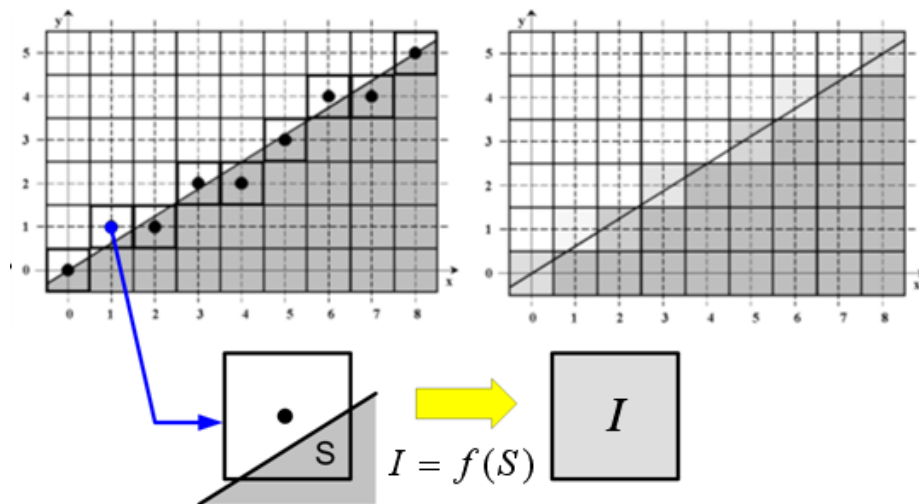


Рисунок 4.7 – Залежність інтенсивності кольору від площі покриття пікселя

Визначення площі  $S$  покриття пікселя вимагає суттєвих обчислювальних витрат, що зменшує швидкість формування зображення. Тому актуальними є питання розробки підходів до зменшення трудомісткості обчислень.

Пропонується новий метод визначення площі покриття пікселя.

Метод полягає у поділі пікселя на субпікселі, в центрах яких розраховується оцінювальна функція. Усі функції обчислюються одночасно та незалежно. Знак оцінювальної функції вказує на розташування субпікселя відносно вектора: при додатному значенні субпіксель розташовується вище відрізка прямої, а при від'ємному – нижче відрізка. Аналіз знаків оцінювальних функцій дозволяє легко визначити, які частини пікселя розміщені по різні сторони від вектора. Обчислюючи загальну кількість субпікселів однакового знаку, можна визначити площу покриття пікселя.

На рис. 4.8 показано два варіанта розміщення субпікселів у межах правильного шестикутника. Варіант б) є прийнятнішим, оскільки в ньому субпікселі краще заміщують площу пікселя, а розміщення субпікселів відповідає концепції побудови гексагонального растру. У варіанті б) в якості базових комірок використовуються гексагональні субпікселі та їх половинки. Як видно з рисунку, піксель включає 31 субпіксель, з них 19 повністю розміщені в межі пікселя.

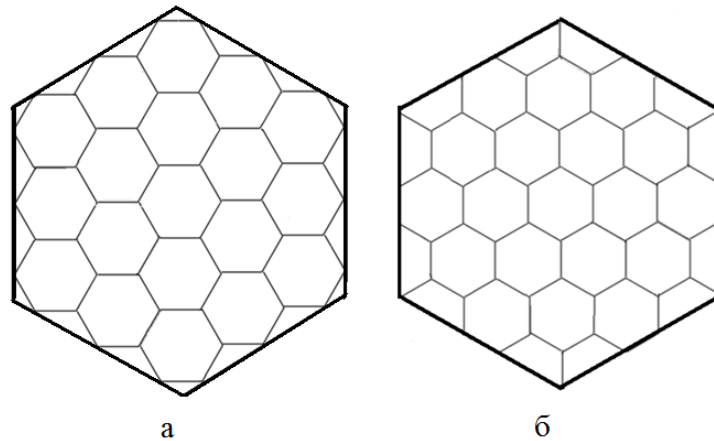


Рисунок 4.8 – Варіанти розміщення субпікселів: а) з частковим замощенням, б) з повним замощенням

На рис. 4.9 наведено приклад перетину відрізком прямої пікселя. Для даного випадку оцінювальні функції  $OF_8, OF_5, OF_6, OF_{19}, OF_{16}, OF_{17}, OF_{30}, OF_{18}, OF_{28}, OF_{29}, OF_{31}, OF_{20}$  мають від'ємний знак. На рис. 4.9 субпікселі, в яких оцінювальна функція від'ємна, заштриховані. Кожен цілісний субпіксель має вагу 1, отже, загальна вага семи таких субпікселів становить 7. Якщо субпіксель є неповним (наприклад, 29), його вага становить  $1/2$ . У заштрихованій ділянці є 5 неповних субпікселів, тому їх загальна вага дорівнює 2,5. Таким чином, сумарна вага повних і неповних субпікселів складає 9,5.

Якщо піксель підлягає зафарбовуванню інтенсивністю кольору  $I$ , то для антиаліайзингу необхідно використати інтенсивність кольору  $I \cdot \frac{9,5}{25}$ .

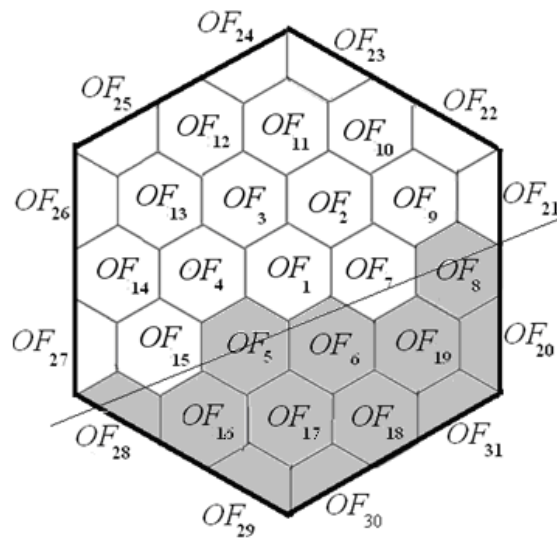


Рисунок 4.9 – Приклад визначення площі покриття пікселя

Розглянемо для прикладу визначення значення  $OF_i$  для деяких субпікселів. Номери субпікселів відповідають індексам оцінювальних функцій.

Для досягнення субпікселя 8 (рис. 4.9) треба сформулювати два горизонтальних крокових переміщення. Тому:

$$OF_8 = y_i \Delta x - \Delta y (x_i + 2) = y_i \Delta x - x_i \Delta y - 2 \Delta y = OF_i - 2 \Delta y.$$

Для досягнення субпікселя 9 (рис. 4.9) необхідно виконати горизонтальне та наступне за ним діагональне переміщення, тому:

$$OF_9 = (y_i + \frac{\sqrt{3}}{2}) \Delta x - \Delta y (x_i + 1 + \frac{1}{2}) = OF_i + \frac{\sqrt{3}}{2} \Delta x - \frac{3}{2} \Delta y.$$

У таблиці 4.1 представлено формули для обчислення значень оціночних функцій для цілих субпікселів, тоді як у таблиці 4.2 вказано формули для розрахунку значень оціночних функцій для неповних субпікселів.

Для розрахунків значень оцінювальних функцій використовується операнд  $\sqrt{3}$ .

Щоб спростити обчислення, останнє значення представимо сумою степенів двійки.

$$\sqrt{3} \approx 1 + \frac{1}{2} + \frac{1}{4}.$$

При використанні запропонованої формули відносна похибка обчислень складає 1,03 %.

Проаналізуємо максимальну помилку у визначенні площі покриття. Враховуємо найдовший можливий шлях, який включає 5 повних субпікселів (згідно з рис. 4.10). Якщо вектор розміщено безпосередньо близько до центрів субпікселів, то при обчисленні площі покриття може мати місце помилка для кожного субпікселя, рівна половині його площі. Таким чином, максимальна помилка при обчисленні площі не перевищуватиме  $5 \times 0,5 = 2,5$  площі повного субпікселя.

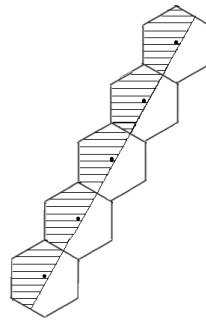


Рисунок 4.10 – Найдовший шлях у гексагональному пікселі

Якщо вектор перетинає приблизно половину пікселя (як показано на рис. 4.11), то похибка у визначенні інтенсивності кольору буде дорівнювати половині рівня інтенсивності цього кольору. У наведеному прикладі це 31 рівень інтенсивності. Така похибка вважається прийнятною.

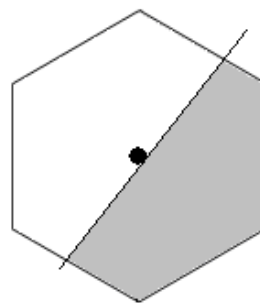


Рисунок 4.11 – Перетин субпікселя

Таблиця 4.1 - Значення оцінювальних функцій для повних субпікселів

№ суб- піксела	Значення оцінювальних функцій
2	$OF_2 = (y_i + \frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i + \frac{1}{2}) = OF_i + \frac{\sqrt{3}}{2}\Delta x - \Delta y \frac{1}{2}.$
7	$OF_7 = y_i\Delta x - \Delta y(x_i + 1) = y_i\Delta x - x_i\Delta y - \Delta y = OF_i - \Delta y.$
8	$OF_8 = y_i\Delta x - \Delta y(x_i + 2) = y_i\Delta x - x_i\Delta y - 2\Delta y = OF_i - 2\Delta y.$
4	$OF_4 = y_i\Delta x - \Delta y(x_i - 1) = y_i\Delta x - x_i\Delta y + \Delta y = OF_i + \Delta y.$
14	$OF_{14} = y_i\Delta x - \Delta y(x_i - 2) = y_i\Delta x - x_i\Delta y + \Delta y = OF_i + 2\Delta y.$
3	$OF_3 = (y_i + \frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i - \frac{1}{2}) = y_i\Delta x - x_i\Delta y + \frac{\sqrt{3}}{2}\Delta x + \frac{1}{2}\Delta y = OF_i + \frac{\sqrt{3}}{2}\Delta x + \frac{1}{2}\Delta y.$
9	$OF_9 = (y_i + \frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i + 1 + \frac{1}{2}) = OF_i + \frac{\sqrt{3}}{2}\Delta x - \frac{3}{2}\Delta y.$
13	$OF_{13} = (y_i + \frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i - 1 - \frac{1}{2}) = OF_i + \frac{\sqrt{3}}{2}\Delta x + \frac{3}{2}\Delta y.$
11	$OF_{11} = (y_i + 2\frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i - 0) = OF_i + \sqrt{3}\Delta x.$
12	$OF_{12} = (y_i + 2\frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i - 1) = OF_i + \sqrt{3}\Delta x + \Delta y.$
10	$OF_{10} = (y_i + 2\frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i + 1) = OF_i + \sqrt{3}\Delta x - \Delta y.$
5	$OF_5 = (y_i - \frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i - \frac{1}{2}) = OF_i - \frac{\sqrt{3}}{2}\Delta x + \Delta y \frac{1}{2}.$
6	$OF_6 = (y_i - \frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i + \frac{1}{2}) = OF_i - \frac{\sqrt{3}}{2}\Delta x - \Delta y \frac{1}{2}.$
15	$OF_{15} = (y_i - \frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i - 1 - \frac{1}{2}) = OF_i - \frac{\sqrt{3}}{2}\Delta x + \Delta y \frac{3}{2}.$
19	$OF_{19} = (y_i - \frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i + 1 + \frac{1}{2}) = OF_i - \frac{\sqrt{3}}{2}\Delta x - \Delta y \frac{3}{2}.$
16	$OF_{16} = (y_i - \frac{3\sqrt{3}}{2})\Delta x - \Delta y(x_i - 1 - \frac{1}{2}) = OF_i - \frac{3\sqrt{3}}{2}\Delta x + \Delta y \frac{3}{2}.$

Таблиця 4.2 - Значення оцінювальних функцій для неповних субпікселів

№ субпіксела	Значення оцінювальних функцій
18	$OF_{18} = (y_i - \frac{2\sqrt{3}}{2})\Delta x - \Delta y(x_i + 1) = OF_i - \sqrt{3} \Delta x - \Delta y.$
17	$OF_{17} = (y_i - \frac{2\sqrt{3}}{2})\Delta x - \Delta y x_i = OF_i - \sqrt{3} \Delta x.$
23	$OF_{23} = (y_i - \frac{4\sqrt{3}}{2})\Delta x - \Delta y(x_i + \frac{1}{2}) = OF_i - 2\sqrt{3} \Delta x - \frac{1}{2} \Delta y.$
24	$OF_{24} = (y_i - \frac{4\sqrt{3}}{2})\Delta x - \Delta y(x_i - \frac{1}{2}) = OF_i - 2\sqrt{3} \Delta x + \frac{1}{2} \Delta y.$
25	$OF_{24} = (y_i - \frac{3\sqrt{3}}{2})\Delta x - \Delta y(x_i - 2) = OF_i - 2\sqrt{3} \Delta x + 2\Delta y.$
26	$OF_{26} = (y_i - \frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i - \frac{3}{2}) = OF_i - \frac{\sqrt{3}}{2} \Delta x + \frac{3}{2} \Delta y.$
27	$OF_{27} = (y_i - \frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i - \frac{3}{2}) = OF_i - \frac{\sqrt{3}}{2} \Delta x + \frac{3}{2} \Delta y.$
28	$OF_{28} = (y_i - \frac{2\sqrt{3}}{2})\Delta x - \Delta y(x_i - 2) = OF_i - \sqrt{3} \Delta x + 2\Delta y.$
29	$OF_{29} = (y_i - \frac{3\sqrt{3}}{2})\Delta x - \Delta y(x_i - \frac{1}{2}) = OF_i - \frac{3\sqrt{3}}{2} \Delta x - \frac{1}{2} \Delta y.$
30	$OF_{29} = (y_i - \frac{3\sqrt{3}}{2})\Delta x - \Delta y(x_i + \frac{1}{2}) = OF_i - \frac{3\sqrt{3}}{2} \Delta x + \frac{1}{2} \Delta y.$
31	$OF_{31} = (y_i - \frac{2\sqrt{3}}{2})\Delta x - \Delta y(x_i + \frac{3}{2}) = OF_i - \sqrt{3} \Delta x - \frac{3}{2} \Delta y.$
20	$OF_{20} = (y_i - \frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i + 2) = OF_i - \frac{\sqrt{3}}{2} \Delta x - 2\Delta y.$
21	$OF_{21} = (y_i + \frac{\sqrt{3}}{2})\Delta x - \Delta y(x_i + \frac{5}{2}) = OF_i + \frac{\sqrt{3}}{2} \Delta x - \frac{5}{2} \Delta y.$
22	$OF_{22} = (y_i + \frac{2\sqrt{3}}{2})\Delta x - \Delta y(x_i + 2) = OF_i + \sqrt{3} \Delta x - 2\Delta y.$

Запропонований метод [27] дозволяє зменшити час реалізації процедури антиаліазингу за рахунок розпаралелення обчислювального процесу. Доцільно реалізувати запропонований метод на основі відеокарт, кожне ядро яких необхідно налаштувати на розрахунок окремого субпікселя.

### 4.3 Метод антиаліазингу зображення з використанням додаткових оцінювальних функцій

Розглянемо можливість застосування гексагональної моделі пікселя для антиаліазингу зображень, яка може слугувати апроксимацією кругової або гаусівської моделей.

На рис. 4.12 зображено форму гексагональної моделі пікселя, яка наближається до круга краще, ніж квадратна модель. Фактично, якщо діаметр пікселя становить 1, тоді його площа за круговою моделлю буде дорівнювати  $\pi/4$ . Одна з характеристик гексагона полягає у тому, що довжина його сторони дорівнює радіусу описаного навколо нього кола.

Якщо задано діаметр описаного навколо гексагону кола, то

$$S = 2 \cdot \sqrt{3} \cdot \frac{D^2}{4} = \frac{2 \cdot \sqrt{3}}{4} \cdot D^2.$$

Співвідношення площі круга до площі гексагона складає 0,906. При використанні квадратної моделі пікселя це співвідношення дорівнює 0,785. Зрозуміло, що для гексагональної моделі пікселя досягається підвищення точності антиаліазингу.

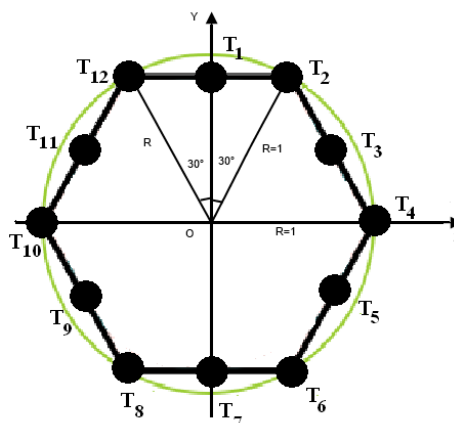


Рисунок 4.12 - Форма гексагональної моделі пікселя



У моделі, представленій на рис. 4.12, ключовою особливістю є те, що відстані від субпікселів до координатних осей є кратними  $1/2^i$  радіусу описаного кола. Це спрощує апаратну реалізацію пристрою для процедури антиаліазингу, оскільки ділення на степінь двійки може бути реалізовано за допомогою мікрооперації зсуву.

Щоб підвищити точність визначення інтенсивності кольору, можливе введення більшої кількості семплів. Для визначення інтенсивності кольору точок траєкторії потрібно визначити площу частини пікселя, яка перетинається з відрізком прямої, як це демонструється на рис. 4.13.

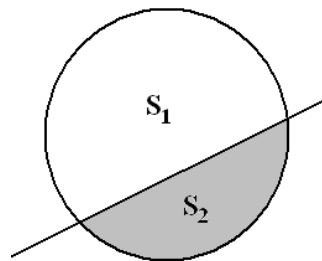


Рисунок 4.13 - Перетин пікселя відрізком прямої

Якщо інтенсивність кольору для відтворення траєкторії вектора дорівнює  $I$ , то інтенсивність кольору у точці, що розглядається, можна знайти за формулою  $I \cdot \frac{S_1}{S_n}$ , де  $S$  - площа частини пікселя, що перетинається з відрізком, де  $S_n$  - площа пікселя.

Для визначення площі сегмента кола, в моделі було використано дванадцять додаткових субпікселів  $T_1 - T_{12}$ . У кожному з цих субпікселів знаходять знак оцінювальної функції. З метою підвищення продуктивності виконання антиаліазингу, обчислення цих функцій виконуватимемо незалежно одне від одного.

Для визначення ординатної відстані точки  $T1$  скористаємося властивістю гексагона, згідно з якою трикутник  $O T_{12} T_2$  є рівностороннім, де кожна сторона дорівнює радіусу описаного кола навколо гексагона. Ясно,

що трикутник  $OT_1T_2$  є прямокутним. Тому виконаємо розрахунки за формулою  $OT_1 = OT_2 \cdot \sin 60^\circ = \frac{\sqrt{3}}{2}$ .

Відстань між кожною парою сусідніх точок є однаковою, а самі точки розміщені таким чином, щоб знаходитися точно на половині відстані між послідовними точками  $T$ . Координати складових субпікселів мають таке значення:

$$\begin{aligned} T_1(x, y + \frac{\sqrt{3}}{2}), & \quad T_2(x + \frac{1}{2}, y + \frac{\sqrt{3}}{2}), & \quad T_3(x + \frac{3}{4}, y + \frac{\sqrt{3}}{4}), & \quad T_4(x + 1, y), \\ T_5(x + \frac{3}{4}, y - \frac{\sqrt{3}}{4}), & \quad T_6(x + \frac{1}{2}, y - \frac{\sqrt{3}}{2}), & \quad T_7(x, y - \frac{\sqrt{3}}{2}), & \quad T_8(x - \frac{1}{2}, y - \frac{\sqrt{3}}{2}), \\ T_9(x - \frac{3}{4}, y - \frac{\sqrt{3}}{4}), & T_{10}(x - 1, y), & T_{11}(x - \frac{3}{4}, y + \frac{\sqrt{3}}{4}), & T_{12}(x - \frac{1}{2}, y + \frac{\sqrt{3}}{2}). \end{aligned}$$

Оцінювальну функцію розраховують виразом:

$$OF_{i,j} = y_i \Delta x - x_i \Delta y.$$

Розрахуємо значення оцінювальних функцій у точках  $T_1 - T_3$  :

$$\begin{aligned} OF_{1i} &= (y + \frac{\sqrt{3}}{2}) \Delta x - x \Delta y = y \Delta x + \frac{\sqrt{3}}{2} \Delta x - x \Delta y = OF_{i,j} + \frac{\sqrt{3}}{2} \Delta x, \\ OF_{2i} &= (y + \frac{\sqrt{3}}{2}) \Delta x - (x + \frac{1}{2}) \Delta y = y \Delta x + \frac{\sqrt{3}}{2} \Delta x - x \Delta y - \frac{1}{2} \Delta y = OF_{i,j} + \frac{\sqrt{3}}{2} \Delta x - \frac{1}{2} \Delta y, \\ OF_{3i} &= (y + \frac{\sqrt{3}}{4}) \Delta x - (x + \frac{3}{4}) \Delta y = y \Delta x + \frac{\sqrt{3}}{4} \Delta x - x \Delta y - \frac{3}{4} \Delta y = OF_{i,j} + \frac{\sqrt{3}}{4} \Delta x - \frac{3}{4} \Delta y, \end{aligned}$$

Так само можна знайти значення оцінювальних функцій  $OF_{4i} - OF_{12i}$  для інших субпікселів. Ці формули залежать від конкретної геометрії та властивостей розглянутої моделі:

$$OF_{4i} = OF_{i,j} - \Delta y, \quad OF_{5i} = OF_{i,j} - \frac{\sqrt{3}}{4} \Delta x - \frac{3}{4} \Delta y, \quad OF_{6i} = OF_{i,j} - \frac{\sqrt{3}}{2} \Delta x - \frac{1}{2} \Delta y,$$

$$OF_{7i} = OF_{i,j} - \frac{\sqrt{3}}{2} \Delta x, \quad OF_{8i} = OF_{i,j} - \frac{\sqrt{3}}{2} \Delta x + \frac{1}{2} \Delta y, \quad OF_{9i} = OF_{i,j} - \frac{\sqrt{3}}{4} \Delta x + \frac{3}{4} \Delta y.$$

Відповідно до визначення, оцінювальна функція набуває додатного значення, якщо піксель розташований вище відрізка прямої, і від'ємного - у всіх інших випадках. Використовуючи знаки оцінювальних функцій  $OF_1 - OF_{12}$  можна ідентифікувати всі можливі сценарії перетину вектора з правильним шестикутником, який репрезентує модель пікселя (табл. 4. 3). Розглянемо конкретний приклад: якщо знаки функцій  $OF_4 - OF_7$  є від'ємними, тоді як знаки  $OF_8 - OF_{12}$ ,  $OF_1 - OF_3$  є додатними, то маємо випадок, зображений на рис. 4.14.

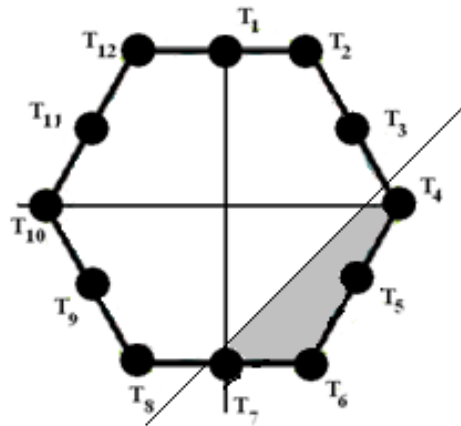


Рисунок 4.14 - Випадок перетину пікселя вектором за умови, що знаки  $OF_4 - OF_7$  від'ємні, а знаки  $OF_8 - OF_{12}$ ,  $OF_1 - OF_3$  додатні

Для цього випадку площа сегмента, що відтинається дорівнює  $S_{min} = 0,75145745$ , а тому інтенсивність кольору буде мати значення  $I \cdot \frac{S_4}{S_i} = I \cdot \frac{0,75145745}{\pi} = 0,23931766 \cdot I$ .

Якщо відрізок прямої проходить між двома субпікселями, то площу сегменту можна знайти більш точно за рахунок віднесення для нього того пікселя, який ближче знаходиться до відрізка прямої.

Доведено [30], що похибка інтерполяції пропорційна значенню модуля оцінювальної функції.

На рис. 4.15 показані абсолютні похибки інтерполяції для двох сусідніх субпікселів. Для зображеного випадку оцінювальна функція має додатне значення для верхнього субпікселя, тоді як для нижнього результат буде від'ємний.

Якщо визначити модуль  $OF_i$  для нижнього субпікселя та співставити його зі значенням функції для верхнього субпікселя, то до сегменту відносять той субпіксель, для якого  $|OF_i|$  буде меншим.

Зрозуміло, що таке уточнення вимагає додаткових обчислювальних затрат.

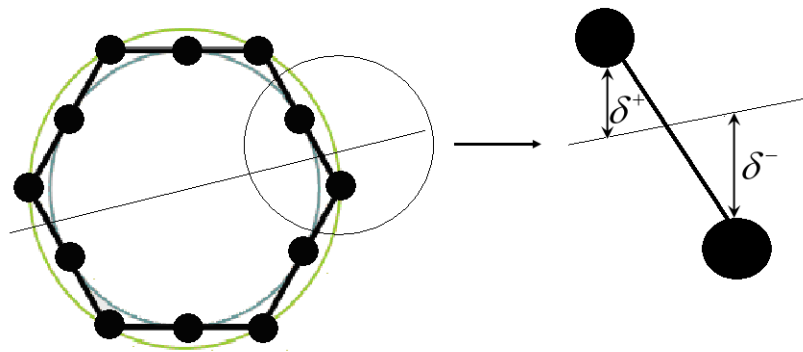
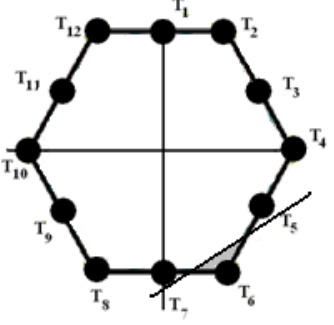
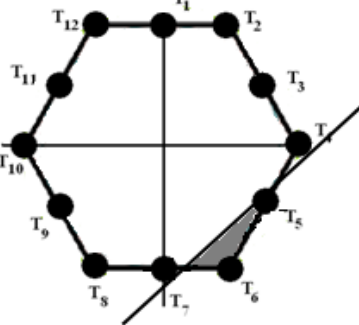
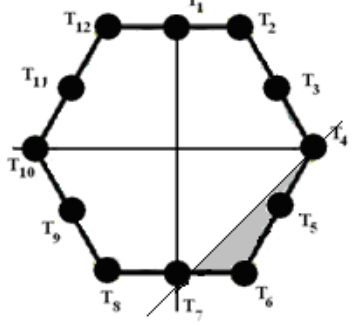
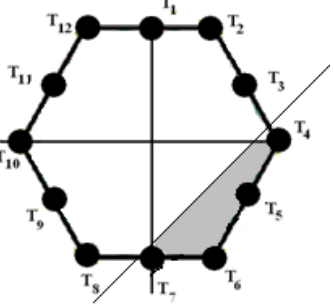
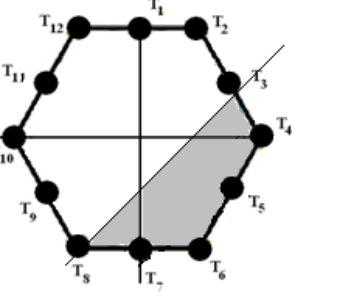
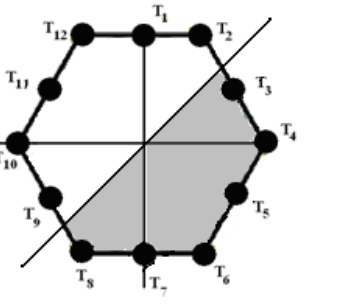
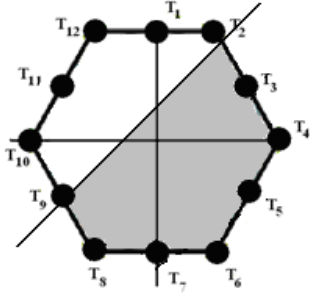
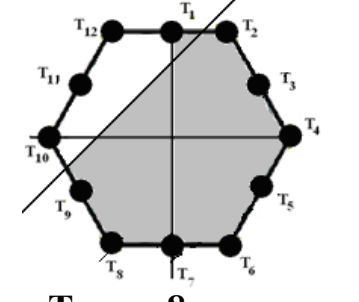
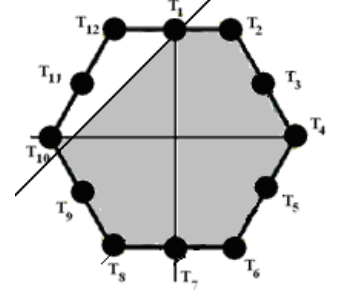
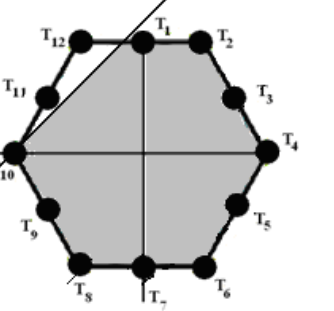
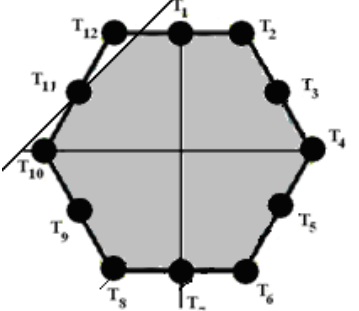
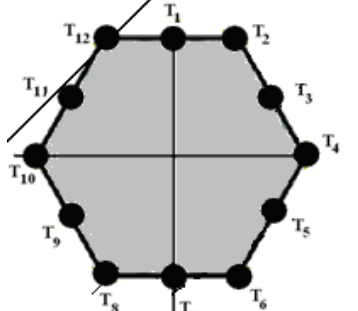


Рисунок 4.15 - Абсолютні похибки інтерполяції для двох сусідніх субпікселів

Таблиця 4.3 - Можливі перетини пікселя відрізком прямих

 <p><b>Точок 1,</b> <math>S_{\min} = 0,0905934</math></p>	 <p><b>Точок 2,</b> <math>S_{\min} = 0,3802725</math></p>	 <p><b>Точок 3,</b> <math>S_{\min} = 0,56586498</math></p>
 <p><b>Точок 4,</b> <math>S_{\min} = 0,75145745</math></p>	 <p><b>Точок 5,</b> <math>S_{\min} = 1,39672605</math></p>	 <p><b>Точок 6,</b> <math>S_{\min} = 1,57</math></p>
 <p><b>Точок 7,</b> <math>S_{\min} = 1,9147276</math></p>	 <p><b>Точок 8,</b> <math>S_{\min} = 2,2396469</math></p>	 <p><b>Точок 9,</b> <math>S_{\min} = 2,5258007</math></p>
 <p><b>Точок 10,</b> <math>S_{\min} = 2,9342628</math></p>	 <p><b>Точок 11,</b> <math>S_{\min} = 2,99897295</math></p>	 <p><b>Точок 12,</b> <math>S_{\min} = 3,0494066</math></p>

#### 4.4 Використання гаусівської моделі пікселя для задач антиаліайзингу

Перспективний напрямок використання гексагонального растру пов'язують з розробкою та виготовленням світлодіодних дисплеїв на основі технології InGaN [78], [79]. У таких дисплеях пікселі мають форму гексагону.

У мікросвітлодіодних дисплеях рис. 4.16 використовують світлодіоди розміром всього 50 нм з повним видимим спектром, доступним в одній системі матеріалів, вирощення індикаторів на одній пластині за один етап процесу.

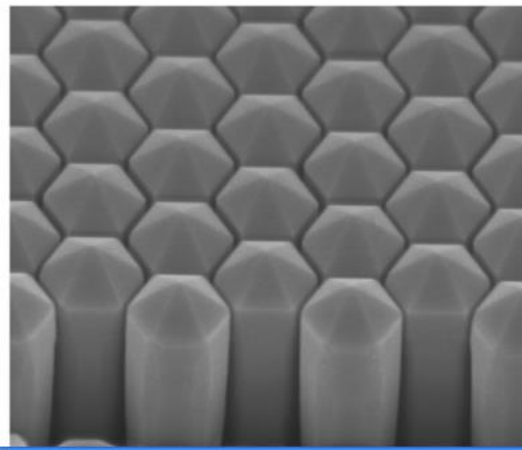


Рисунок 4.16 - Зображення, отримане за допомогою електронного мікроскопа, невеликого масиву наноструктур InGaN [78]

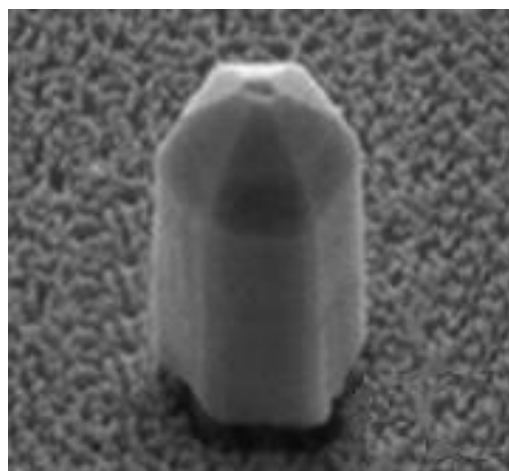


Рисунок 4.17 - Форма мікросвітлодіода [79]

При цьому враховується, що інтенсивність освітлення, яке генерується

пікселем, досягає максимуму в його центрі і знижується до країв, стаючи нульовою на відстані  $R$ . Цей процес узагальнено відображає Гаусівську модель пікселя, згадану в джерелах [33], [34].

Розрахунок інтенсивності кольору для такої моделі полягає у визначенні об'єму фігури, що утворюється внаслідок перетину фільтруючої функції та вертикальної площини, що проходить через примітив.

Через високі обчислювальні витрати, Гаусівська модель пікселя використовується лише у випадках, коли існують строгі вимоги до якості згладжування контурів.

Тому, актуальним є питання спрощення процедури антиаліайзингу з використанням Гаусівської моделі пікселя, що можна досягти за рахунок виведення простих апроксимаційних формул для розрахунків.

Будемо використовувати модель розподілу інтенсивності пікселя, коли щільність інтенсивності розподілена відповідно до нормального закону:

$$p(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right), \quad (4.1)$$

У цій моделі, максимальна інтенсивність зосереджена у центрі пікселя, який прийнято за центр координатної системи. Важливо зазначити, що обрана функція щільності утворює поверхню обертаня і відповідає умові нормування: об'єм, утворений між поверхнею функції  $p(x,y)$  і площиною  $xOy$ , дорівнює одиниці.

Можна змінювати властивості цієї моделі, змінюючи параметр  $p(x,y)$ . Наприклад, встановлюючи значення  $\sigma=0.2$ , знаходимо, що приблизно 99% загального об'єму припадає на прямокутник  $\{(x,y): -0.5 \leq x \leq 0.5, -0.5 \leq y \leq 0.5\}$  (див. рис. 4.18).

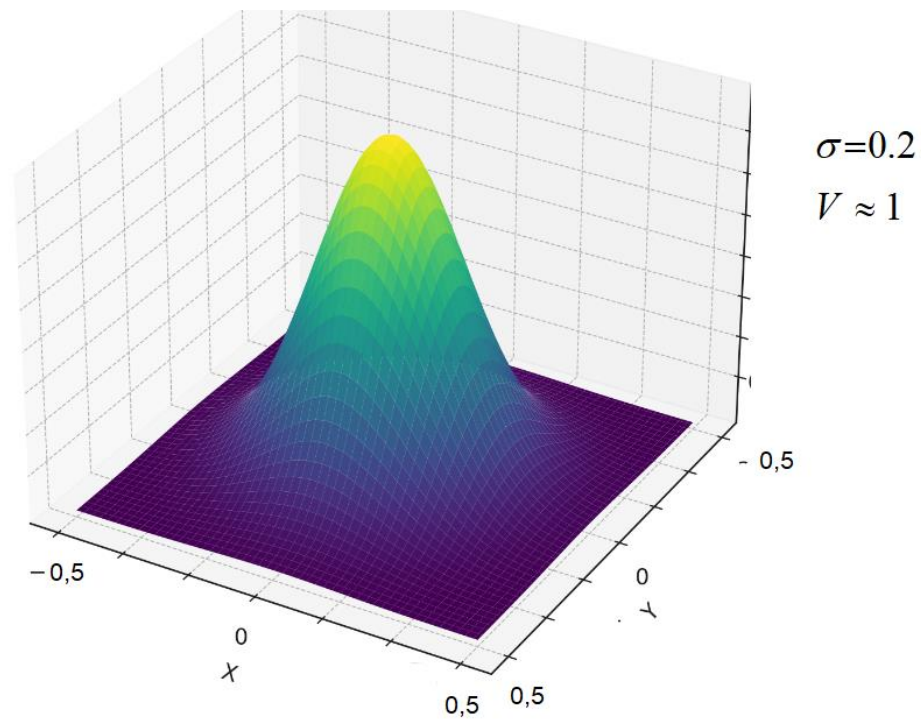


Рисунок 4.18 – Форма пікселя

Розглянемо модифікацію одного з найбільш відомих алгоритмів антиаліазингу відрізків прямих — алгоритму Гупти-Спроула [35]. Нехай відрізок прямої віддалений від центра пікселя на відстань  $d$ .

Якщо  $V(d)$  - об'єм тіла, обмеженого поверхнею (4.1), площиною  $xOy$  та площиною, що знаходиться на відстані  $d$  від осі  $Oz$ . Тоді:

$$I = F(d) = V(d) - V(d). \quad (4.2)$$

Для задач антиаліазингу необхідно обчислювати значення функції  $V(d)$ . Зображення частини поверхні (4.1), що має об'єм  $V(d)$  при  $\sigma=0.2$ ,  $d=0.1$  подано на рис. 1.5.

Для визначення інтенсивності кольору пікселя використовується формула:

$$I_A(P_x, P_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I_{ideal}(x, y) \cdot F(x - P_x, y - P_y) dx dy,$$

де  $I_A$  - інтенсивність кольору пікселя з координатами  $(P_x, P_y)$ ;  $I_{ideal}(x, y)$  - функція, яка визначає інтенсивність кольору в кожній точці простору;  $F(x, y)$



- модель пікселя, тобто функція, яка описує просторове розподілення світла, випромінюваного пікселем.

Враховуємо, що  $I_A(P_x, P_y)$  є поверхнею обертанья. Зорієнтуємо систему координат таким чином, щоб пряма мала рівняння  $x=d$ , тоді:

$$V(d) = \int_{-\infty}^{+\infty} dy \int_{-\infty}^d p(x, y) dx, \quad (4.3)$$

Відокремимо змінні:

$$\begin{aligned} V(d) &= \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{+\infty} e^{-\frac{y^2}{2\sigma^2}} dy \cdot \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^d e^{-\frac{x^2}{2\sigma^2}} dx = \\ &= 1 \cdot \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^d e^{-\frac{x^2}{2\sigma^2}} dx = N(d, 0, \sigma), \end{aligned} \quad (4.4)$$

де  $N(d, 0, \sigma)$  - формула нормального закону розподілу з математичним сподіванням  $a = 0$  і середнім квадратичним відхиленням  $\sigma$ .

Розрахунок  $V(d)$  за формулою (4.4) є доволі громіздким. Апроксимуємо функцію  $N(d, 0, \sigma)$  многочленом  $P_n(d)$  невисокого порядку  $n = 2$  або  $n = 3$ . При цьому для розрахунків використаємо виключно операції додавання та множення. Оскільки функція  $N(d, 0, \sigma)$  є симетричною відносно точки  $(0, 0,5)$  виконаємо апроксимацію за методом найменших квадратів з адаптивними вузлами на інтервалі  $d \in [0; c \cdot \sigma]$ , де  $c$  — константа.

Наприклад, якщо  $\sigma = 0,2$ , то на інтервалі  $d \in [0; 0,5]$  при  $n = 2$  отримаємо такий вираз:

$$V(d) \approx P_2(d) = 0.5 + 2.151d - 2.358d^2. \quad (4.5)$$

Для  $n=3$

$$V(d) \approx P_3(d) = 0.496 + 2.288d - 3.074d^2 + 0.956d^3 \quad (4.6)$$

Графіки функцій  $P_2(d)$ ,  $P_3(d)$ ,  $N(d,0,0.2)$  предсталено на рис. 4.19.

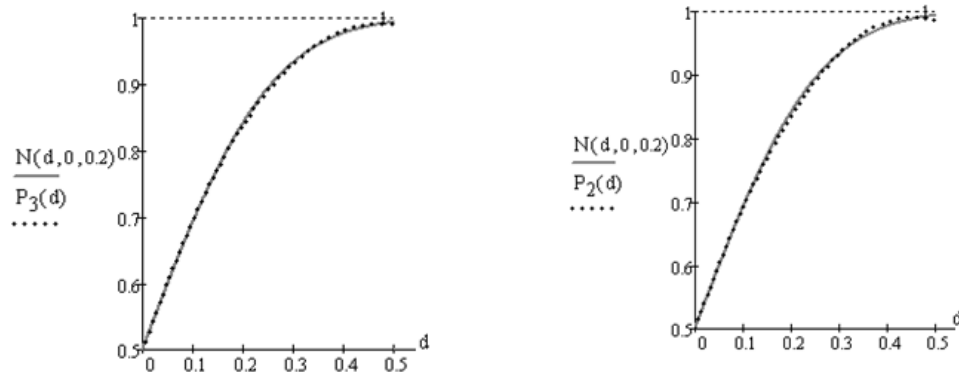


Рисунок 4.19 – Графіки функцій  $P_2(d)$ ,  $P_3(d)$

Позначимо за  $\Delta_2(d) = |N(d,0,0.2) - P_2(d)|$ ,  $\Delta_3(d) = |N(d,0,0.2) - P_3(d)|$  - абсолютні похибки апроксимації на інтервалі  $d \in [0; 0.5]$ . З рис. 4.20 видно, що їх значення не перевищують  $7 \cdot 10^{-3}$  і  $5 \cdot 10^{-3}$  відповідно.

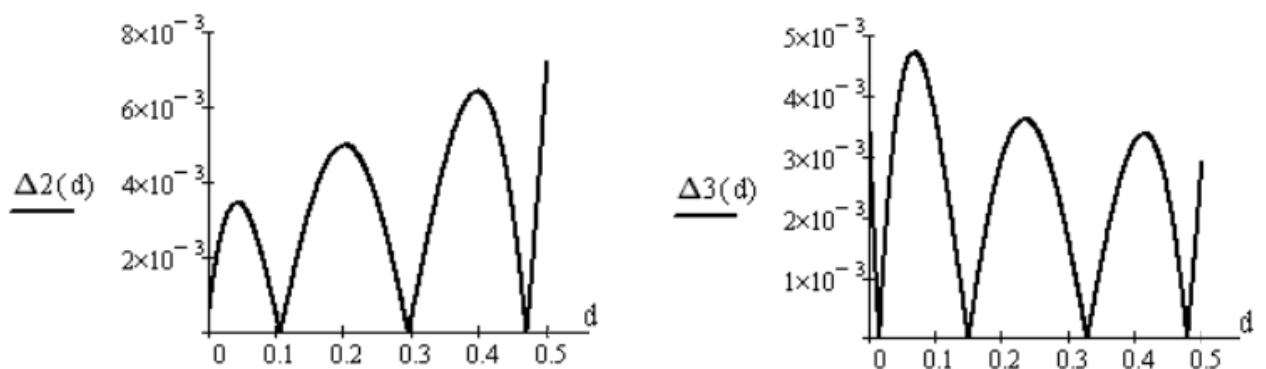


Рисунок 4.20 – Залежності абсолютних похибок  $\Delta_2(d)$  і  $\Delta_3(d)$  від  $d$

Вираз для розрахунку  $P_2(d)$  можна записати так:

$$P_2(d) = 0.5 + (2.151 - 2.358d)d \quad (4.7)$$

На рис. 4.21 відображено структурну схему блока для визначення  $P_2(d)$ .

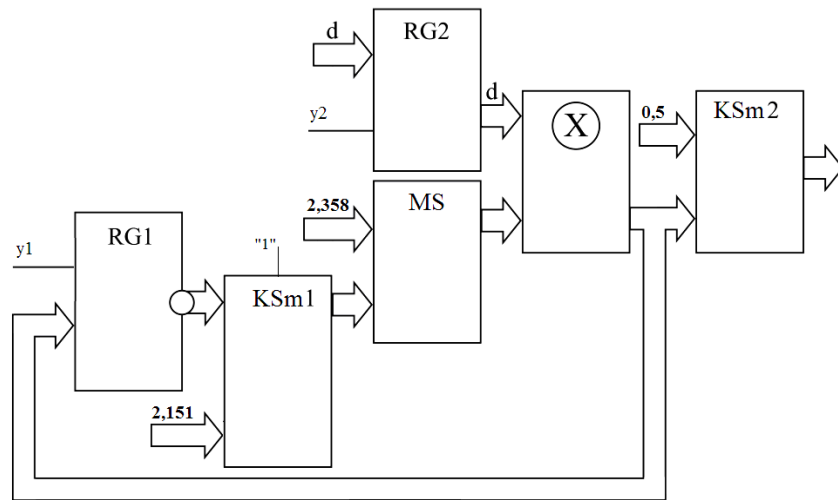


Рисунок 4.21 – Структурна схема блока для визначення  $V(d)$ , використовуючи поліном  $P_2(d)$

Значення відстані  $d$  записується у регістр RG2. За допомогою мультиплексора MS значення 2,358 передається на вхід блоку множення. Добуток  $2.358d$  з виходу блоку множення записується у регістр RG1.

На виході комбінаційного суматора  $KSm1$  отримуємо значення  $(2.151 - 2.358d)$ , яке передається через мультиплексор  $MS$  на вхід блоку множення. У суматорі  $KSm2$  до добутку, отриманого на виході блоку множення, додається значення 0,5.

Розглянемо наближення функції  $V(d) = N(d, 0, \sigma)$ ,  $\sigma = 0.2$ ,  $d \in [0, 0.5]$ , використовуючи кусково-лінійну апроксимацію.

При цьому використаємо обмежену кількість проміжків при діленні відрізка, забезпечуючи при цьому достатньо високу точність апроксимації. Щоб відповідати цим вимогам, проведемо чисельне моделювання, змінюючи кількість сегментів ділення, їх довжину і застосовуючи на кожному з таких сегментів лінійну функцію, яка є частиною найкращого чебишевського наближення. Обчислюючи коефіцієнти апроксимуючих функцій з точністю

до  $10^{-3}$ , отримано такі вирази:

$$V(d) \approx P_1(d) = \begin{cases} 0.506+1.79d, & 0 \leq d < 0.165; \\ 0.636+1.008d, & 0.165 \leq d < 0.305; \\ 0.853+0.295d, & 0.305 \leq d < 0.5. \end{cases} \quad (4.8)$$

Як видно з рис. 4.23, абсолютна похибка  $\Delta_1(d) = |N(d,0,\sigma) - P_1(d)|$  не перевищує 0.01, що становить 1% від максимального рівня інтенсивності. У випадку використання 256 рівнів інтенсивності похибка не перевищуватиме 2.6 рівня. Така похибка є прийнятною для людського ока при нормальних умовах.

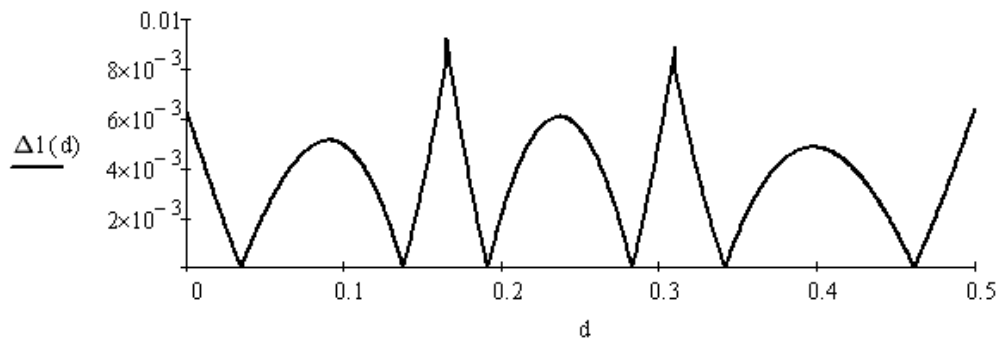


Рисунок 4.22 – Графік абсолютної похибки  $\Delta_1(d)$  від  $d$

Представимо коефіцієнти отриманих функцій як бінарні, зберігаючи розряди до  $2^{-8}$ :

$$V(d) \approx P_b(d) = \begin{cases} 0.1000001_b + 1.11001_b \cdot d, & 0 \leq d < 0.165; \\ 0.101001_b + 1_b \cdot d, & 0.165 \leq d < 0.305; \\ 0.11011011_b + 0.01001_b \cdot d, & 0.305 \leq d < 0.5. \end{cases} \quad (4.9)$$

Як видно з рис. 4.23 абсолютна похибка  $\Delta_2(d) = |N(d,0,\sigma) - P_b(d)|$  менша за 0.01, що відповідає 1% від значення максимальної інтенсивності. Подання  $V(d)$  у формі (4.9) не погіршує наближення  $P_1(d)$ , проте забезпечує вищу продуктивність розрахунків.

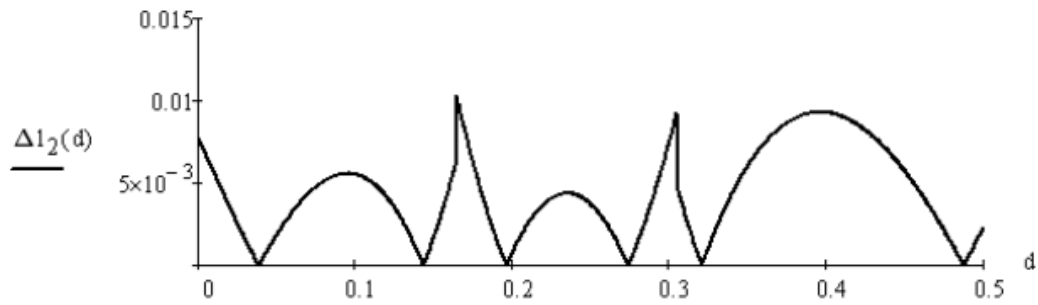


Рисунок 4.23 – Графік абсолютної похибки  $\Delta l_2(d)$

На рис. 4.24 зображено структурну схему блока для розрахунку  $V(d)$  поліномом  $P_1(d)$ .

Запропонована модифікація Гаусівської моделі пікселя, яка не потребує використання заздалегідь розрахованих значень об'єму перетину.

Отримано апроксимаційні вирази для розрахунку об'ємів перетину, які використовують тільки операції множення та додавання, що дає можливість їх реалізувати апаратно. Аналіз показав, що запропоновані структури пристроїв доцільно реалізувати на основі БМК і ПЛІС.

Отримані методи, моделі та структурні схеми можуть бути реалізовані в високопродуктивних системах візуалізації.

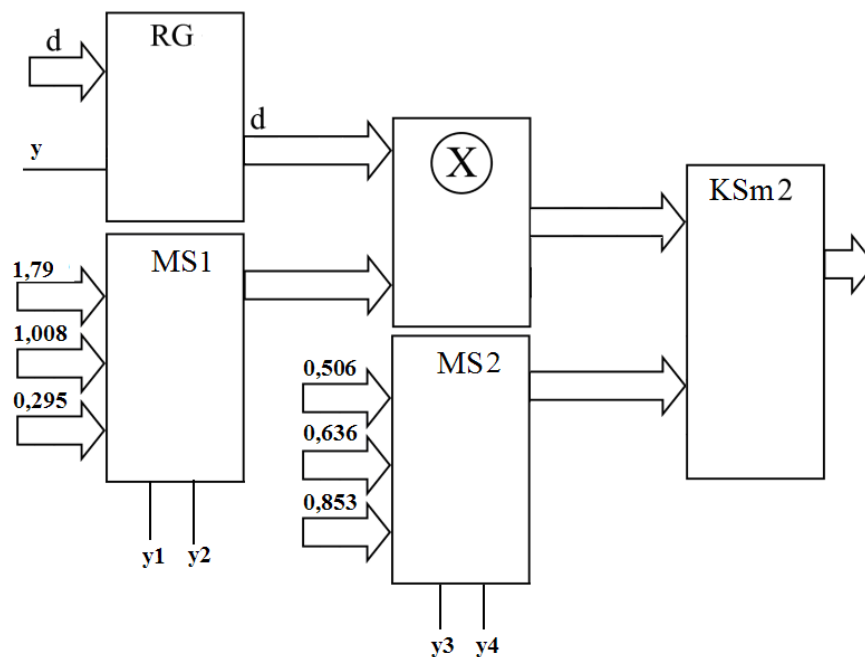


Рисунок 4.24 – Структурна схема блока для розрахунку поліному  $P_1(d)$

#### 4.5 Висновки до розділу 4

1. Запропоновано варіанти розташування семплів при реалізації суперсемплінгу на гексагональному растрі.

2. Розроблено метод згладження крокової траєкторії відрізків прямих, основна ідея якого полягає в розділенні кожного пікселя на субпікселі, для кожного з яких розраховується оцінювальна функція. Це дозволяє визначити, які ділянки пікселя розташовані з різних сторін вектора. При цьому враховуються однакові знаки оцінювальних функцій. Обчислюючи загальну кількість субпікселів із однаковим знаком, можна визначити площу пікселя, яку покрито. Цей метод значно підвищує швидкодію реалізації антиаліайзингу завдяки можливості паралельних обчислень.

3. Подальшого розвитку отримав метод антиаліайзингу векторів на гексагональному растрі, який відрізняється від відомих використанням для обчислення площі покриття пікселя додаткових оцінювальних функцій, що дозволило виконувати антиаліайзинг безпосередньо під час формування зображення крокової траєкторії та усунути етап постобробки, і, як наслідок, підвищити продуктивність формування зображення згладженої траєкторії.

4. Подальшого розвитку отримала гаусівська модель пікселя. Отримано апроксимаційні вирази для розрахунку об'ємів перетину, які мають просту апаратну реалізацію, оскільки включають тільки операції множення та додавання.

5. Розроблено структурні схеми пристроїв для реалізації антиаліайзингу на гексагональному растрі.

Результати досліджень цього розділу наведено в публікаціях: [26], [27], [32-34], [43], [44], [47], [53], [54], [98-100].

## РОЗДІЛ 5

# ПРАКТИЧНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНИХ МЕТОДІВ ФОРМУВАННЯ ЗОБРАЖЕНЬ НА ГЕКСАГОНАЛЬНОМУ РАСТРІ

### 5.1 Графічний редактор на гексагональному растрі

Для формування зображень на гексагональному растрі було розроблено спеціалізований графічний редактор. Для цього було використано технологію Electron, яка дала можливість формувати програмний модуль на мультиплатформі, незалежно від ядерної реалізації та операційної системи. При розробці використано модульний підхід і фреймворк з використанням HTML, CSS і JavaScript. На рис. 5.1 наведено зображення інтерфейсу графічного редактора.

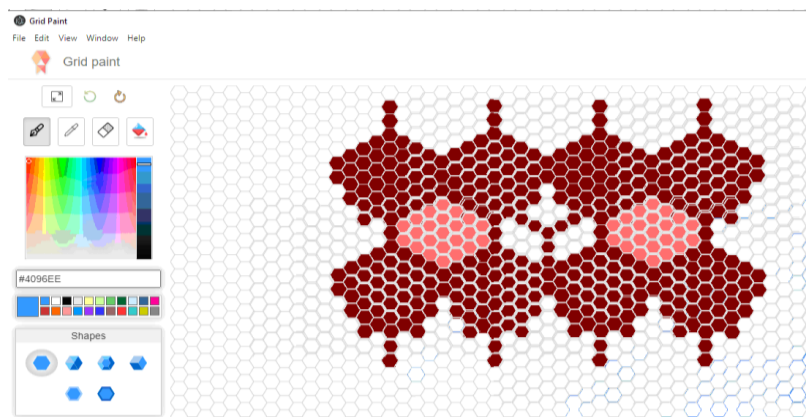


Рисунок 5.1 – Зовнішній вигляд інтерфейсу

Передбачена можливість вибору стандартних зображень пікселів (рис. 5.2) для формування графічних зображень.



Рисунок 5.2 – Типи пікселів

Графічний редактор містить кнопки (рис. 5.3):

- пензлик – для формування зображень пікселів заданого кольору:

- піпетка – для визначення кольору заданої ділянки (пікселя):
- гумка – для витирання вибраних пікселів:
- заповнення (працює тільки тоді, коли здійснюється натиснення на гексагон).

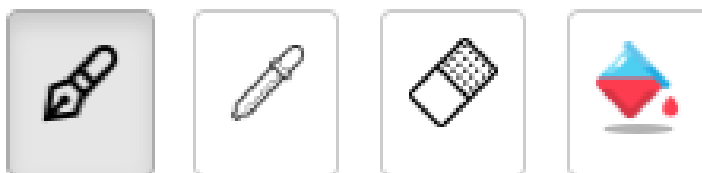


Рисунок 5.3 – Кнопки вибору опцій

У програмному модулі є можливість змінювати параметри гексагонального растру, кольору фону та переднього плану (рис. 5.4).

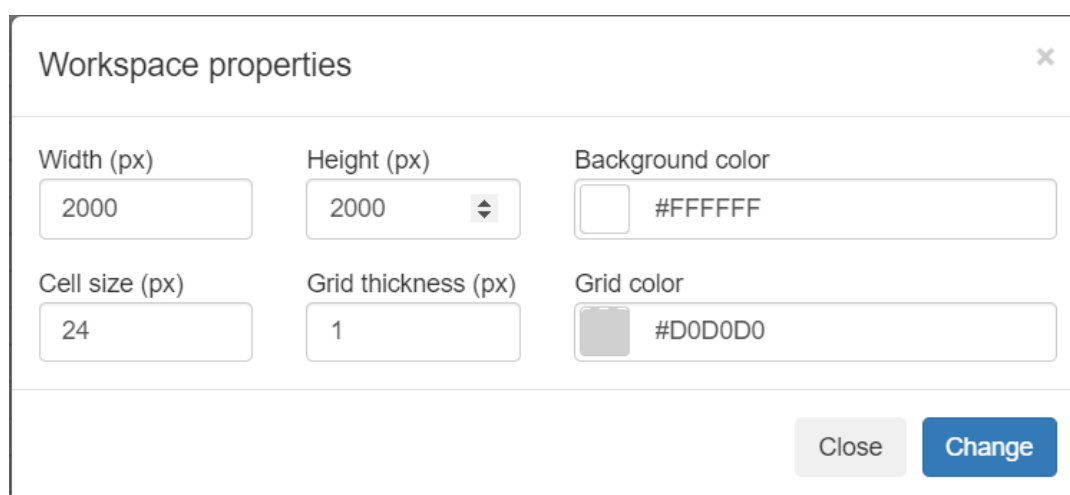


Рисунок 5.4 – Меню керування властивостями робочої зони

Передбачена можливість вибору кольорів пікселів. Для цього розроблено два режими. Перший полягає у виборі кольору шляхом наведення курсора на заданий колір. Другий режим передбачає вибір кольору з палітри кольорів (рис. 5.5). При цьому, для зручності, відображається код кольору.



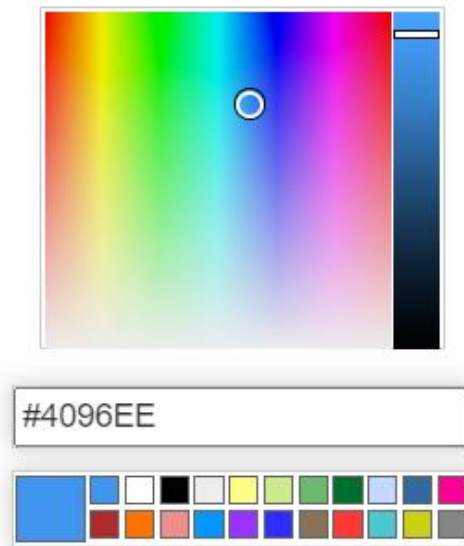


Рисунок 5.5 – Меню обрання кольору

Передбачено три кнопки (рис. 5.6), які відповідають за розмір гексагональної сітки (розмір гексагона), відміну останніх змін. Також є кнопка, яка дозволяє повернутися до останніх змін.

У графічному редакторі є можливість переміщення сформованих зображень (рис. 5.7).

Завдяки цій функціональності можна швидко, без копіювання, змістити зображення у горизонтальній або вертикальній проекції (рис. 5.8).



Рисунок 5.6 – Меню керування гексагональним растром

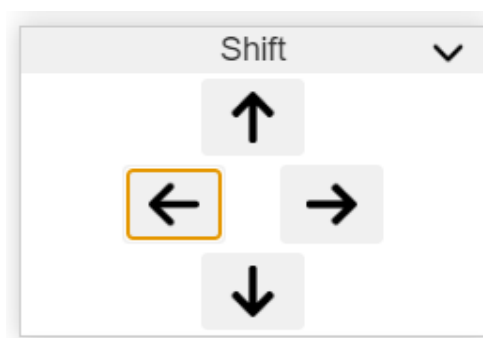


Рисунок 5.7 – Меню зміни положення зображення

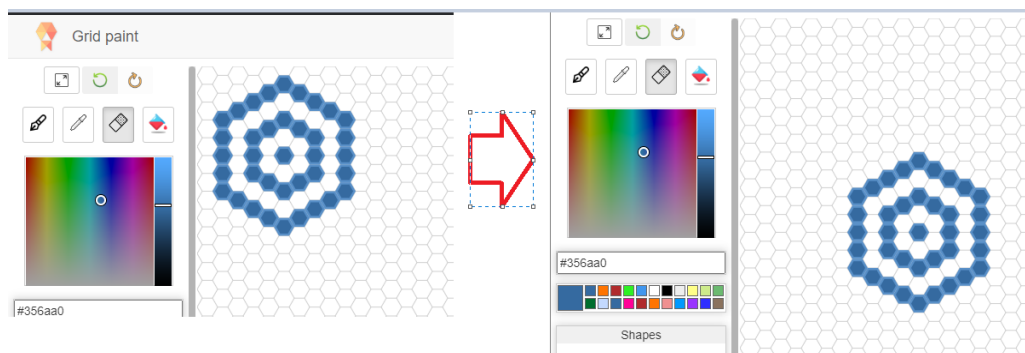


Рисунок 5.8 –Зміна положення зображення

Графічний редактор має функціональні можливості, достатні для формування зображень на гексагональному растрі.

## 5.2 Конвертація зображення для гексагонального растру

Метою розробки програмного застосунку є конвертація зображення в гексагональний растр.

Для внесення вхідних даних у додаток розроблено графічний інтерфейс, побудований із дотриманням UX/UI рекомендацій. Є можливість змінювати такі параметри, як:

- точність – даний функціонал перемикає режими позиціонування слайдера, який відповідає за розмір гексагонального растру;
- коефіцієнт розміру пікселя –встановлюється в межах від 0.2 до 20, де менше значення буде генерувати гексагональний растр більшого розміру, а більше – меншого відповідно.
- вибір зображення – користувач може обрати будь-яке зображення для конвертації його в гексагональний растр. Рекомендовано використовувати формати збереження: PNG, JPEG, BMP, TIFF.

Потенційний клієнт повинен мати екран з роздільною здатністю не менше 1200x440. Програмний застосунок розроблено на мові програмування Java з графічною складовою JavaFX. Основою проекту є засоби створення

додатків Apache Maven.

Інтерфейс програмного застосунку умовно поділений на три робочих ділянки, які відповідають за поставлений функціонал:

- зміна параметрів – зона, де користувач може змінювати вхідні дані, які впливають на кінцевий результат конвертованого зображення;
- робоча ділянка – частина програмного інтерфейсу спроектована для взаємодії користувача та детальним переглядом готового рисунка;

На рисунках 5.9, 5.10 показано ділянки програми для конвертування зображень.

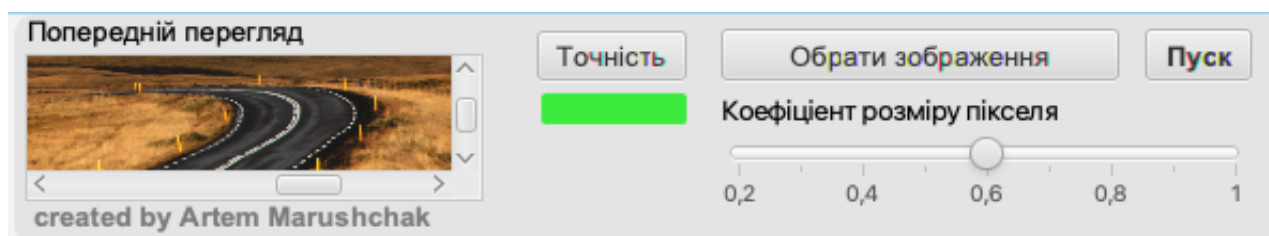


Рисунок 5.9 – Ділянка зміни параметрів



Рисунок 5.10 – Екранна площина

Робота у програмному застосунку розпочинається із вибору потрібного зображення для конвертації. Для початку роботи потрібно натиснути на кнопку «Обрати зображення». Виконається завантаження примітива у вікно попереднього перегляду. Користувач обирає необхідну частину зображення, яка буде конвертована. На рисунку 5.11 показано ділянку попереднього перегляду.

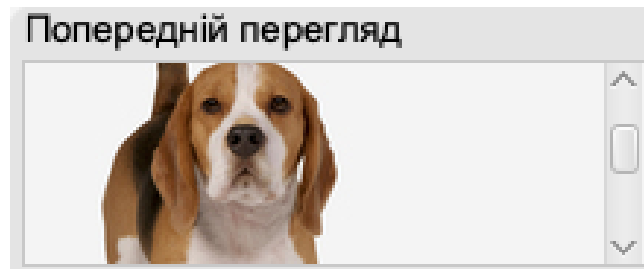


Рисунок 5.11 – Попередній перегляд фрагмента для конвертації зображення

На рис. 5.12 показано стан головного вікна програмного застосунку після виконання конвертації, де ліва частина робочої ділянки буде вхідним зображенням, а права – результатом конвертації.

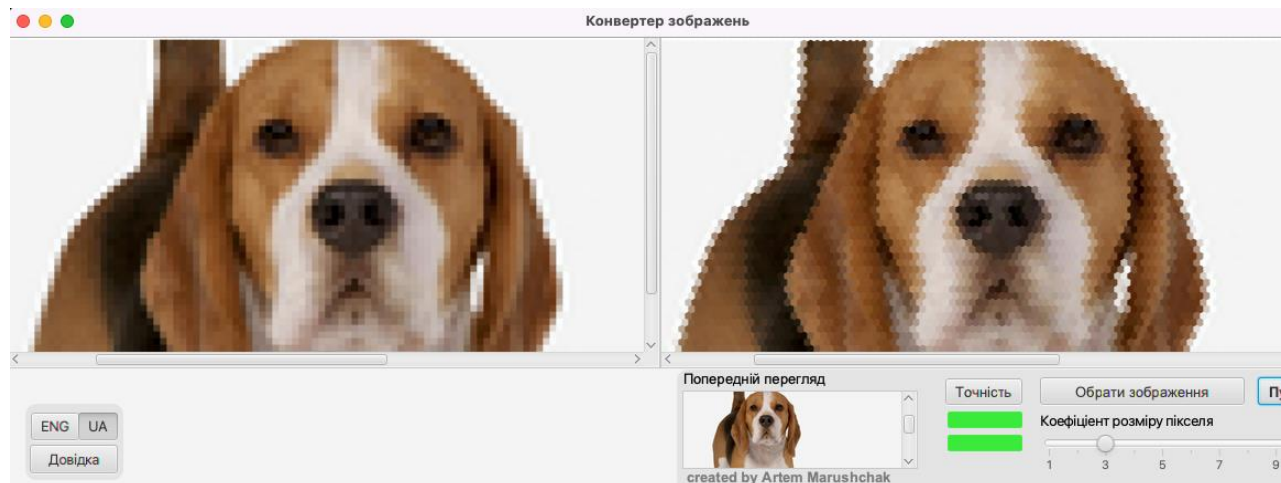


Рисунок 5.12 – Стан головного вікна після процесу конвертації

У програмному застосунку передбачена можливість зміни розміру пікселя. Дана властивість реалізується за допомогою повзунка «Коефіцієнт розміру пікселя». Під час взаємодії із повзунком з'являється гексагональний примітив у правій частині робочої області. Додатковий параметр «Точність» має три режими та змінює розмір пікселя, для того, щоб розширити вибір наявних розмірів піксельного растру. На рисунку 5.13 показано стан програми зі збільшеним розміром піксельної сітки.

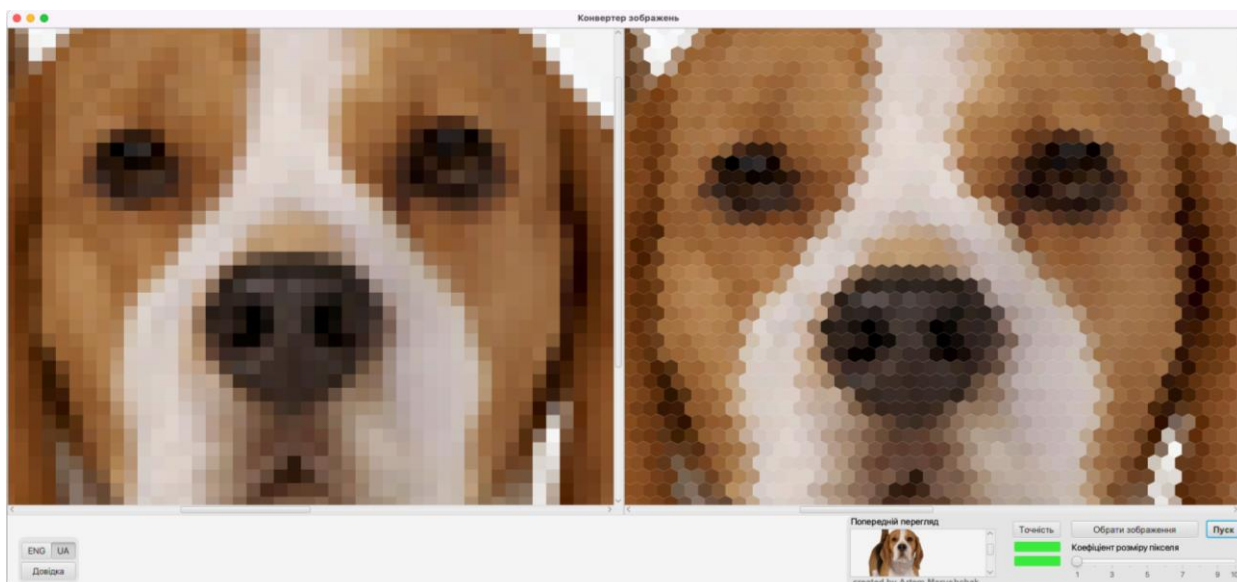


Рисунок 5.13 – Вигляд головного вікна із зміненим параметром точності

### 5.3 Програмні засоби для формування примітивів

Програму було розроблено мовою програмування Java, використовуючи середовище розробки IntelliJ IDEA. Проект був зібраний за допомогою інструментів Apache Maven, використовуючи JDK Java, і був розроблений у середовищі IntelliJ IDEA версії 2022. Для реалізації користувацького інтерфейсу та візуалізації результатів використовувалась бібліотека JavaFX. наведено зовнішній вигляд панелі налаштування програмного модуля. На рис. 5.14 зображено зовнішній вигляд інтерфейсу програмного модуля.

На рис. 5.15 і 5.16 наведено відповідно граф-схеми алгоритмів формування векторів і кіл на гексагональному растрі.

У режимі генерації векторів користувач вибирає кінцеві точки на екрані, клацаючи мишею по гексагонах робочої області. Коли ця дія завершена, програма генерує новий відрізок, базуючись на позиції кінцевої точки в пікселях. В результаті, програма показує на екрані вектор, подібно до того, як показано на рис. 5.17.

Передбачена можливість змінювати розмір гексагонів, які утворюють

екранну площину. Для формування кола задається його радіус, колір, розмір гексагонального пікселя. На рис. 5.18 і 5.19 зображено відповідно приклади генерації кіл у різних масштабах.

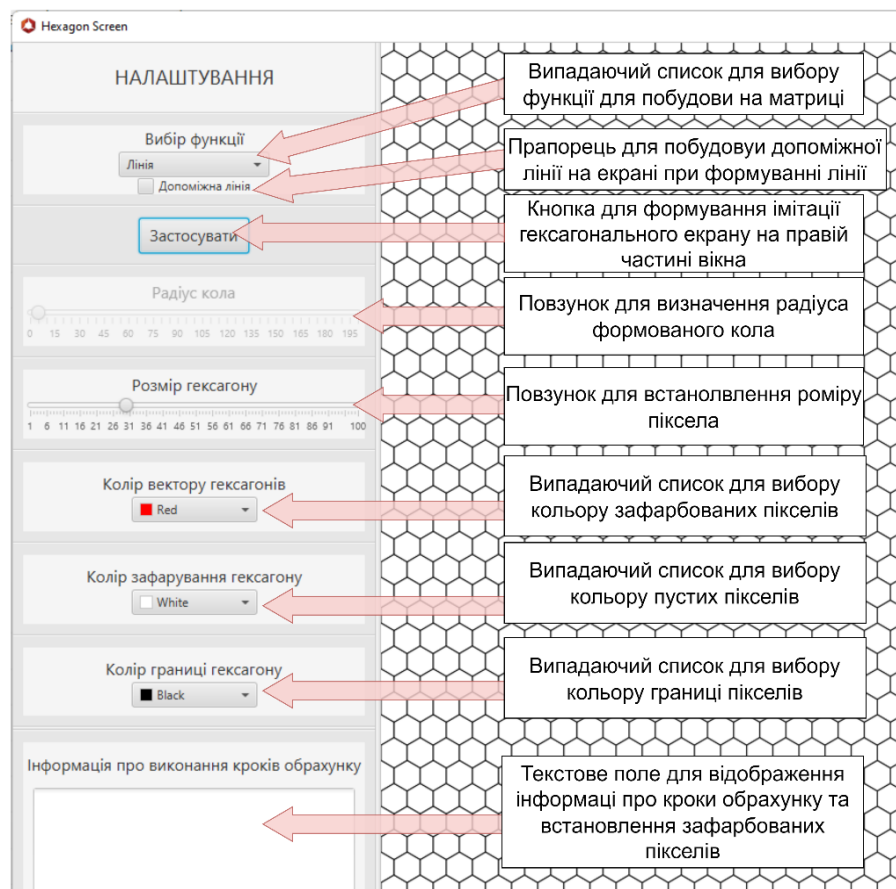


Рисунок 5.14 – Опис пунктів поля із налаштуваннями

Перед формування кола задається його радіус, колір, розмір гексагонального пікселя.

В програмному модулі реалізовано метод антиаліайзингу для векторів і кіл. При цьому використано дослідження, розроблені в підрозділі 4.1.

Для антиаліайзингу використано значення інтенсивностей кольору, які відображено в табл. 5.1.

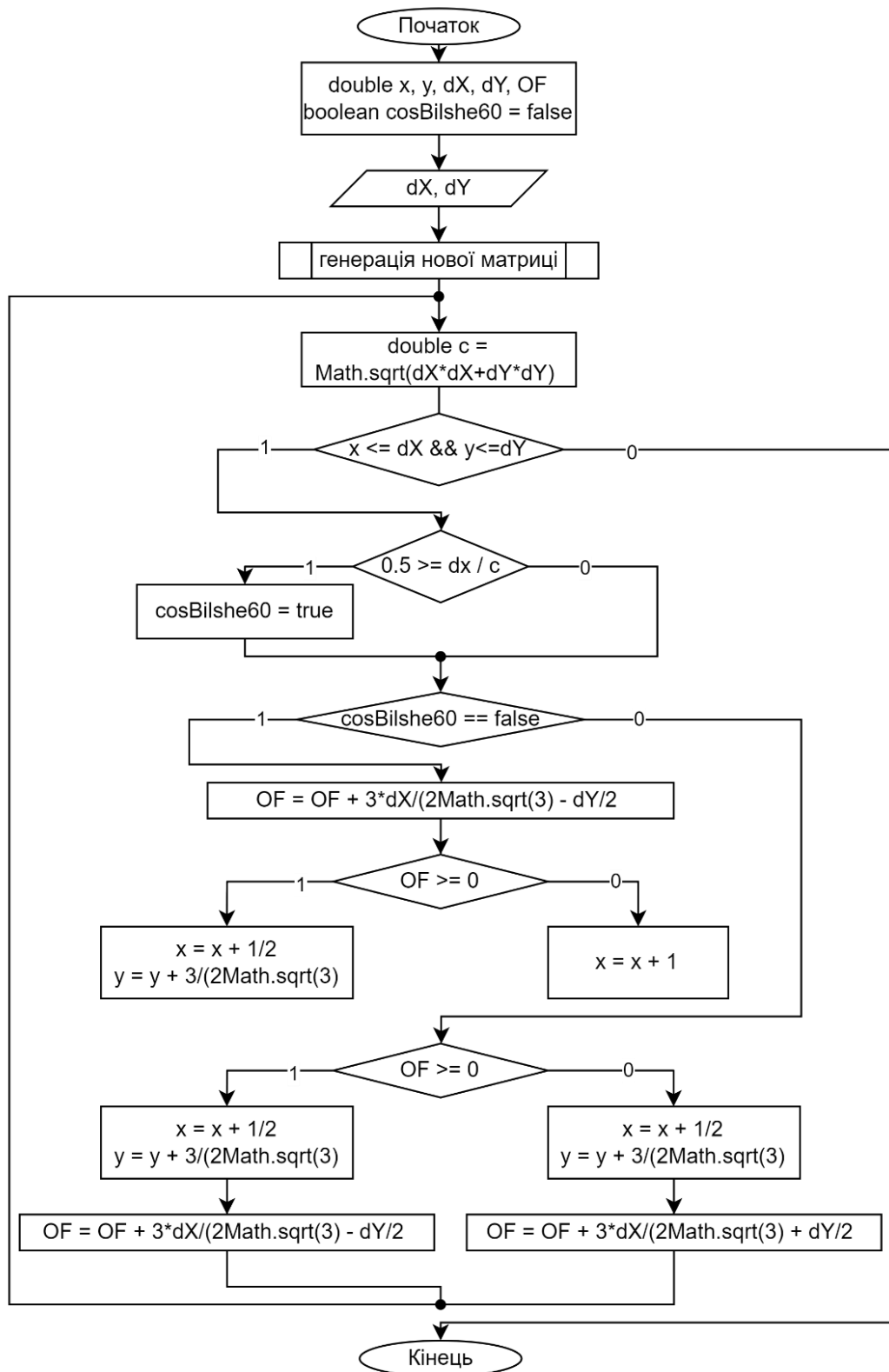


Рисунок 5.15 – Блок- схема алгоритму формування відрізка

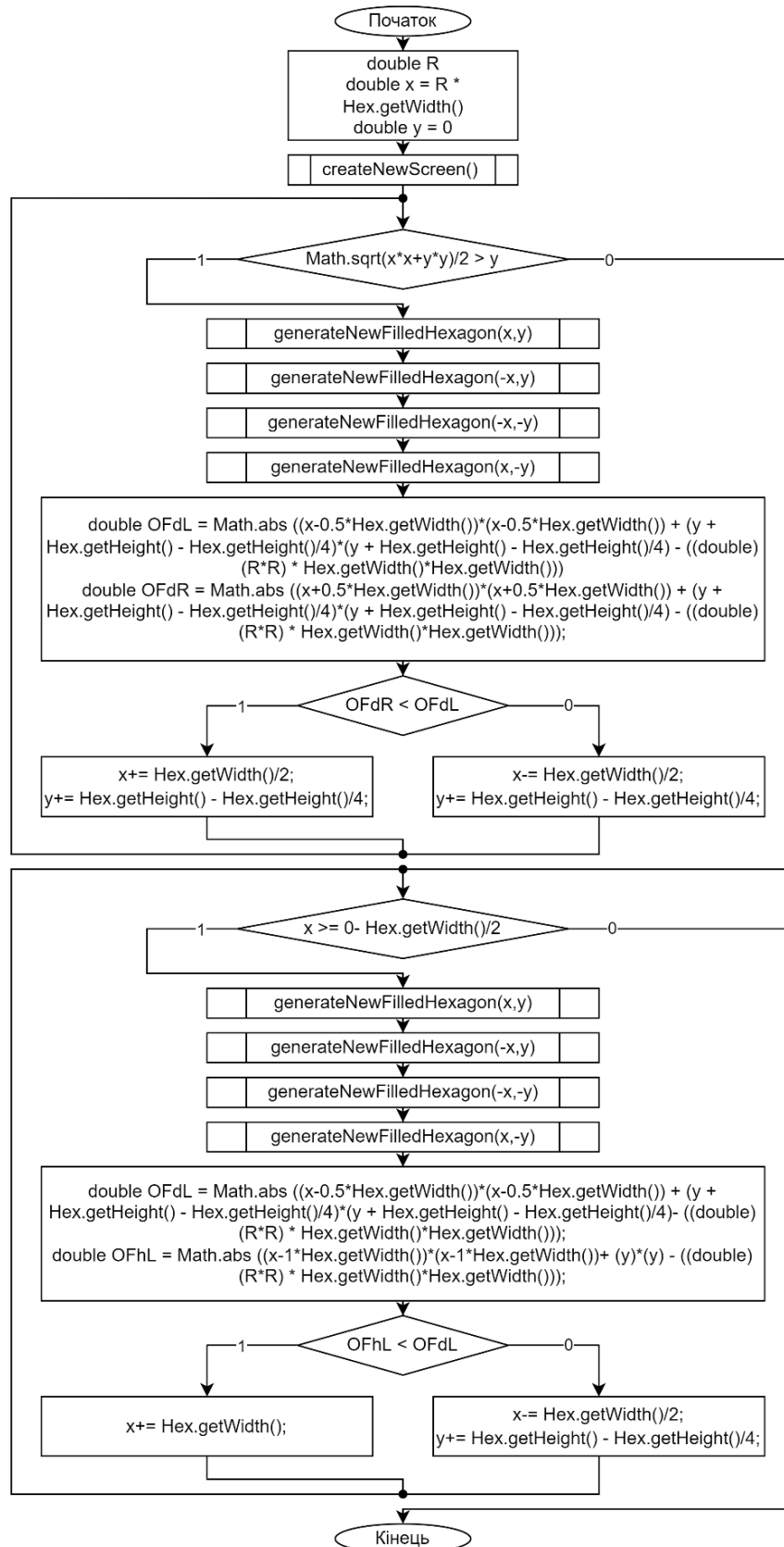


Рисунок 5.16 – Блок-схема алгоритму формування кола



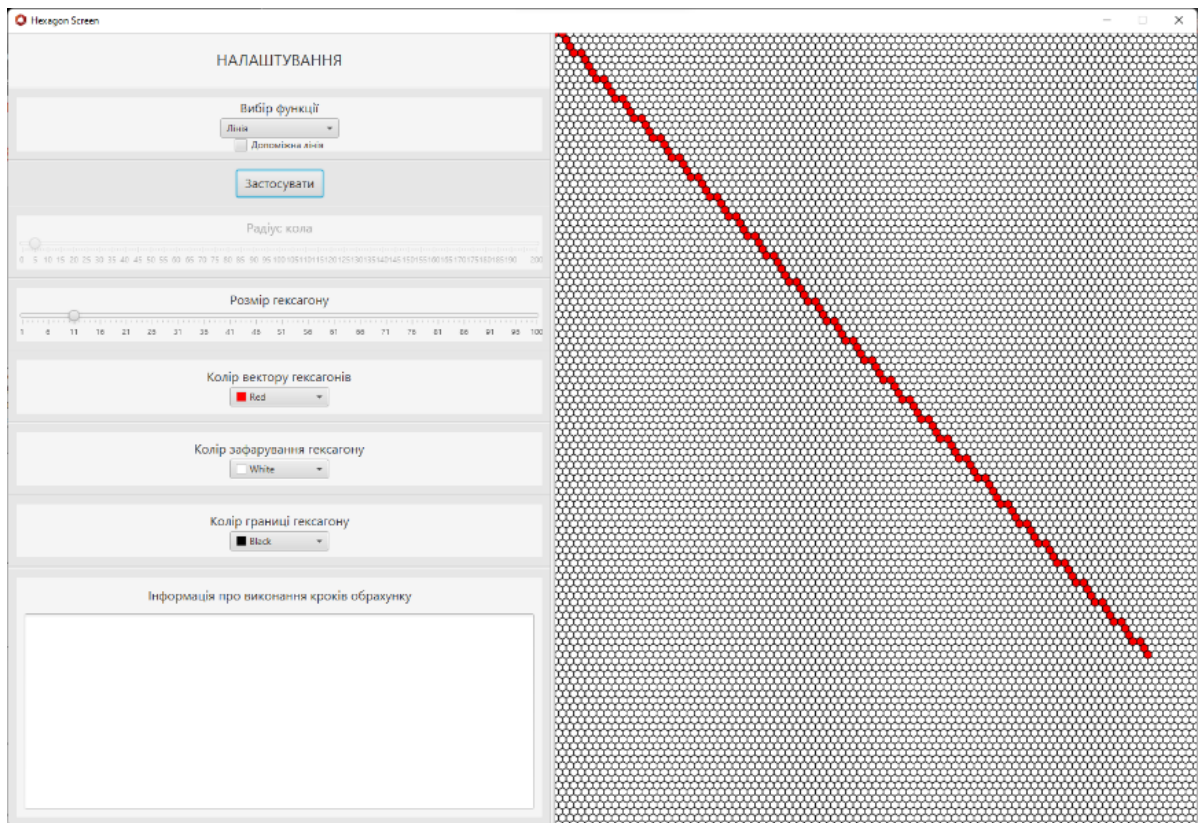


Рисунок 5.17 – Формування відрізка зі розміром гексагону в 11 пікселів

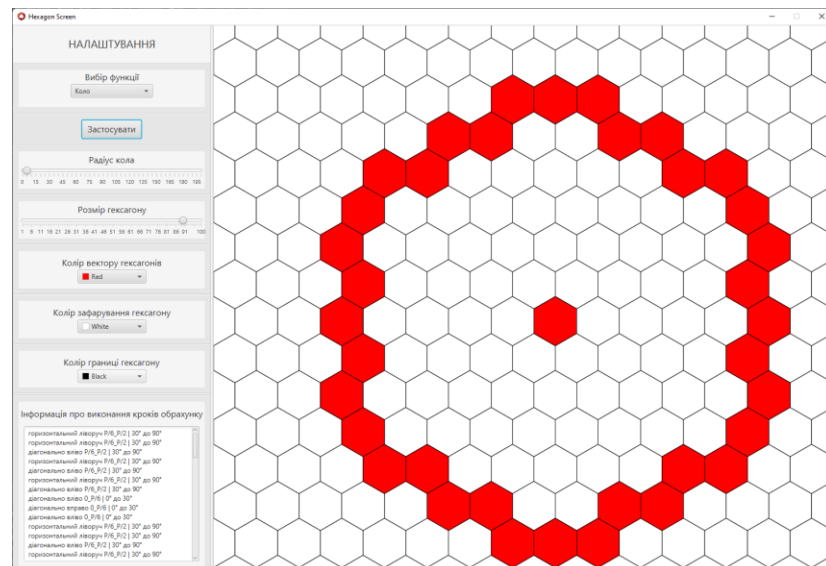


Рисунок 5.18 – Результат формування кола у великому масштабі



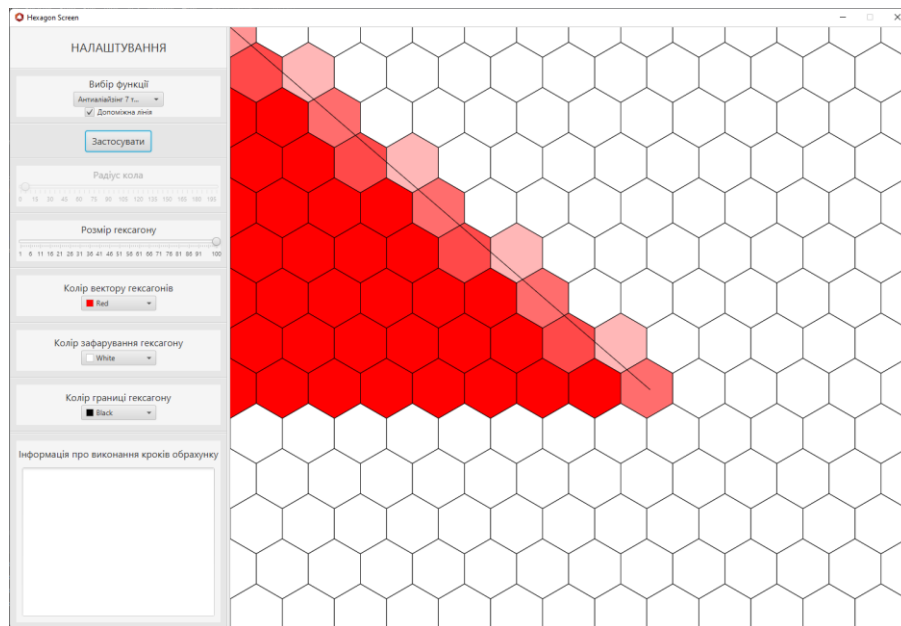


Рисунок 5.20 – Формування трикутника із антиаліазингом зображення крокової траєкторії

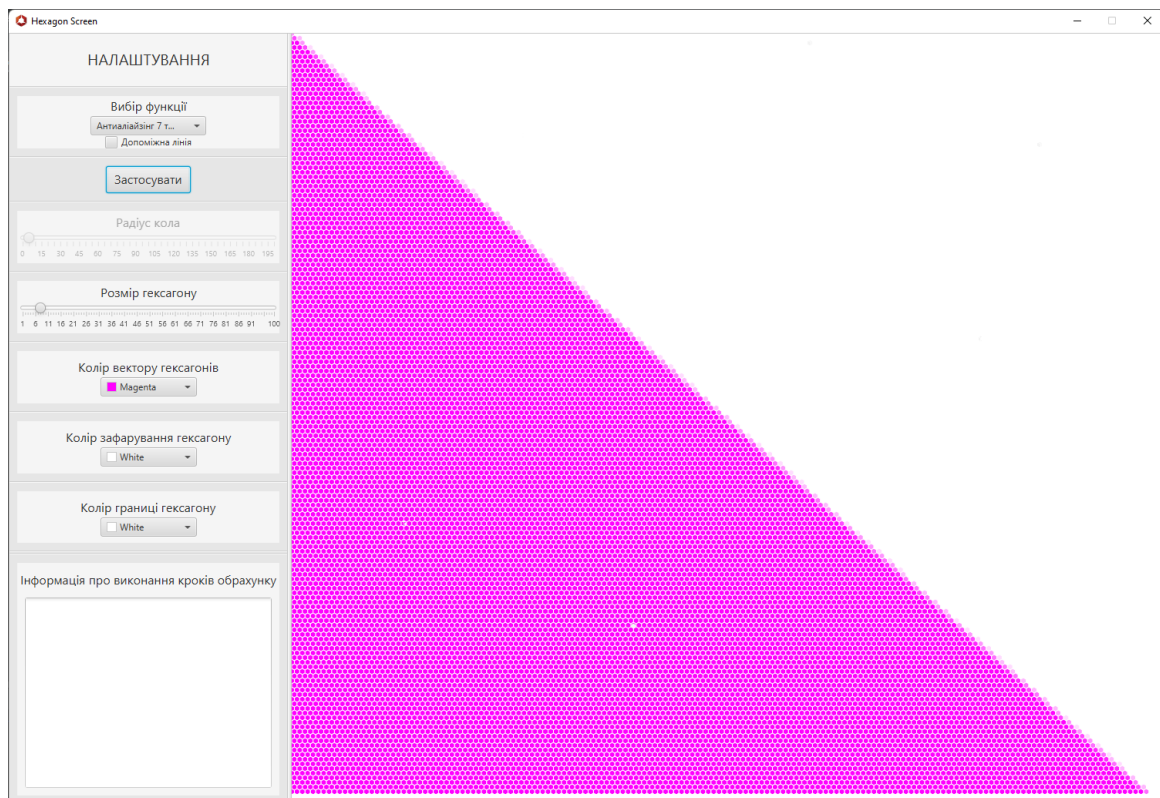


Рисунок 5.21 – Формування трикутника із антиаліазингом

#### 5.4 Апаратні засоби для формування примітивів

У розділі 2 розроблено метод формування покрокової траєкторії відрізка прямої на гексагональному растрі.

На рис. 5. 22 наведено структурну схему лінійного інтерполятора. Пристрій включає  $RG1$ ,  $RG2$  для зберігання операндів, які використовують для розрахунку оцінювальної функції; блок керування ключами  $BK$ ; лічильник  $CT2$  для визначення кінця інтерполяції, мультиплексор  $MX$ , нагромаджувальний суматор  $PSm$ , який включає регістр  $RG$  і комбінаційний суматор  $Sm$ ; регістр ознак  $RGS$ ; блок вихідної логіки для формування крокових переміщень  $OLB$ .

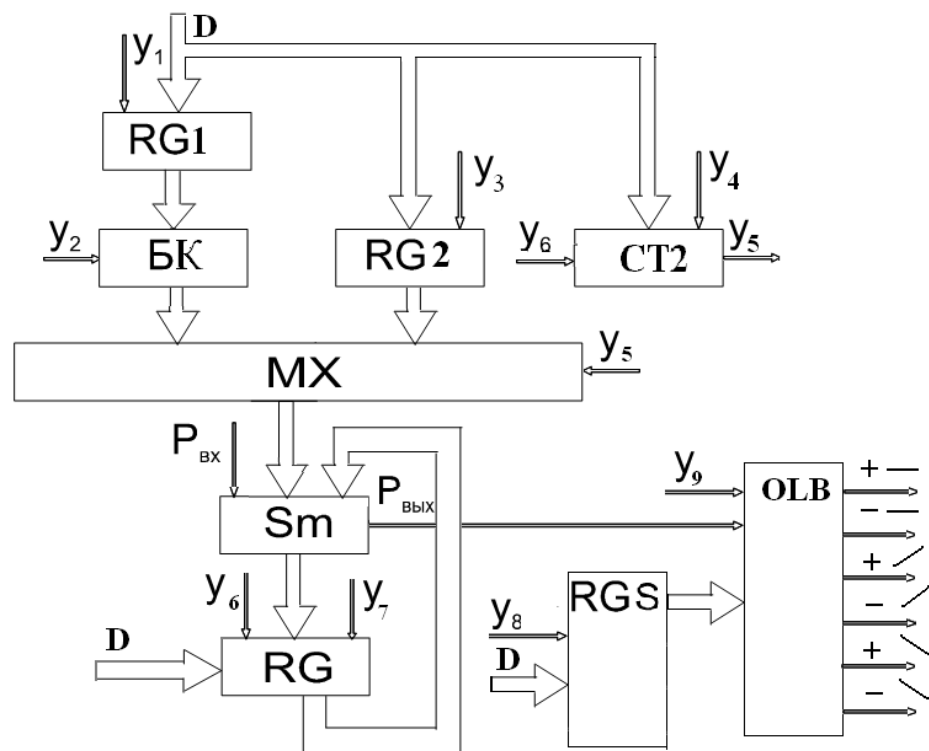


Рисунок 5. 22 – Структурна схема лінійного інтерполятора

Пристрій працює так.

У регістр  $RG$  заноситься значення  $BP/2$ , що визначає значення оцінювальної функції. У лічильник  $CT2$  записується сума координатних приростів відрізка прямих вздовж координатних приростів.

Для визначення значень  $OF$  у  $RG1, RG2$  заносяться відповідно

операнди  $(\Delta y \text{ або } \frac{\Delta y}{2}), \frac{3\Delta x}{2\sqrt{3}}$ . Значення  $(\Delta y \text{ або } \frac{\Delta y}{2})$  позначимо через  $O$ .

Для формування вихідних приростів у реєстр  $RGS$  записуються ознаки для формування відрізків прямих, зокрема, знак  $\Delta x$ , знак  $\Delta y$ , положення вектора. Ці дії визначають цикл підготовки.

В циклі інтерполяції в кожному такті визначається значення оцінювальної функції, для чого через мультиплексор подається один із операндів  $O, \frac{3\Delta x}{2\sqrt{3}}$  залежно від знака попередньої оцінювальної функції. Як

видно з табл. 5.3 при формуванні оцінювальної функції використовуються операнди  $\frac{\Delta y}{2}$  чи  $-\frac{\Delta y}{2}$  залежно від положення відрізка прямої по відношенню до координатних осей. Для перетворення з прямого коду в обернений використовується блок ключів, які реалізують логічну функцію нерівнозначності.

В нагромаджувальному суматорі до попереднього значення оцінювальної функції додається один з операндів, який приймає участь у формуванні  $OF_i$ . Вихідний перенос нагромаджувального суматора  $PSm$  визначає знак  $OF_i$ . На основі вмісту реєстру ознак  $RGS$  і знака оцінювальної функції на виході блок  $OLB$  вихідної логіки формується значення крокового приростів.

З кожним інтерполяційним тактом вміст лічильника  $CT2$  зменшується на одиницю до його нульового значення, про що свідчить одиничний рівень сигналу перенесення  $Y5$ . Після цього процес інтерполяції відрізка прямої завершують.





Особливість пристрою полягає у використанні для лінійного інтерполювання мікрооперації нагромаджувального додавання. Це дало можливість формувати крокові прирости в кожному інтерполяційному такті.

У підрозділі 3.3 розроблено метод формування відрізків прямих

комбінованими приростами на гексагональному растрі.

У таблиці 5.3 наведено типи можливих комбінацій координатних приростів і відповідні значення оцінювальних функцій. Раніше доведено, що на довільній ділянці координатного приросту можуть бути сформовані тільки два типи координатних приростів.

Таблиця 5.2 – Типи координатних приростів

Тип сполучення крокових переміщень	Формули для розрахунку $OF$
	$OF_{i+1} = OF_i - 2\Delta y$
	$OF_{i+1} = OF_i + \frac{\sqrt{3}}{2}\Delta x - \frac{3}{2}\Delta y$
	$OF_{i+1} = OF_i + \sqrt{3}\Delta x - \Delta y$
	$OF_{i+1} = OF_i + \sqrt{3}\Delta x$

На рис. 5.23 наведено структурну схему лінійного інтерполятора, особливість якого полягає у формуванні подвійних координатних приростів, що дає можливість підвищити продуктивність.

Пристрій включає  $RG1, RG2, RG3, RG4$  для зберігання операндів: які використовують для розрахунку оцінювальної функції, лічильник  $CT2$  для визначення кінця інтерполяції, мультиплексори  $MX1, MX2$ , суматор  $Sm1$  нагромаджувальний суматор  $PSm$ , який включає регістр  $RG5$  і комбінаційний суматор  $Sm2$ ; регістр ознак  $RGS$ : блок вихідної логіки для формування крокових переміщень  $OLB$ .

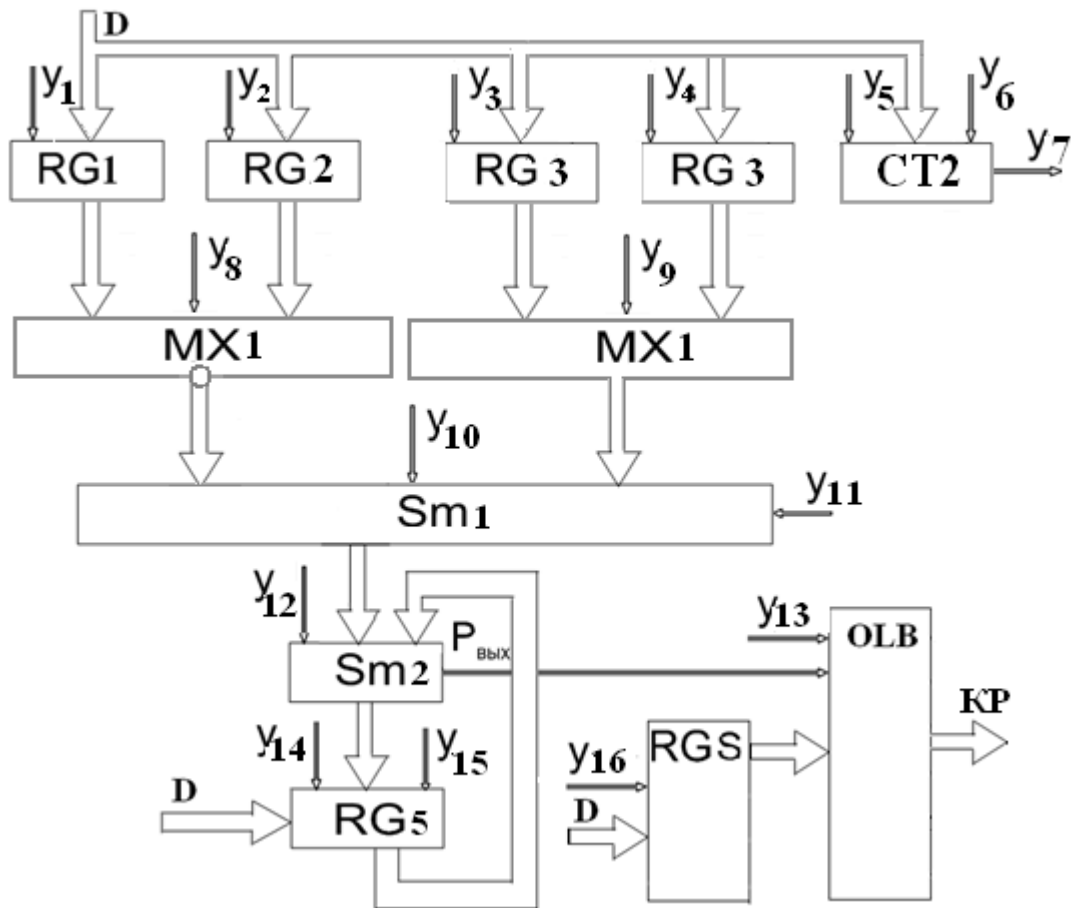


Рисунок 5.23 – Структурна схема лінійного інтерполятора з формуванням комбінованих пристроїв

Із табл. 5.3 видно, що при розрахунку оцінювальної функції всі множники з  $\Delta y$  мають від'ємний знак. При цьому для кожної ділянки інтерполювання використовується тільки два значення для розрахунку. Тому введено два регістри для зберігання пар операндів  $(2\Delta y, \frac{3}{2}\Delta y)$ ,  $(\Delta y, \frac{3}{2}\Delta y)$ ,  $(\Delta y, 0)$  відповідно для ділянок Д1, Д2, Д3 (див. рис. 3.10), а також мультиплексор  $MX2$  з інверсним виходом.

У регістр  $RG2$  заносяться множники, які співвідносять з  $\Delta x$ . А саме такі пари  $(0, \frac{\sqrt{3}}{2}\Delta x)$ ,  $(\sqrt{3}\Delta x, \frac{\sqrt{3}\Delta x}{2})$ ,  $(\sqrt{3}\Delta x, 0)$  відповідно для ділянок Д1, Д2, Д3 як на рис. 3.10.

На виході суматора  $Sm1$  формується операнд, який додається до

попереднього значення оцінювальної функції, яке зберігається в регістрі  $RG5$  нагромаджувального суматора.

Регістр ознак  $RGS$ , на відмінну від попередньої схеми містить ознаку парності суми приростів вихідного відрізка прямої. Перед початком інтерполяції в лічильник  $CT2$  заноситься ціла частина половини суми приростів відрізка прямої. Цей лічильник використовується для визначення закінчення інтерполяції. За умови парності суми приростів вихідного відрізка прямої при формуванні сигналу нульового значення лічильника  $CT2$ , процес інтерполювання закінчується. В протилежному випадку виконується ще один такт інтерполяції, однак другий розряд приросту блокується.

Вихідний перенос нагромаджувального суматора  $Psm$  визначає знак оцінювальної функції, який разом з ознаками, що зберігається в регістрі  $RGS$ , використовується для формування.

Особливість пристрою полягає у формуванні комбінованих приростів, що дає можливість підвищити продуктивність.

Розроблений у підрозділі 3.2 алгоритм кругової інтерполяції може бути реалізований пристроєм, структурну схему якого наведено на рис. 5. 24. Пристрій містить нагромаджувальний суматор, два координатних лічильника, блок ідентифікації ділянки інтерполювання, комутатор і блок керування.

Пристрій працює так.

В циклі підготування в лічильник  $X$  заноситься абсциса початкової точки кола щодо центру, а в лічильник  $Y$  - координата початкової точки.

В нагромаджувальний суматор заноситься початкове значення оцінювальної функції. Надалі, в кожному такті, в нагромаджувальному суматорі обчислюється значення оцінювальної функції в точках, відповідних такому направленню руху, яке сприяє зміні знаку оцінювальної функції. Розрахунок оцінювальної функції проводиться відповідно до виразів, наведених в таблиці 3.1.

Знак оцінювальної функції поступає з нагромаджувального суматора в



блок керування.

В останньому формуються керуючі сигнали, які поступають на виходи інтерполятора і на входи координатних лічильників.

Блок визначення різниці служить для визначення ділянки координатної площини, де відбувається інтерполювання.

Введення в інтерполятор початкової установки накопичувального суматора дозволяє підвищити точність інтерполяції порівняно з раніше відомим пристроєм.

На рис. 5.25 наведено структурну схему блоку антиаліазингу для визначення основної та додаткових оцінювальних функцій. Блок розраховує інтенсивності кольору, які залежать від площі ділянки пікселя.

Пристрій реалізовано згідно методу антиаліазингу, який запропоновано в підрозділі 4.2.

Основну оцінювальну функцію визначають на основі приростів  $\Delta x$ ,  $\Delta y$ . У блоці розрахунку додаткових  $OF_1 - OF_{12}$ , знаходять  $OF_i$ , знаки яких ідентифікується випадок перетину пікселя вектором. Блок для визначення площі сегмента пікселя може бути реалізований як пристрій постійної пам'яті, на адресні входи якого подаються знаки  $OF_1 - OF_{12}$ .

Розроблені пристрої можуть бути використані в графічних акселераторах для підвищення продуктивності формування графічних сцен.

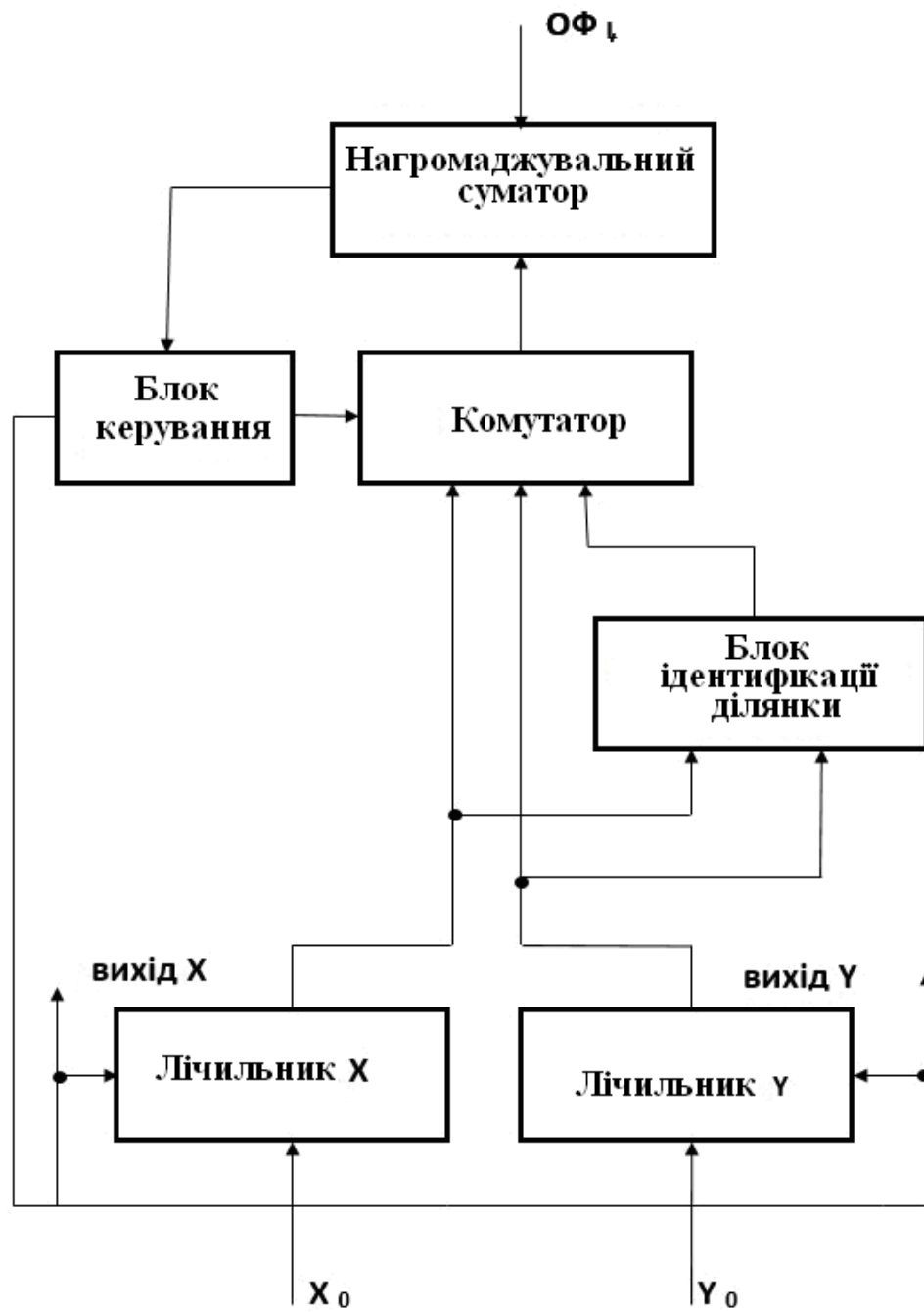


Рисунок 5. 24 – Структурна схема кругового інтерполятора

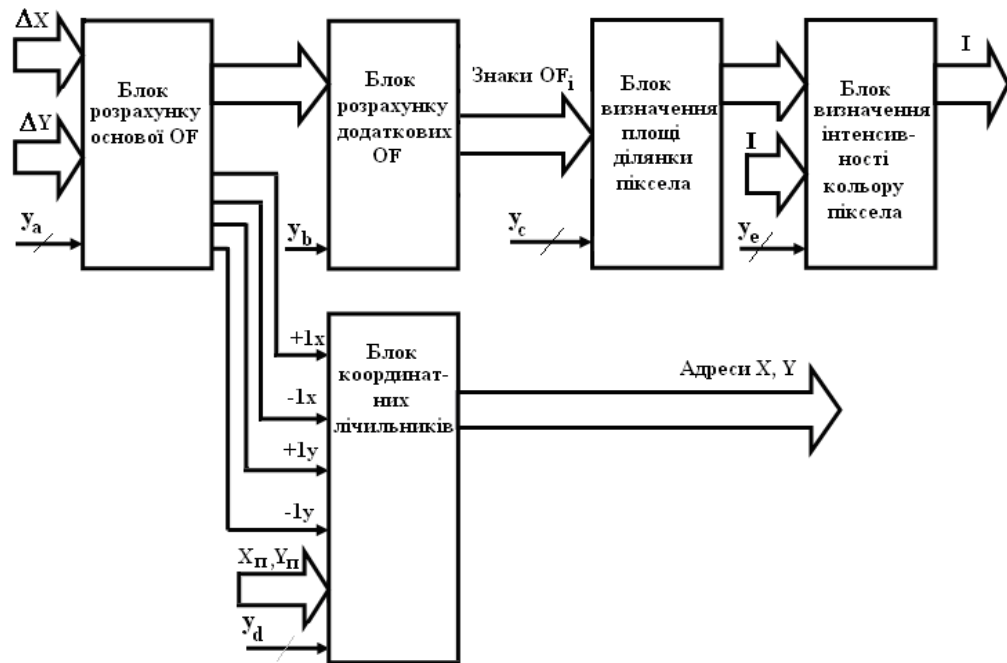


Рисунок 5. 25 – Структурна схема блоку антиаліазингу

### 5.5 Висновки до розділу 5

1. Розроблено графічний редактор для формування графічних зображень на гексагональному растрі. Редактор має функціональні можливості, достатні для формування зображень на гексагональному растрі.

2. Для конвертації зображень з прямокутного в гексагональний растр розроблено відповідний конвектор.

3. Розроблено програмні засоби для формування відрізків прямих і кіл на гексагональному растрі. Наведено граф-схеми алгоритмів.

4. Розроблено структурні схеми пристроїв для формування графічних примітивів на гексагональному растрі, а також для антиаліазингу зображень крокових траєкторій.

Результати досліджень цього розділу наведено в публікаціях: [45], [46], [48], [50], [105], [109], [113], [115].

## ВИСНОВКИ

У дисертаційній роботі розвинуто теорію кінцевої візуалізації графічних об'єктів, що дозволило вирішити задачу підвищення продуктивності та реалістичності формування зображень у системах комп'ютерної графіки з використанням нових методів формування графічних примітивів на гексагональному растрі. Вирішення даної задачі дозволило створити нові алгоритми та структури пристроїв формування графічних примітивів, що створює умови для побудови на їх основі конкурентоспроможних систем візуалізації.

У ході досліджень було отримано такі результати:

1. Подальшого розвитку отримав метод оцінювальної функції для формування відрізків прямих на гексагональному растрі, який відрізняється від відомих визначенням в кожному інтерполяційному такті подвійних крокових приростів, що дозволило до двох разів зменшити час лінійної інтерполяції.

2. Вперше запропоновано метод кругової інтерполяції на гексагональному растрі, особливість якого полягає у використанні апріорно визначеному стохастичного розподілу крокових приростів залежно від ділянки формування крокової траєкторії, що дало можливість в 1,7 збільшити швидкодію інтерполяції за рахунок прогнозування найбільш вірогідної комбінації кроків.

3. Подальшого розвитку отримав метод антиаліайзингу векторів на гексагональному растрі, який відрізняється від відомих використанням для обчислення площі покриття пікселя додаткових оцінювальних функцій, розрахованих у виділених точках контуру гексагону, що дозволило виконувати антиаліайзинг безпосередньо під час формування зображення крокової траєкторії та усунути етап постобробки, і, як наслідок, підвищити продуктивність формування зображення згладженої траєкторії.

4. Подальшого розвитку отримав метод антиаліайзингу векторів на гексагональному растрі, який відрізняється від відомих визначенням площі

покриття пікселя шляхом визначення знаків оцінювальних функцій в центрах субпікселів, на які розбивається піксель, що дозволило розпаралелити процес антиаліазингу та підвищити його продуктивність.

5. Вперше отриманні співвідношення для визначення типів крокових переміщень для еліпсів і кіл залежно від ділянки формування крокової траєкторії, що спрощує методи формування еліпсів за рахунок влучення надлишкових обчислень.

6. Створено діючі програмні засоби для формування тривимірних графічних об'єктів у системах комп'ютерної графіки, які захищено захищені свідоцтвами на реєстрацію авторського права на твір у державному департаменту інтелектуальної власності України.

7. На основі запропонованих методів розроблено структурні схеми пристроїв для апаратної реалізації графічних примітивів на гексагональному растрі.

8. Результати дисертаційної роботи впроваджено на таких підприємствах «ДРЕССЛАБ Юей» (акт від 03.08.2022 р.), ТОВ «3Д Дженерайшн Юей» (акт від 27.04.2023 р.), ПП «Фотоніка плюс» (акт від 24.04.2023 р.), кафедрі програмного забезпечення Вінницького національного технічного університету для використання у навчальному процесі (акт від 17.05.2023 р.).

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. S. Marschner and P. Shirley, *Fundamentals of Computer Graphics*, Boca Raton, USA: CRC Press, 2021.
2. L. Moody, *Computer Graphics and Multimedia*, Wilmington, USA: Kaufman Press, 2022.
3. A. Godse and D. Godse, *Computer Graphics and Multimedia: Concepts, Algorithms and Implementation using C*, Pune, India: Technical Publications, 2020.
4. M. Tistarelli, S. Dubey, S. Kumar Singh and X. Jiang, *Computer Vision and Machine Intelligence*, London, UK of Great Britain: Springer, 2023.
5. P. Shukla, R. Aluvalu, S. Gite and U. Maheswari, *Computer Vision: Applications of Visual AI and Image Processing*, Berlin, Germany: De Gruyter, 2023.
6. L. Middleton and J. Sivaswamy, *Hexagonal Image Processing: A Practical Approach*, London, UK of Great Britain: Springer, 2005.
7. O. Melnik, O. Romanyuk, and V. Savratsky, *Applying of hexagonal raster in image formation. Monography*, Boston, USA, 2020.
8. K. T. Tytkowski, "Hexagonal raster for computer graphic", in *Proc. IEEE Conference on Information Visualization*, London, UK, 2000, pp. 69-73.
9. H. Si-Tao, "Real RGB printing AMOLED with high pixel per inch value", *Journal of the Society for Information Display*, pp. 36-43, 2020.
10. Y. Liu and J. Shi, "Display of graphics on hexagonal grids", in *2000 IEEE Conference on Information Visualization. An International Conference on Computer Visualization and Graphics*, London, 2004, pp. 331-336.
11. Heroes of Might and Magic 3. [Електронний ресурс]. Доступно: <https://www.ubisoft.com/ru-ru/game/heroes-of-might-and-magic-3-hd>. Дата звернення: 09.2016.
12. О. Н. Романюк, О. В. Мельник, та Ю. О. Панфілова, "Використання гексагонального растру в комп'ютерних іграх", на *XII Міжнародній науково-*

технічній конференції «Інформаційні комп'ютерні технології – 2021 (ІКТ-2021)», Житомир: Житомирська політехніка, 1-3 квітня 2021, с. 5-6.

13. A. J. Cho and J. Y. Kwon, “Hexagonal boron nitride for surface passivation of two-dimensional van der Waals heterojunction solar cells”, *ACS Appl. Mater. Interfaces*. Vol. 11, pp. 39765–39771, October 2019.

14. W. Wen and S. Khatibi, “Virtual deformable image sensors: towards to a general framework for image sensors with flexible grids and forms”, *Image Sensors*, vol. 18(6), pp. 1-16, 2018.

15. G. Iacobucci, R. Cardarelli, S. Débieux, F. A. Di Bello, Y. Favre, D. Hayakawa, and P. Valerio, “A 50 ps resolution monolithic active pixel sensor without internal gain in SiGe BiCMOS technology”, *Journal of Instrumentation*, vol. 14(11), pp. 1-9, 2019.

16. О. Н. Романюк, О. В. Мельник та Л. Г. Коваль, “Використання гексагональних комірок у видавничій справі”, на *П'ятій міжнародній науковій конференції «Інформація, комунікація та управління знаннями в глобалізованому світі»*, Київ, 23-24 червня 2022, с. 45-47.

17. Особливості виготовлення друкарських форм для трафаретного друку. [Електронний ресурс]. Доступно: [http://4ua.co.ua/manufacture/ra2bc78b4d43b89421316d37\\_0.html](http://4ua.co.ua/manufacture/ra2bc78b4d43b89421316d37_0.html). Дата звернення: 08.2023.

18. D. Edler, J. Keil, A. Bestgen, L. Kuchinke and F. Dickmann, “Hexagonal map grids – an experimental study on the performance in memory of object locations”, *Cartography and Geographic Information Science*, Vol. 46, Issue 5, pp. 401-411, 2019.

19. A. Mahdavi-Amiri, E. Harrison and F. Samavati, “Hexagonal connectivity maps for Digital Earth”, *International Journal of Digital Earth*, vol. 8, Issue 9, pp. 750-769, 2015.

20. Модуль фотоелектричного протеза з гексагональними пікселями. [Електронний ресурс]. Доступно: [http://www.researchgate.net/publication/266659406\\_Pinlight\\_Displays\\_Wide\\_Field\\_of\\_View\\_Augmented\\_Rea](http://www.researchgate.net/publication/266659406_Pinlight_Displays_Wide_Field_of_View_Augmented_Rea)

lity\_Eyeglasses\_using\_Defocused\_Point\_Light\_Sources. Дата звернення: 10.2021.

21. Гексагональний масив мікролінз Лазерного мікроскопу ZEISS LSM 800. [Електронний ресурс]. Доступно: <http://www.intechopen.com/books/microscopy-and-analysis/super-resolution-confocal-microscopy-through-pixel-reassignment>. Дата звернення: 12.2019.

22. Дисплей окулярів доповненої реальності Google Glass. [Електронний ресурс]. Доступно: <http://www.laserfocusworld.com/articles/print/volume-50/issue-10/world-news/augmented-reality-displays-pinlight-arrays-enable-lightweight-see-through-head-worn-display.html>. Дата звернення: 11.2015.

23. О. Н. Романюк та М. С. Курінний, *Методи та засоби антиаліайзингу контурів об'єктів у системах комп'ютерної графіки*, Вінниця, Україна: УНІВЕСУМ-Вінниця, 2006.

24. M. Dippe, *Antialiasing in Computer Graphics*, University of California, Berkeley, USA, 1985.

25. J. Jimenez, J. Echevarria, T. Sousa and D. Gutierrez, "SMAA: Enhanced Subpixel Morphological Antialiasing", *EUROGRAPHICS 2012*, volume 31, number 2, 2012.

26. О. Н. Романюк, О. В. Мельник та С. І. Вяткін, "Класифікація методів антиаліайзингу", *Вісник Херсонського національного технічного університету*, Херсон, ХНТУ, №3 (50), с. 154-159, 2014р.

27. О. Н. Романюк, М. С. Курінний та О. В. Мельник, "Антиаліайзинг зображення відрізків прямих з використанням нової моделі пікселя", *Оптико-електронні інформаційно-енергетичні технології. Міжнародний науково-технічний журнал*. №2 (20), с. 30-35, Вінниця, ВНТУ, 2010.

28. Anti-aliasing, [Online]. Available: <https://docs.unity3d.com/Packages/com.unity.postprocessing3.0/manual/Anti-aliasing.html> Accessed on: 10.2023.



29. X. He, and W. Jia, “Hexagonal structure for intelligent vision”, in *2005 International Conference on Information and Communication Technologies, IEEE, Karachi, Pakistan*, pp. 52-64, 2005.

30. О. Н. Романюк, О. В. Романюк та Р. Ю. Чехмestrucк, *Комп’ютерна графіка: навчальний посібник*. Вінниця, Україна: ВНТУ, 2022.

31. О. Н. Романюк та М. С. Курінний, “Математичні моделі пікселів для задач антиаліаїзінгу”, *Вісник Житомирського інженерно-технологічного інституту*, випуск №3, с. 35-47, 2002.

32. О. Н. Романюк, М. С. Курінний, О. В. Мельник та С. В. Олійник, “Кругова модель пікселя для задач антиаліаїзінгу”, *Наукові праці Донецького національного технічного університету. Серія «Інформатика, кібернетика та обчислювальна техніка»*, Випуск 15(203), Донецьк, с. 90-94, 2012.

33. О. Н. Романюк, О. В. Мельник, І. В. Абрамчук та О. О. Дудник, “Модифікація гаусівської моделі пікселя для задач антиаліаїзінгу”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*. № 1, с. 84-88, 2015.

34. О. Н. Романюк, І. В. Абрамчук та О. О. Дудник, “Анізотропна фільтрація з використанням вагової функції на основі гаусівської моделі пікселя”, *Вимірювальна та обчислювальна техніка в технологічних процесах*, випуск № 2, с. 117-121, 2016.

35. С. О. Романюк, О. Н. Романюк, С. В. Павлов, та О. В. Мельник, “Модель для відтворення спекулярної складової кольору в засобах комп’ютерної графіки”, *Інформаційні технології та комп’ютерна інженерія*, ВНТУ, №3(34), с. 50-57, 2015.

36. О. Н. Романюк та О. В. Мельник, “Особливості використання гексагонального растра при побудові пристроїв відображення”, *Міжнародний науково-технічний журнал «Вимірювальна та обчислювальна техніка в технологічних процесах»*, Хмельницький, №3 (56), с. 105-109, 2016.

37. О. Н. Романюк, О. В. Романюк та О. В. Мельник, “Формування відрізків прямих на гексагональному растрі”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*, № 2, с. 69-72, 2016.

38. О. Н. Романюк, І. В. Абрамчук та О. В. Мельник, “Визначення типів крокових приростів для побудови кола на гексагональному растрі”, *Вісник Хмельницького національного університету. Серія: Технічні науки, Хмельницький, ХНУ, №3(249)*, с. 172-176, 2017.

39. О. Н. Романюк, О. В. Мельник, І. В. Абрамчук та Н. С. Костюкова, “Особливості формування еліпсів на гексагональному растрі”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*, № 1(26), с. 86-90, 2018.

40. О. Н. Романюк, С. О. Романюк, О. В. Мельник, та Д. А. Озерчук, “Особливості формування еліпсів, повернутих на заданий кут, на гексагональному растрі”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*, № 2(31), с. 23-28, 2020.

41. О. Н. Романюк, О. В. Мельник, С. В. Котлик, С. О. Романюк, та Р. Ю. Чехмestрук, “Методи підвищення продуктивності формування векторів на гексагональному растрі”, *Автоматизація технологічних і бізнес-процесів*, Volume 14, Issue 3, с. 27-36, 2022. DOI: 10.15673/atbp.v14i3.2350

42. О. Н. Романюк, О. В. Мельник та В. А. Шмалюх, “Метод прискореної кругової інтерполяції на гексагональному растрі”, *Вісник ВПІ*, № 2, с. 81–88, 2023. doi.org/10.31649/1997-9266-2023-167-2-81-88

43. О. Н. Романюк, М. С. Курінний, О. В. Мельник, та Н. С. Костюкова, “Високопродуктивна конусна модель пікселя для антиаліайзингу відрізків прямих”, *Наукові праці Донецького національного технічного університету. Серія «Інформатика, кібернетика та обчислювальна техніка»*, Випуск 14(188), Донецьк, с. 216-220, 2011.

44. О. Н. Романюк та О. В. Мельник, “Морфологічний антиаліайзинг для гексагонального растру”, на *III Міжнародній науково-практичній конференції «Актуальні проблеми гуманітарних та природничих наук»*, Київ, 28-29 жовтня 2016, с. 112-115.

45. О. Н. Романюк, О. В. Мельник та О. В. Стукач, “Моделювання гексагонального растра на квадратному растрі”, на *Сьомій міжнародній науково-технічній конференції «Моделювання і комп'ютерна графіка»*, Київ, 18-24 вересня 2017, с.18-24.

46. О. Н. Романюк, О. В. Мельник, та Ю. О. Панфілова, “Формування параболи на гексагональному растрі”, на *Міжнародній науково-практичній Інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ»*, Суми-Вінниця, 9-10 листопада 2021, с. 196-198.

47. О. Н. Романюк, О. В. Романюк, О. В. Мельник, та Р. Ю. Чехместрук, “Суперсемплінг зображень, сформованих на гексагональному растрі”, in *Modern research in world science Proceedings of the 3rd International scientific and practical conference. SPC “Sci-conf.com.ua”*, Lviv, Ukraine, 2022, pp. 517-522.

48. О. Н. Романюк, О. В. Мельник та В. А. Шмалюх, “Розробка застосунку для формування відрізків на гексагональному растрі”, на *Всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи»*, Вінниця, 2022, с 1-7.

49. О. Н. Романюк, М. В. Козубенко, О. В. Мельник та С. В. Котлик, “Використання гексагонального растру в картографії», на *XV Міжнародній науково-практичній конференції Інформаційні технології і автоматизація – 2022*, Одеса, 20-21 жовтня 2022, с. 30-33.

50. О. Н. Романюк, О. В. Романюк, та О. В. Мельник, “Програмна реалізація графічного редактора на гексагональному растрі”, in *The 8th International scientific and practical conference “Modern research in world science”*, (October 29-31, 2022) SPC “Sci-conf.com.ua”, Lviv, Ukraine, 2022, pp. 389-392.

51. О. Н. Романюк, О. В. Мельник, Р. Ю. Чехмestrucк та С. О. Романюк, “Основні співвідношення гексагонального растру”, на VII Міжнародній науково-практичній конференції Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі, Київ, 2022, с. 59-61.

52. O. N. Romanyuk, A. V. Melnik, A. P. Goncharuk, and Y. L. Lyashenko, “Effective Models for the Specular Color Constituent Computing”, *Journal of Computer Science and Engineering*, Volume 2, Issue 2, pp. 25-29, august 2010.

53. S. O. Romanyuk, S. V. Pavlov and O. V. Melnyk, “New method to control color intensity for antialiasing”, in *Control and Communications (SIBCON)*, 2015 International Siberian Conference on 02 July 2015, DOI: 10.1109/SIBCON.2015.7147194.

54. O. Romanyuk, S. Pavlov, O. Melnyk, S. Romanyuk, A. Smolarz and M. Bazarova, “Method of anti-aliasing with the use of the new pixel model”, in *Optical Fibers and Their Applications*, Lublin, Poland, 2015, pp. 274-278, <https://doi.org/10.1117/12.2229013>.

55. D. Kuri, E. Root and H. Theisel, “Hexagonal Image Quilting for Texture Synthesis”, in *WSCG 2014 Conference on Computer Graphics, Visualization and Computer Vision*, 2014, pp.67-76.

56. С. В. Ходус, та О. В. Самощенко, "Просторове представлення цифрових зображень у гексагональній формі", на «Информатика и компьютерные технологии-2009», Краматорськ, 2009, с. 163-166.

57. P. Burt, “Tree and pyramid structures for coding hexagonally sampled binary images”, *Computer Graphics and Image Processing*, Volume 14, Issue 3, pp. 271-280, 1980.

58. D. J. Whitehouse and M. J. Phillips, “Sampling in a two-dimensional plane”, *Journal of Physics*, Volume 18, pp. 2465-2477, 1985.

59. R. Vitulli, “Aliasing Effects Mitigation by Optimized Sampling Grids and Impact on Image Acquisition Chains”, in *Geoscience and Remote Sensing Symposium*, New York, 2022, pp. 979–981.

60. R. Mersereau, "The processing of hexagonally sampled two-dimensional signals", *Proceedings of the IEEE*, Volume 67, Issue 6, pp. 930-949, 1979, DOI: 10.1109/PROC.1979.11356.
61. M. Golay, "Hexagonal Parallel Pattern Transformations", *IEEE Transactions on Computers*, Volume 18, Issue 8, pp. 733–740, 1969.
62. R. Staunton, "The design of hexagonal sampling structures for imagedigitisation and their use with local operators", *Image and Vision Computing*, volume 7 (3), pp. 162- 166, 1989.
63. M. Azam, M. Anjum and M. Javed, "Discrete cosine transform (DCT) based face recognition in hexagonal images", in *proceedings of the 2010 The 2-nd International Conference on Computer and Automation Engineering (ICCAE)*, Singapore, 26–28 February 2010, pp. 474–479.
64. M. Sharif, A. Khalid, M. Raza and S. Mohsin, "Face detection and recognition through hexagonal image processing", *Sindh University Research Journal*, Vol. 44(4), pp. 541–548, 2012.
65. T. Cevik, N. Cevik, J. Rasheed, A. Abu-Mahfouz and O. Osman, "Facial Recognition in Hexagonal Domain - A Frontier Approach", *IEEE Access*, Volume 11, pp. 46577-46591, 2023, DOI: [10.1109/ACCESS.2023.3274840](https://doi.org/10.1109/ACCESS.2023.3274840).
66. P. Duszak, "SLAM on the Hexagonal Grid", *Multidisciplinary Digital Publishing Institute*, vol. 22, no. 16, 2022, Article number: 6221, pp. 1-15, DOI:10.3390/s22166221.
67. N. Cevik, T. Cevik, O. Osman, A. Gurhanli, S. Nematzadeh and F. Sahin, "Improved exploiting modification direction steganography for hexagonal image processing", *Computer and Information Sciences*, volume 34, Issue 10, pp. 9273-9283, 2022, <https://doi.org/10.1016/j.jksuci.2022.09.007>.
68. V. Prathibha, and G. Arockia Selva Saroja, "An efficient hexagonal image framework using pseudo hexagonal pixel for computer vision applications", *Journal of Intelligent & Fuzzy Systems*, vol. 42, no. 4, pp. 3879-3892, 2022.
69. A. Garliauskas, "The visual cortex modeling by the hexagonal topology", *Neurocomputing*, Volumes 38–40, pp. 1229-1238, 2001.

70. N. Kartheek Medathati, H. Neumann, Guillaume S. Masson and P. Kornprobst, "Bio-Inspired Computer Vision: Towards a Synergistic Approach of Artificial and Biological Vision", *Computer Vision and Image Understanding*, volume 150, pp. 1-30, 2016, <https://doi.org/10.1016/j.cviu.2016.04.009>.

71. Stimets, W. Richard and H. Tang. "Procedures for rapid pattern recognition of three-dimensional objects using parallel processing on a hexagonal pixel grid", *Image and Video Processing IV*. Vol. 2666. SPIE, 1996.

72. М. М. Гінзбург та Є. П. Путятін, "Порівняльний аналіз прямокутної та гексагональної ґраток для дискретизації кривих", *Біоніка інтелекту № 2 (79)*, с. 13-18, 2012.

73. Peter R. Massopust, Patrick J. Van Fleet, "Fractional cone and hex splines", *Rocky Mountain J. Math.* 47 (5) pp. 1655-1691, 2017, <https://doi.org/10.1216/RMJ-2017-47-5-1655>.

74. D. Van De Ville, T. Blu M. Unser, W. Philips, I. Lemahieu and R. Van de Walle, "Hex-splines: a novel spline family for hexagonal lattices", *IEEE Transactions on Image Processing*, Volume: 13, Issue: 6, pp. 758 - 772, June 2004.

75. Y. Zhao, Q. Ke, F. Korn, J. Qi and R. Zhang, "A Framework for Native Hexagonal Convolutional Neural Networks", *in conference: 2020 IEEE International Conference on Data Mining (ICDM)*, Sorrento, Italy, 2021, DOI: 10.1109/ICDM50108.2020.00188.

76. Yaying Liu, Zhaojun Liu, and Kei May Lau, "Monolithic integrated all-GaN-based u-LED display by selective area regrowth", *Optics Express*, Vol. 31, No. 19, pp. 31300-3137, 2023. <https://doi.org/10.1364/OE.502275>.

77. E. F. Schubert and J. K. Kim, "Solid-State Light Sources Getting Smart", *Science*, vol. 308, pp.1274-1278, 2005.

78. Gun-Hee Lee et al., "Hexagonal Boron Nitride Passivation Layer for Improving the Performance and Reliability of InGaN/GaN Light-Emitting Diodes", *Applied Sciences*, vol. 11(19), pp. 1-7, October 2021.

79. X. Zhao, K. Sun, S. Cui, B. Tang, H. Hu and S. Zhou, “Recent Progress in Long-Wavelength InGaN Light-Emitting Diodes from the Perspective of Epitaxial Structure”,. *Advanced Photonics Researc*, Volume4, Issue9, pp.1-16, 2023.

80. Iphone 13 May Use Boe’s Oled Panel For The First Time [Електронний ресурс] – Режим доступу до ресурсу: <https://www.gizchina.com/2021/07/17/iphone-13-may-use-boes-oled-panel-for-the-first-time//> Дата звернення: Жовт. 12, 2023.

81. S. Zhang, t al., “Research Progress of Micro-LED Display Technology”, *Crystals* , volume 13(7), pp. 1-32, 2023, <https://doi.org/10.3390/cryst13071001>.

82. I. G. Juma, G. Kim, D. Jariwala and S. K. Behura, “Direct growth of hexagonal boron nitride on non-metallic substrates and its heterostructures with grapheme”, *iScience*, Volume 24, Issue 11, 2021.

83. S. Zhang, Y. Yan, T. Feng, Y. Yin, F. Ren, M. Liang and C. Wu “Wafer-Scale Semipolar Micro-Pyramid Lighting-Emitting Diode Array”, *Crystals* , 11(6), pp. 1-11, 2021, DOI:10.3390/cryst11060686.

84. “Future of Screens” [Електронний ресурс] – Режим доступу до ресурсу: <https://www.deloitte.com/global/en/Industries/tmt/perspectives/future-of-screens.html>.

85. H. Huang, Z. Ma, W. Chen, X. Cao, H. Fang and J. Yan “A sub-500 mV monolayer hexagonal boron nitride based memory device”, *Materials & Design*, Volume 198, pp. 1-8, 2021, <https://doi.org/10.1016/j.matdes.2020.109366>.

86. N. A. Sousa, “Hexagonal Grid Image Processing Algorithms in Neural Vision and Prediction”, *Faculdade de Engenharia da Universidade do Porto*, pp. 1-19, June 6, 2014.

87. А. М. Петух, Д. Т. Обідник та О. Н. Романюк, *Інтерполяція в задачах контурного формоутворення. Монографія*, Вінниця, Україна: ВНТУ, 2007.

88. S. Marschner and P. Shirley, *Fundamentals of computer graphics*, New York, USA, CRC press, 2018.

89. В. Г. Маценко, *Комп'ютерна графіка: Навчальний посібник*, Чернівці, Україна: Рута, 2009.
90. P. Manoharan, "Line Drawing Algorithm on an Interleaved Grid", *International Journal of Computer Applications*, Number 4, Article 1, 2011.
91. P. Manoharan and B. K. Ray, "An alternate line drawing algorithm on hexagonal grid", in *Proceedings of the 6th ACM India Computing Convention*, Tamil Nadu, Vellore, India, 2013, pp. 1-7.
92. Bresenham's line drawing algorithm on a hexagonal grid, Jan. 25, 2010. [Online]. Available: [http://zvold.blogspot.com/2010/01/bresenhams-line-drawing-algorithm-on\\_26.html](http://zvold.blogspot.com/2010/01/bresenhams-line-drawing-algorithm-on_26.html) Accessed on: October 28, 2023.
93. M. K. Kaleem, D. Verma and M. J. Idrisi, "Generalization of Line Drawing Algorithm – An effective approach to minimize the error in the existing Bresenham's Line Drawing Algorithm", in 2021 *International Conference on Emerging Smart Computing and Informatics*, Pune, India, 2021, pp. 1-15. DOI:10.1109/ESCI 50559.2021.9396940.
94. B. K. Ray, "An Improved Circle Drawing Algorithm on a Hexagonal Grid", *Journal of Graphics Tools*, Volume 17, pp. 5-15, 2013.
95. M. Pitteway, "Drawing conics on a hexagonal grid", in *Proceedings Fifth International Conference on Information Visualisation*, London, UK, July 2001, pp. 1-14. DOI: 10.1109/IV.2001.942101.
96. P. Manoharan and B. Kumar, "Semi-circular angle-based one bit circle generation algorithm on a hexagonal grid", *International Journal of Computer Aided Engineering and Technology*, volume 8 № 3, pp. 199-216, 2016.
97. M. Prabukumar and B. Kumar, "A mid-point ellipse drawing algorithm on a hexagonal grid", *International Journal of Computer Graphics* vol. 3.1, pp. 17-24, 2012.
98. A. Rababah, "Best approximate hyperbola drawing algorithm on hexagonal grid", *Journal of Inequalities and Special Functions*, Volume 7, Issue 3, pp. 1-11, 2016.



99. С. О. Романюк, О. В. Мельник, та І. Г. Бабій, “Антиаліайзинг дуги кола з використанням модифікованої оцінювальної функції”, на *II Всеукраїнській науково-технічній конференції студентів, аспірантів та молодих вчених “Інформаційні управляючі системи та комп’ютерний моніторинг”*, Донецьк, ДонНТУ, 2011, с. 37-41.

100. О. Н. Романюк, О. В. Мельник, С. О. Романюк, Т. І. Коробейнікова, та О. П. Прозор, “Антиаліайзинг зображення крокової траєкторії відрізка прямої на гексагональному растрі”, *Modern engineering and innovative technologies*, Issue 22, Part 1, с. 113-121, 2022. DOI: 10.30890/2567-5273.2022-22-01-043

101. О. Н. Романюк, Ю. О. Панфілова, та О. В. Мельник, “Використання гексагонів у комп’ютерних іграх” на *XLIX науково-технічній конференції підрозділів ВНТУ*, Вінниця, 27-28 квітня 2020 р. [Електронний ресурс]. Доступно:<https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/view/10431>. Дата звертання: 01.11.2023.

102. О. Н. Романюк, В. А. Шмалюх та О. В. Мельник, “Дисплейні технології формування гексагонального растру в сучасних пристроях відображення інформації”, in *Global Society in Formation of New Security System and World Order: Proceedings of the 2nd International Scientific and Practical Internet Conference*, Dnipro, Ukraine, July 27-28, 2023, с. 341-344.

103. О. Н. Романюк, М. Д. Захарчук, О. В. Мельник, О. В. Романюк та С. В. Котлик “Аналіз гексагональних ігор”, на *II Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів “Комп’ютерні ігри та мультимедіа як інноваційний підхід до комунікації”*, Одеса, 29-30 вересня 2022, с. 139-143.

104. О. Н. Романюк, О. В. Мельник та В. М. Чорний, “Комп’ютерна програма “Формування відрізків прямих з використанням для антиаліайзингу моделі пікселя у вигляді додекагона”, *Свідоцтво про реєстрацію авторського права на твір №57041* від 17.10.2014.

105. О. В. Мельник, “Комп’ютерна програма для формування відрізків прямих на гексагональному растрі”, *Свідоцтво про реєстрацію авторського права на твір* №74884 від 21.11.2017.

106. О. Н. Романюк, О. В. Романюк та О. В. Мельник, “Реалізація кругової інтерполяції при використанні гексагонального растру”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*, № 1(24), с. 53-58, 2017.

107. R. Lyons, S. Wang, R. Inkol and A. Joyal, “Another contender in the arctangent race”, *IEEE Signal Processing Magazine*. 2004. Vol. 20, No. 1. pp. 109-111.

108. Rajan S., Wang S., Inkol R., Joyal A. Efficient approximations for the arctangent. *IEEE Signal Processing Magazine*. 2006. Vol. 23, No. 3, P. 108-111.

109. О. Н. Романюк, О. В. Мельник, Р. Ю. Чехмestrucк та В. А. Шмалюх, “Програмний модуль для формування кіл на гексагональному растрі”, in *The 12th International scientific and practical conference “Modern research in world science”*, (2023) Lviv, Ukraine, 2023, pp. 326-332.

110. О. В. Мельник, “Формування кола незалежними оцінювальними функціями”, на *XXXI Міжнародній науково-практичній конференції MicroCAD-2023*, Харків, 2023, с. 1187-1188.

111. О. В. Мельник, “Стохастичний розподіл комбінованих крокових приростів при формуванні кіл на гексагональному растрі” на *Scientific Research and Innovation: Proceedings of the 2nd International Scientific and Practical Internet Conference*, Дніпро, 2023, с. 247-249.

112. О. Н. Романюк, М. С. Курінний, О. В. Мельник та Ю. Л. Ляшенко, “Комп’ютерна програма «Програма для антиаліазингу кривих другого порядку заданих рівнянням у загальній формі з використанням нового методу мультисемп-лінгу»”, *Свідоцтво про реєстрацію авторського права на твір* №35655 від 11.11.2010.

113. О. Н. Романюк, О. В. Мельник та О. О. Дудник, “Комп’ютерна програма для формування дуг кіл на гексагональному растрі”, *Свідоцтво про реєстрацію авторського права на твір №74925* від 22.11.2017.

114. Leif Kobbelt, Vision, modeling, and visualization 2006: proceedings, 2006, Aachen, Germany. Berlin: Akademische Verlagsgesellschaft, 412 p., 2006.

115. О. Н. Романюк, О. В. Романюк та О. В. Мельник, “Метод антиаліазингу зображень відрізків прямих з використанням додаткових оцінювальних функцій”, *Міжнародний науково-технічний журнал «Вимірювальна та обчислювальна техніка в технологічних процесах»*, Хмельницький, ХНУ, №2 (47), с. 210-214, 2014.

## **ДОДАТКИ**

## ДОДАТОК А

### Акти впровадження

ЗАТВЕРДЖУЮ

Директор ТОВ “ДРЕССЛАБ ЮЕЙ”

Роман Чехместрук

“3” серпня 2022р.



### АКТ

#### впровадження результатів кандидатської дисертаційної роботи Мельника Олександра Васильовича

Комісія у складі: директора Чехместрука Р.Ю., референта Чехместрук Л.Т. та інженера-програміста Перуна І.В. склали цей акт про те, що у ТОВ “Дресслаб Юей” впроваджено матеріали кандидатської дисертаційної роботи Мельника Олександра Васильовича.

У ТОВ “Дресслаб Юей” впроваджено методи та програмні засоби формування графічних примітивів на гексагональному растрі для формування текстурних карт, що використовуються при рендерингу 3Д-об'єктів.

Використання кандидатської дисертаційної роботи дозволило підвищити реалістичність відображення об'єктів при використанні 3Д технологій

В ТОВ “Дресслаб Юей” передано програмне забезпечення для формування графічних примітивів.

Члени комісії

Роман Чехместрук

Лариса Чехместрук

Іван Перун



Заступник директора

ТОВ «ЗД ДЖЕНЕРЕЙШН ЮЕЙ»

Швець О.В.

«1» березня 2023 р.

**АКТ**

**впровадження результатів дисертаційної роботи  
Мельника Олександра Васильовича**

Комісією у складі заступника директора Швеця О.В., технічного директора Чехместрука Р.Ю. та інженера-розробника Перуна І.В. склали цей акт про те, що у ТОВ «ЗД ДЖЕНЕРЕЙШН ЮЕЙ» впроваджено такі матеріали дисертаційної роботи Мельника Олександра Васильовича:

- методи формування графічних примітивів на гексагональному растрі для реалізації в комп'ютерних іграх;
- метод конвертації графічних зображень з прямокутного в гексагональний растр.

Використання результатів дисертаційної роботи дозволило підвищити стратегічні можливості комп'ютерних ігор за рахунок збільшення напрямків руху.

Мельником О.В. передано в ТОВ «ЗД ДЖЕНЕРЕЙШН ЮЕЙ» програмне забезпечення для формування зображень на гексагональному растрі.

Члени комісії:



Швець О.В.

Чехместрук Р.Ю.

Перун І.В.



Директор ПП «Фотоніка Плюс» Черкаси

Холін В.В

«24» КВІТНЯ 2023 р.

### АКТ

до впровадження результатів кандидатської дисертаційної роботи

**Мельника Олександра Васильовича**


Комісія у складі: заступника директора ПП «Фотоніка Плюс» Мулярової О.В., інженера ПП «Фотоніка Плюс» Петраша М.Т., інженера ПП «Фотоніка Плюс» Петраківського О.А. склали цей акт про те, що на базі ПП «Фотоніка Плюс» впроваджено практичні результати дисертаційної роботи Мельника О.В., а саме:

- Конвертор зображення з квадратного в гексагональний растр.
- Алгоритми формування графічних примітивів на гексагональному растрі в системах візуалізації та диференціальної діагностики злоякісних пухлин.
- Залежності між елементами гексагонального растру для побудови систем визначення границь пухлини та точок опромінення.

Мельником О.В. передано в ПП «Фотоніка Плюс» програмне забезпечення для формування зображень на гексагональному растрі.

  
\_\_\_\_\_ О.В. Мулярова

  
\_\_\_\_\_ М.Т. Петраш

  
\_\_\_\_\_ О.А. Петраківський

**ЗАТВЕРДЖУЮ:**

Проректор з науково-педагогічної роботи  
та організації освітнього процесу  
Вінницького національного  
технічного університету



Олександр ПЕТРОВ

«17» травня 2023р.

**АКТ**

**про впровадження результатів кандидатської дисертаційної роботи  
Мельника Олександра Васильовича  
у навчальний процес**

Комісія в складі голови - декана факультету інформаційних технологій та комп'ютерної інженерії (ФІТКІ) Вінницького національного технічного університету к.пед.н., доцента Кирилащук С.А. і членів: доцента кафедри програмного забезпечення к.т.н. доц. Бабюк Н.П. і завідувача кафедри захисту інформації д.т.н. проф. Лужецького В.А., склали цей акт про те, що у Вінницькому національному технічному університеті при вивченні дисциплін «Комп'ютерна графіка», «Графічні редактори», «Основи веб-дизайну», «Мультимедійні системи та комп'ютерний відеодизайн», а також при виконанні курсових і дипломних робіт у період 2020-2023 навчальних років впроваджено такі результати роботи Мельника О. В.:

- математична модель пікселя гексагонального растру;
- формування графічних примітивів на гексагональному растрі;
- графічний редактор для формування зображень на гексагональному растрі;
- конвертор зображення для гексагонального растру

Впровадження результатів дисертаційної роботи Мельника О. В. у навчальний процес дозволило покращити якість підготовки студентів Вінницького національного технічного університету.

Голова комісії:

Світлана КИРИЛАЩУК

Члени комісії:

Наталія БАБЮК

Володимир ЛУЖЕЦЬКИЙ



**ДОДАТОК Б****СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ**

[1] O. Melnik, O. Romanyuk, and V. Savratsky, *Applying of hexagonal raster in image formation. Monography*, Boston, USA, 2020.

[2] S. O. Romanyuk, S. V. Pavlov, and O. V. Melnyk, “New method to control color intensity for antialiasing”, *Control and Communications (SIBCON)*, 2015 International Siberian Conference on 02 July 2015, DOI: 10.1109/SIBCON.2015.7147194.

[3] О. Н. Романюк, О. В. Романюк, та О. В. Мельник, “Формування відрізків прямих на гексагональному растрі”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*, № 2, с. 69-72, 2016.

[4] О. Н. Романюк, та О. В. Мельник, “Особливості використання гексагонального растра при побудові пристроїв відображення”, *Міжнародний науково-технічний журнал «Вимірювальна та обчислювальна техніка в технологічних процесах»*, Хмельницький, №3 (56), с. 105-109, 2016.

[5] О. Н. Романюк, О. В. Романюк, та О. В. Мельник, “Реалізація кругової інтерполяції при використанні гексагонального растру”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*, № 1(24), с. 53-58, 2017.

[6] О. Н. Романюк, І. В. Абрамчук, та О. В. Мельник, “Визначення типів крокових приростів для побудови кола на гексагональному растрі”, *Вісник Хмельницького національного університету. Серія: Технічні науки*, Хмельницький, ХНУ, №3(249), с. 172-176, 2017.

[7] О. Н. Романюк, О. В. Мельник, І. В. Абрамчук, та Н. С. Костюкова, “Особливості формування еліпсів на гексагональному растрі”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*, № 1(26), с. 86-90, 2018.

[8] О. Н. Романюк, О. В. Мельник, та С. І. Вяткін, “Класифікація методів антиаліаїзінгу”, *Вісник Херсонського національного технічного університету*, Херсон, ХНТУ, №3(50), с. 154-159, 2014 р.

[9] O. N. Romanyuk, A. V. Melnik, A. P. Goncharuk, and Y. L. Lyashenko, “Effective Models for the Specular Color Constituent Computing”, *Journal of Computer Science and Engineering*, Volume 2, Issue 2, pp. 25-29, august 2010.

[10] O. Romanyuk, S. Pavlov, O. Melnyk, S. Romanyuk, A. Smolarz, and M. Bazarova, “Method of anti-aliasing with the use of the new pixel model”, *Optical Fibers and Their Applications*, Lublin, Poland, pp. 274-278, 2015. <https://doi.org/10.1117/12.2229013>.

[11] О. Н. Романюк, М. С. Курінний, О. В. Мельник, та Н. С. Костюкова, “Високопродуктивна конусна модель пікселя для антиаліаїзінгу відрізків прямих”, *Наукові праці Донецького національного технічного університету. Серія «Інформатика, кібернетика та обчислювальна техніка»*, Випуск 14(188), Донецьк, с. 216-220, 2011.

[12] О. Н. Романюк, М. С. Курінний, О. В. Мельник, та С. В. Олійник, “Кругова модель пікселя для задач антиаліаїзінгу”, *Наукові праці Донецького національного технічного університету. Серія «Інформатика, кібернетика та обчислювальна техніка»*, Випуск 15(203), Донецьк, с. 90-94, 2012.

[13] О. Н. Романюк, М. С. Курінний, та О. В. Мельник, “Антиаліаїзінг зображення відрізків прямих з використанням нової моделі пікселя”, *Оптико-електронні інформаційно-енергетичні технології. Міжнародний науково-технічний журнал. №2(20)*, с. 30-35, Вінниця, ВНТУ, 2010.

[14] О. Н. Романюк, О. В. Романюк, та О. В. Мельник, “Метод антиаліаїзінгу зображень відрізків прямих з використанням додаткових оцінювальних функцій”, *Міжнародний науково-технічний журнал «Вимірювальна та обчислювальна техніка в технологічних процесах»*, Хмельницький, ХНУ, №2 (47), с. 210-214, 2014.

[15] О. Н. Романюк, О. В. Мельник, І. В. Абрамчук, та О. О. Дудник, “Модифікація гаусівської моделі пікселя для задач антиаліайзингу”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*. № 1, с. 84-88, 2015.

[16] О. Н. Романюк, С. О. Романюк, О. В. Мельник, та Д. А. Озерчук, “Особливості формування еліпсів, повернутих на заданий кут, на гексагональному растрі”, *Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка»*, № 2(31), с. 23-28, 2020.

[17] О. Н. Романюк, О. В. Мельник, С. В. Котлик, С. О. Романюк, та Р. Ю. Чехместрук, “Методи підвищення продуктивності формування векторів на гексагональному растрі”, *Автоматизація технологічних і бізнес-процесів*, Volume 14, Issue 3, с. 27-36, 2022. DOI: 10.15673/atbp.v14i3.2350

[18] О. Н. Романюк, О. В. Мельник, та В. А. Шмалюх, “Метод прискореної кругової інтерполяції на гексагональному растрі”, *Вісник ВПІ*, № 2, с. 81–88, 2023. doi.org/10.31649/1997-9266-2023-167-2-81-88

[19] О. Н. Романюк, О. В. Мельник, С. О. Романюк, Т. І. Коробейнікова, та О. П. Прозор, “Антиаліайзинг зображення крокової траєкторії відрізка прямої на гексагональному растрі”, *Modern engineering and innovative technologies*, Issue 22, Part 1, с. 113-121, 2022. DOI: 10.30890/2567-5273.2022-22-01-043

[20] С. О. Романюк, О. Н. Романюк, С. В. Павлов, та О. В. Мельник, “Модель для відтворення спекулярної складової кольору в засобах комп’ютерної графіки”, *Інформаційні технології та комп’ютерна інженерія*, ВНТУ, №3(34), с. 50-57, 2015.

[21] О. Н. Романюк, Ю. О. Панфілова, та О. В. Мельник, “Використання гексагонів у комп’ютерних іграх” на *XLIX науково-технічній конференції підрозділів ВНТУ*, Вінниця, 27-28 квітня 2020 р. [Електронний ресурс]. Доступно:<https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/view/10431>. Дата звертання: 01.11.2023.

[22] С. О. Романюк, О. В. Мельник, та І. Г. Бабій, “Антиаліайзинг дуги кола з використанням модифікованої оцінювальної функції”, на *II Всеукраїнській науково-технічній конференції студентів, аспірантів та молодих вчених “Інформаційні управляючі системи та комп’ютерний моніторинг”*, Донецьк, ДонНТУ, 2011, с. 37-41.

[23] О. Н. Романюк, та О. В. Мельник, “Морфологічний антиаліайзинг для гексагонального растру”, на *III Міжнародній науково-практичній конференції «Актуальні проблеми гуманітарних та природничих наук»*, Київ, 28-29 жовтня 2016, с. 112-115.

[24] О. Н. Романюк, О. В. Мельник, та О. В. Стукач, “Моделювання гексагонального растра на квадратному растрі”, на *VII Міжнародній науково-технічній конференції «Моделювання і комп’ютерна графіка»*, Київ, 18-24 вересня 2017, с.18-24.

[25] О. Н. Романюк, О. В. Мельник, та Ю. О. Панфілова, “Використання гексагонального растру в комп’ютерних іграх”, на *XII Міжнародній науково-технічній конференції «Інформаційні комп’ютерні технології – 2021 (ІКТ-2021)»*, Житомир: Житомирська політехніка, 1-3 квітня 2021, с. 5-6.

[26] О. Н. Романюк, О. В. Мельник, та Ю. О. Панфілова, “Формування параболи на гексагональному растрі”, на *Міжнародній науково-практичній Інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ»*, Суми-Вінниця, 9-10 листопада 2021, с. 196-198.

[27] О. Н. Романюк, О. В. Романюк, О. В. Мельник, та Р. Ю. Чехмestрук, “Суперсемплінг зображень, сформованих на гексагональному растрі”, in *Modern research in world science Proceedings of the 3rd International scientific and practical conference. SPC “Sci-conf.com.ua”*, Lviv, Ukraine, 2022, pp. 517-522.

[28] О. Н. Романюк, О. В. Мельник, та Л. Г. Коваль, “Використання гексагональних комірок у видавничій справі”, на *V Міжнародній науковій*

конференції «Інформація, комунікація та управління знаннями в глобалізованому світі», Київ, 23-24 червня 2022, с. 45-47.

[29] О. Н. Романюк, О. В. Мельник, та В. А. Шмалюх, “Розробка застосунку для формування відрізків на гексагональному растрі”, на *Всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи»*, Вінниця, 2022, с 1-7.

[30] О. Н. Романюк, М. Д. Захарчук, О. В. Мельник, О. В. Романюк, та С. В. Котлик “Аналіз гексагональних ігор”, на *II Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів “Комп’ютерні ігри та мультимедіа як інноваційний підхід до комунікації”*, Одеса, 29-30 вересня 2022, с. 139-143.

[31] О. Н. Романюк, М. В. Козубенко, О. В. Мельник, та С. В. Котлик, “Використання гексагонального растру в картографії», на *XV Міжнародній науково-практичній конференції Інформаційні технології і автоматизація – 2022*, Одеса, 20-21 жовтня 2022, с. 30-33.

[32] О. Н. Романюк, О. В. Романюк, та О. В. Мельник, “Програмна реалізація графічного редактора на гексагональному растрі”, in *The 8th International scientific and practical conference “Modern research in world science”*, (October 29-31, 2022) SPC “Sci-conf.com.ua”, Lviv, Ukraine, 2022, pp. 389-392.

[33] О. Н. Романюк, О. В. Мельник, Р. Ю. Чехместрук, та В. А. Шмалюх, “Програмний модуль для формування кіл на гексагональному растрі”, in *The 12th International scientific and practical conference “Modern research in world science”*, (February 26-28, 2023) SPC “Sci-conf.com.ua”, Lviv, Ukraine, 2023, pp. 326-332.

[34] О. В. Мельник, “Формування кола незалежними оцінювальними функціями”, на *XXXI Міжнародній науково-практичній конференції MicroCAD-2023*, Харків, 2023, с. 1187-1188.

[35] О. В. Мельник, “Стохастичний розподіл комбінованих крокових приростів при формуванні кіл на гексагональному растрі” на *Scientific*

*Research and Innovation: Proceedings of the 2nd International Scientific and Practical Internet Conference*, Дніпро, 2023, с. 247-249.

[36] О. Н. Романюк, О. В. Мельник, Р. Ю. Чехмestruc та С. О. Романюк, “Основні співвідношення гексагонального растру”, на *VII Міжнародній науково-практичній конференції Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі*, Київ, 2022, с. 59-61.

[37] О. Н. Романюк, В. А. Шмалюх, та О. В. Мельник, “Дисплейні технології формування гексагонального растру в сучасних пристроях відображення інформації”, in *Global Society in Formation of New Security System and World Order: Proceedings of the 2nd International Scientific and Practical Internet Conference*, July 27-28, 2023, Dnipro, Ukraine, с. 341-344.

[38] О. Н. Романюк, М. С. Курінний, О. В. Мельник та Ю. Л. Ляшенко, “Комп’ютерна програма «Програма для антиаліайзингу кривих другого порядку заданих рівнянням у загальній формі з використанням нового методу мультисемплінгу”, *Свідоцтво про реєстрацію авторського права на твір* №35655 від 11.11.2010.

[39] О. Н. Романюк, О. В. Мельник та В. М. Чорний, “Комп’ютерна програма “Формування відрізків прямих з використанням для антиаліайзингу моделі пікселя у вигляді додекагона”, *Свідоцтво про реєстрацію авторського права на твір* №57041 від 17.10.2014.

[40] О. В. Мельник, “Комп’ютерна програма для формування відрізків прямих на гексагональному растрі”, *Свідоцтво про реєстрацію авторського права на твір* №74884 від 21.11.2017.

[41] О. Н. Романюк, О. В. Мельник та О. О. Дудник, “Комп’ютерна програма для формування дуг кіл на гексагональному растрі”, *Свідоцтво про реєстрацію авторського права на твір* №74925 від 22.11.2017.

## ДОДАТОК В

### ЛІСТИНГИ ПРОГРАМ

#### Лістинг програми графічного редактора на гексагональному растрі

##### Файл painter.js

```

const artwork = {
  version: { major: 2, minor: 0 },
  effectiveRect: { left: 288, top: 166, width: 1087, height: 625 },
  canvasSize: { width: 2000, height: 2000 },
  backgroundColor: "#ffffff",c
const exchangeToken = "";
const exchangeUrl = "https://gp-exchange.avhost.info/socket.io";

var mode = "paint"; //available values "paint", "pick-color", "copy", "paste"
var paper;
var grid;
var selectedShapeName = null;
var selectedColor = "#3377ee";
var backgroundColor = "#ffffff";
var recentColors = [];
var changed = false;
var paperMouseDown = false;

var shapesToolbarGrid;

var gridArtwork = new GridArtwork();
var workspaceWidth;
var workspaceHeight;

var undoStack = [];
var redoStack = [];

var selection = new GridSelection();
a;

var drawToolProperties = {};
var drawToolTemporaryCells = [];
var drawToolTemporaryShapes = {};

var modalDescriptionVisible = false;
var modalTagsVisible = false;
var modalSquareGridSpecialPropertiesVisible = false;
var initialTagsValue = null;

var socket = null;
var collaboratorsOnline = [];
var collaboratorBadges = [];

//
function adjustCanvasWrapper() {
  $("#canvas-wrapper").height($(window).height() - 60);
  $(".painter-toolbar-full").height($(window).height() - 60);
}

function paintShapeToolButton(shapeName) {

```

```

var shape = shapesToolbarGrid.shapes[shapeName];

$("#shape-" + shapeName).html("");

var shapeRect = shapesToolbarGrid.getCellRect(0, 0);
var shapePaper = new Raphael(
    "shape-" + shapeName,
    shapeRect.width + 20,
    shapeRect.height + 20
);

if (shapeName == selectedShapeName) {
    var bgElement = shapePaper.ellipse(
        shapeRect.width / 2 + 10,
        shapeRect.height / 2 + 10,
        shapeRect.width / 2 + 10,
        shapeRect.height / 2 + 10
    );
    bgElement.attr({ fill: "r#ffffff:40-#cccccc", "stroke-width": 0 });
}
shape.paint(shapePaper, 0, 0, selectedColor, 10, 10);
}

function selectShape(shapeName) {
    var oldSelectedShapeName = selectedShapeName;
    selectedShapeName = shapeName;
    for (var shapeName in grid.shapes) {
        if (shapeName == selectedShapeName || shapeName == oldSelectedShapeName) {
            paintShapeToolButton(shapeName);
        }
    }
}

function paintOnCanvas(col, row, shapeName, color) {
    var cell = gridArtwork.setCell(col, row, shapeName, color);
    if (!cell.element && cell.shapeName != "empty") {
        var element = grid.shapes[shapeName].paint(paper, col, row, color, 0, 0);
        cell.element = element;
    }
}

/**
 * Return cell coordintes {row: XXX, col: XXX} by mouse event
 */
function getCellCoordByMouseEvent(event) {
    return grid.pointToCell(
        event.pageX - $("#canvas").position().left,
        event.pageY - $("#canvas").position().top
    );
}

function updateCellCoordiantesPanel(event) {
    var cell = getCellCoordByMouseEvent(event);
    $("#coordinates").text(cell.col + " " + cell.row);
}

function storeUndoCell(col, row, newShapeName, newColor) {

```



```

var oldCell = gridArtwork.getCell(col, row);
var oldShapeName = "empty";
var oldColor = "#ffffff";
if (oldCell) {
  oldShapeName = oldCell.shapeName;
  oldColor = oldCell.color;
}

undoStack[undoStack.length - 1].pushCellChange(
  col,
  row,
  oldShapeName,
  newShapeName,
  oldColor,
  newColor
);
}

function paintOnCanvasByMouseEvent(event) {
  var cell = getCellCoordByMouseEvent(event);

  if (paperMouseDown) {
    newShapeName = selectedShapeName;
    if (event.which === 3) {
      newShapeName = "empty";
    }

    // Return immediately if there are no actual changes
    var oldCell = gridArtwork.getCell(cell.col, cell.row);
    if (
      (oldCell === null || oldCell.shapeName === "empty") &&
      newShapeName === "empty"
    ) {
      return;
    }
    if (
      oldCell !== null &&
      oldCell.shapeName === newShapeName &&
      oldCell.color === selectedColor
    ) {
      return;
    }

    storeUndoCell(cell.col, cell.row, newShapeName, selectedColor);
    paintOnCanvas(cell.col, cell.row, newShapeName, selectedColor);
    if (newShapeName !== "empty") {
      pushRecentColor(selectedColor);
    }
    changed = true;

    if (socket !== null) {
      var changes = {
        cells: [
          {
            col: cell.col,
            row: cell.row,
            shapeName: newShapeName,

```

```

        color: selectedColor,
    },
],
};
console.log("socket.io <- changes (paintOnCanvasByMouseEvent)");
console.log(changes);
socket.emit("changes", changes);
}
}
}

```

```

function eraseOnCanvasByMouseEvent(event) {
var cell = getCellCoordByMouseEvent(event);
if (paperMouseDown) {
    // Return immediately if there are no actual changes
    var oldCell = gridArtwork.getCell(cell.col, cell.row);
    if (oldCell == null || oldCell.shapeName == "empty") {
        return;
    }

    storeUndoCell(cell.col, cell.row, "empty", selectedColor);
    paintOnCanvas(cell.col, cell.row, "empty", selectedColor);
    changed = true;

    if (socket != null) {
        var changes = {
            cells: [
                {
                    col: cell.col,
                    row: cell.row,
                    shapeName: "empty",
                    color: selectedColor,
                },
            ],
        };
        console.log("socket.io <- changes (eraseOnCanvasByMouseEvent)");
        console.log(changes);
        socket.emit("changes", changes);
    }
}
}
}

```

```

function selectOnCanvasByMouseEvent(event) {
var cell = getCellCoordByMouseEvent(event);

if (paperMouseDown) {
    var oldCell = gridArtwork.getCell(cell.col, cell.row);

    var cellShapeName = "empty";
    var cellColor = "#ffffff";
    if (oldCell) {
        cellShapeName = oldCell.shapeName;
        cellColor = oldCell.color;
    }

    if (event.which == 3) {
        selection.forgetCell(cell.col, cell.row);
    }
}
}

```

```

    } else {
        selection.saveCell(cell.col, cell.row, cellShapeName, cellColor);
    }
}
}

function changePastePositionByMouseEvent(event) {
    var cell = getCellCoordByMouseEvent(event);
    var nearestCell = grid.nearestSameCell(
        selection.baseCol,
        selection.baseRow,
        cell.col,
        cell.row
    );
    selection.changePasteCell(nearestCell.col, nearestCell.row);
}

function pickColorByMouseEvent(event) {
    var cell = getCellCoordByMouseEvent(event);
    var cellItem = gridArtwork.getCell(cell.col, cell.row);
    if (cellItem) {
        selectColor(cellItem.color);
        $("#btn-pick-color").button("toggle");
        setMode("paint");

        if (selectedShapeName == "empty") {
            selectShape("flat");
        }
    }
}

function draftDrawToolOnCanvasByMouseEvent(event) {
    var cell = getCellCoordByMouseEvent(event);

    var toolName = mode;

    if (paperMouseDown) {
        if (!drawToolProperties["startCell"]) {
            drawToolProperties["startCell"] = cell;
            drawToolProperties["endCell"] = cell;
        } else {
            var startCell = drawToolProperties["startCell"];
            var endCell = drawToolProperties["endCell"];
            if (event.ctrlKey) {
                cell = grid.drawTools[toolName].adjustEndCell(startCell, cell);
            }

            if (startCell.col == cell.col && startCell.row == cell.row) {
                // Line with zero length
                for (var i = 0; i < drawToolTemporaryCells.length; i++) {
                    var key = cellKey(drawToolTemporaryCells[i]);
                    drawToolTemporaryShapes[key].remove();
                    delete drawToolTemporaryShapes[key];
                }
                drawToolTemporaryCells = [];
                drawToolTemporaryShapes = {};
                endCell = cell;
            }
        }
    }
}

```

```

    return;
}

if (endCell.col == cell.col && endCell.row == cell.row) {
    return;
}
drawToolProperties["endCell"] = cell;
endCell = cell;
var newTemporaryCells = grid.drawTools[toolName].calculateCells(
    startCell,
    endCell
);

for (var i = 0; i < drawToolTemporaryCells.length; i++) {
    var found = false;
    var oldKey = cellKey(drawToolTemporaryCells[i]);
    for (var j = 0; j < newTemporaryCells.length; j++) {
        var newKey = cellKey(newTemporaryCells[j]);
        if (oldKey == newKey) {
            found = true;
            break;
        }
    }
    if (!found && drawToolTemporaryShapes[oldKey]) {
        drawToolTemporaryShapes[oldKey].remove();
        delete drawToolTemporaryShapes[oldKey];
    }
}

for (var i = 0; i < newTemporaryCells.length; i++) {
    var newKey = cellKey(newTemporaryCells[i]);
    var found = false;
    for (var j = 0; j < drawToolTemporaryCells.length; j++) {
        var oldKey = cellKey(drawToolTemporaryCells[j]);
        if (newKey == oldKey) {
            found = true;
            break;
        }
    }
    if (!found) {
        var cell = newTemporaryCells[i];
        var element = grid.internalShapes["selected"].paint(
            paper,
            cell.col,
            cell.row,
            "#ffffff",
            0,
            0
        );
        drawToolTemporaryShapes[newKey] = element;
    }
}

drawToolTemporaryCells = newTemporaryCells;
}
}
}

```

```

function drawToolOnCanvas() {
  if (drawToolProperties["startCell"] && drawToolProperties["endCell"]) {
    var startCell = drawToolProperties["startCell"];
    var endCell = drawToolProperties["endCell"];
    var cells = drawToolTemporaryCells;
    for (var i = 0; i < cells.length; i++) {
      storeUndoCell(
        cells[i].col,
        cells[i].row,
        selectedShapeName,
        selectedColor
      );
      paintOnCanvas(
        cells[i].col,
        cells[i].row,
        selectedShapeName,
        selectedColor
      );
    }

    for (var i = 0; i < cells.length; i++) {
      var key = cellKey(cells[i]);
      drawToolTemporaryShapes[key].remove();
    }
    drawToolProperties = {};
    drawToolTemporaryCells = [];
    drawToolTemporaryShapes = {};

    if (selectedShapeName != "empty") {
      pushRecentColor(selectedColor);
    }

    changed = true;

    if (socket != null) {
      var changes = {
        cells: [],
      };
      for (var i = 0; i < cells.length; i++) {
        changes.cells.push({
          col: cells[i].col,
          row: cells[i].row,
          shapeName: selectedShapeName,
          color: selectedColor,
        });
      }

      console.log("socket.io <- changes (drawToolOnCanvas)");
      console.log(changes);
      socket.emit("changes", changes);
    }
  }
}

function getArtworkEffectiveRect(grid, artwork) {
  var x1 = 100000;

```

```

var y1 = 100000;
var x2 = 0;
var y2 = 0;
for (var row = 0; row < artwork.cells.length; row++) {
  for (var col = 0; col < artwork.cells[row].length; col++) {
    if (
      artwork.cells[row][col] &&
      artwork.cells[row][col].shapeName != "empty"
    ) {
      var cellRect = grid.getCellRect(col, row);
      if (cellRect.left < x1) {
        x1 = cellRect.left;
      }
      if (cellRect.top < y1) {
        y1 = cellRect.top;
      }
      if (cellRect.left + cellRect.width > x2) {
        x2 = cellRect.left + cellRect.width;
      }
      if (cellRect.top + cellRect.height > y2) {
        y2 = cellRect.top + cellRect.height;
      }
    }
  }
}

return {
  left: Math.floor(x1),
  top: Math.floor(y1),
  width: Math.round(x2 - x1 + 1),
  height: Math.round(y2 - y1 + 1),
};
}

```

```

function getArtworkEffectivePixelArtRect(grid, artwork) {
  var x1 = 100000;
  var y1 = 100000;
  var x2 = 0;
  var y2 = 0;
  for (var row = 0; row < artwork.cells.length; row++) {
    for (var col = 0; col < artwork.cells[row].length; col++) {
      if (
        artwork.cells[row][col] &&
        artwork.cells[row][col].shapeName != "empty"
      ) {
        if (row < y1) {
          y1 = row;
        }

        if (row > y2) {
          y2 = row;
        }

        if (col < x1) {
          x1 = col;
        }
      }
    }
  }
}

```

```

        if (col > x2) {
            x2 = col;
        }
    }
}

return {
    left: x1,
    top: y1,
    width: x2 - x1 + 1,
    height: y2 - y1 + 1,
};
}

function riseUpButton(selector) {
    if ($(selector).attr("aria-pressed") == "true") {
        $(selector).button("toggle");
    }
}

function setMode(m) {
    if (mode == "copy") {
        selection.copyFinished();
    }

    if (mode == "paste") {
        selection.pasteFinished();
    }

    $("#btn-pencil").removeClass("active").removeAttr("disabled");
    $("#btn-pick-color").removeClass("active").removeAttr("disabled");
    $("#btn-flood-fill").removeClass("active").removeAttr("disabled");
    $("#btn-draw-line").removeClass("active").removeAttr("disabled");
    $("#btn-copy-mode").removeClass("active").removeAttr("disabled");
    $("#btn-paste-mode").removeClass("active").removeAttr("disabled");
    $("#btn-erase").removeClass("active").removeAttr("disabled");

    if (grid.drawTools) {
        for (var toolName in grid.drawTools) {
            $(".btn-draw-tool[tool-name=" + toolName + "]")
                .removeClass("active")
                .removeAttr("disabled");
        }
    }

    mode = m;
    if (mode == "paint") {
        $("#canvas-wrapper").css("cursor", "crosshair");
        $("#btn-pencil").addClass("active");
    } else if (mode == "pick-color") {
        $("#canvas-wrapper").css(
            "cursor",
            "url(/img/cursors/pick-color.png) 2 32, crosshair"
        );
        $("#btn-pick-color").addClass("active");
    } else if (mode == "copy") {

```

```

$("#canvas-wrapper").css(
  "cursor",
  "url(/img/cursors/mark-area.png) 8 8, crosshair"
);
$("#btn-copy-mode").addClass("active");
selection.copyPrepare();
} else if (mode == "paste") {
  $("#canvas-wrapper").css("cursor", "crosshair"); // TODO change cursor
  $("#btn-paste-mode").addClass("active");
  selection.pastePrepare();
} else if (mode == "fill") {
  $("#canvas-wrapper").css(
    "cursor",
    "url(/img/cursors/flood-fill.png) 2 32, crosshair"
  );
  $("#btn-flood-fill").addClass("active");
} else if (mode == "erase") {
  $("#canvas-wrapper").css(
    "cursor",
    "url(/img/cursors/eraser.png) 8 32, crosshair"
  );
  $("#btn-erase").addClass("active");
} else if (grid.drawTools[mode]) {
  $("#canvas-wrapper").css("cursor", "crosshair");
  $(".btn-draw-tool[tool-name=" + mode + "]).addClass("active");
}
}

function prepareArtworkToSave() {
  var a = {
    version: {
      major: 2,
      minor: 0,
    },
    effectiveRect: getArtworkEffectiveRect(grid, gridArtwork),
    canvasSize: {
      width: grid.workspaceWidth,
      height: grid.workspaceHeight,
    },
    backgroundColor: backgroundColor,
    layers: [
      {
        grid: grid.name,
        cellSize: grid.cellSize,
        gridThickness: grid.gridThickness,
        gridColor: grid.gridColor,
        rows: [],
      },
    ],
    recentColors: recentColors,
  };

  if (grid.name == "square") {
    a["effectivePixelArtRect"] = getArtworkEffectivePixelArtRect(
      grid,
      gridArtwork
    );
  }
}

```



```

}

a["transparentBackground"] = $("#modal_transparent_background")[0].checked;
a["gridVisible"] = $("#modal_artwork_grid_visible")[0].checked;
a["additionalPixelImage"] = $("#modal_artwork_pixel_art")[0].checked;

if (artwork.id) {
  a.id = artwork.id;
}

for (var row = 0; row < gridArtwork.cells.length; row++) {
  var rowElement = {
    row: row,
  };
  var cellArray = [];

  for (var col = 0; col < gridArtwork.cells[row].length; col++) {
    if (
      gridArtwork.cells[row][col] &&
      gridArtwork.cells[row][col].shapeName != "empty"
    ) {
      cellArray.push([
        col,
        gridArtwork.cells[row][col].shapeName,
        gridArtwork.cells[row][col].color,
      ]);
    }
  }

  if (cellArray.length > 0) {
    rowElement.cells = cellArray;
    a.layers[0].rows.push(rowElement);
  }
}

return a;
}

function saveArtwork() {
  var a = prepareArtworkToSave();

  if (a.effectiveRect.width < 0 || a.effectiveRect.height < 0) {
    var messageModal = $("#message-modal");
    messageModal.find("#message-text").text("Cannot save empty image");
    messageModal.modal();
    return;
  }

  $("#input[name=artwork_json]").val(JSON.stringify(a));
  changed = false;
  showCircleLoader();
  $.ajax({
    type: "post",
    url: "/json/save-image",
    data: $("form[name=f]").serialize(),
    dataType: "json",
    success: function (data) {

```

```

if (data.error) {
  hideCircleLoader();
  if (data.error === "few_pixels") {
    showWarningMessage(
      "The image contains too few pixels. Please, create more complex image."
    );
  } else {
    showWarningMessage("Something went wrong. Try again.");
  }
  return;
}
if (data.result) {
  var artwork_id = data.result;
  var artwork_tags = $("input[name=artwork_tags]").val();
  if (initialTagsValue === artwork_tags) {
    document.location = "/images/details/" + artwork_id;
  } else {
    $.ajax({
      type: "post",
      url: "/json/save-image-tags",
      data: {
        artwork_id: artwork_id,
        artwork_tags: artwork_tags,
      },
      dataType: "json",
      success: function (data) {
        document.location = "/images/details/" + artwork_id;
      },
      error: function () {
        hideCircleLoader();
        showWarningMessage("Something went wrong. Try again.");
      },
    });
  }
} else {
  hideCircleLoader();
}
},
error: function () {
  hideCircleLoader();
  showWarningMessage("Something went wrong. Try again.");
},
});
}

function propertiesDialog_updateDescriptionLabelPreview() {
  var description = $("#modal_artwork_description").val();
  $("#modal_description_label")
    .find(".save-dialog-expandable-content-preview")
    .text(description);
}

function propertiesDialog_updateTagsLabelPreview() {
  var tags = $("#modal_artwork_tags").val();
  var text = "";
  if (tags) {
    var tagsList = tags.split(",");
  }
}

```

```

for (var i = 0; i < tagsList.length; i++) {
  if (i > 0) {
    text += " ";
  }
  text += "#" + tagsList[i];
}
}
$("#modal_tags_label")
  .find(".save-dialog-expandable-content-preview")
  .text(text);
}

function showPropertiesDialog(modalAction) {
  if (artwork.layers[0].grid == "square") {
    $("#modal_square_grid_special_properties").show();
  } else {
    $("#modal_square_grid_special_properties").hide();
  }
}

propertiesDialog_updateDescriptionLabelPreview();
propertiesDialog_updateTagsLabelPreview();

$("#modal_success_action").val(modalAction);
$("#properties-modal").modal("show");
}

/*
 * Push color of the last painted shape to palette
 */
function pushRecentColor(hexColor) {
  pushColor = hexColor;
  for (var i = 0; i < recentColors.length; i++) {
    var oldColor = recentColors[i];
    recentColors[i] = pushColor;
    pushColor = oldColor;
    if (oldColor == hexColor) {
      break;
    }
  }
}

$paletteDivs = $("#color-palette div");
for (var i = 0; i < $paletteDivs.length && i < recentColors.length; i++) {
  $($paletteDivs[i]).css("background-color", recentColors[i]);
}
}

function selectColorFromPicker(hexColor) {
  if (selectedColor.toUpperCase() == hexColor.toUpperCase()) {
    return;
  }
}

selectedColor = hexColor;
for (var shapeName in grid.shapes) {
  paintShapeToolButton(shapeName);
  $("#selected-color").css("background-color", selectedColor);
}
}

```

```

var colorInText = $("#color-picker-text").val();
if (colorInText.toUpperCase() !== hexColor.toUpperCase()) {
  $("#color-picker-text").val(hexColor);
}
}

function calculateFillArea(cell) {
  var gridCell = gridArtwork.getCell(cell.col, cell.row);
  var sourceColorShape = "empty";
  if (gridCell) {
    if (gridCell.shapeName !== "empty") {
      sourceColorShape = gridCell.shapeName + "-" + gridCell.color;
    }
  }
}

var colCount = workspaceWidth / grid.cellSize;
var rowCount = workspaceHeight / grid.cellSize;

var currentCells = [
  {
    row: cell.row,
    col: cell.col,
  },
];
var result = [
  {
    row: cell.row,
    col: cell.col,
  },
];

var testFill = function (col, row) {
  var testCell = gridArtwork.getCell(col, row);
  var testColorShape = "empty";
  if (testCell) {
    if (testCell.shapeName !== "empty") {
      testColorShape = testCell.shapeName + "-" + testCell.color;
    }
  }
  if (testColorShape !== sourceColorShape) {
    return false;
  }
  for (var i = 0; i < result.length; i++) {
    if (result[i].col === col && result[i].row === row) {
      return false;
    }
  }
  return true;
};

while (currentCells.length > 0) {
  var newCells = [];
  for (var i = 0; i < currentCells.length; i++) {
    var currentCell = currentCells[i];
    var adjacentCells = grid.getAdjacentCells(
      currentCell.col,
      currentCell.row

```

```

);
for (var j = 0; j < adjacentCells.length; j++) {
  var testCell = adjacentCells[j];
  if (!grid.isCellInsideWorkspace(testCell.col, testCell.row)) {
    return [];
  }

  if (testFill(testCell.col, testCell.row)) {
    newCells.push(testCell);
    result.push(testCell);
  }
}
currentCells = newCells;
}

return result;
}

function fillAreaOnCanvasByMouseEvent(event) {
  var cell = getCellCoordByMouseEvent(event);

  if (paperMouseDown) {
    paperMouseDown = false;
    newShapeName = selectedShapeName;
    if (event.which == 3) {
      newShapeName = "empty";
    }

    var fillCells = calculateFillArea(cell);
    if (fillCells.length == 0) {
      showWarningMessage(
        'You cannot fill entire workspace with flood fill tool. It is applicable for closed areas only. Use "Set
background color" instead.'
      );
      return;
    }

    for (var i = 0; i < fillCells.length; i++) {
      var currentCell = fillCells[i];
      storeUndoCell(
        currentCell.col,
        currentCell.row,
        newShapeName,
        selectedColor
      );
      paintOnCanvas(
        currentCell.col,
        currentCell.row,
        newShapeName,
        selectedColor
      );
    }

    if (newShapeName != "empty") {
      pushRecentColor(selectedColor);
    }
  }
}

```

```

changed = true;

if (socket != null) {
  var changes = {
    cells: [],
  };
  for (var i = 0; i < fillCells.length; i++) {
    changes.cells.push({
      col: fillCells[i].col,
      row: fillCells[i].row,
      shapeName: newShapeName,
      color: selectedColor,
    });
  }

  console.log("socket.io <- changes (fillAreaOnCanvasByMouseEvent)");
  console.log(changes);
  socket.emit("changes", changes);
}
}
}

function selectColorFromText(hexColor) {
  if (selectedColor.toUpperCase() == hexColor.toUpperCase()) {
    return;
  }

  var testColor = /^[0-9A-Fa-f]{6}$/i.test(hexColor);

  if (hexColor.length != 7 || !testColor) {
    return;
  }

  if (hexColor.toUpperCase() == selectedColor.toUpperCase()) {
    return;
  }

  selectedColor = hexColor;

  for (var shapeName in grid.shapes) {
    paintShapeToolButton(shapeName);
    $("#selected-color").css("background-color", selectedColor);
  }

  $("#color-picker").minicolors("value", hexColor);
}

function selectColor(hexColor) {
  if (selectedColor.toUpperCase() == hexColor.toUpperCase()) {
    return;
  }

  selectedColor = hexColor;
  for (var shapeName in grid.shapes) {
    paintShapeToolButton(shapeName);
    $("#selected-color").css("background-color", selectedColor);
  }
}

```

```

}

var colorInText = $("#color-picker-text").val();
if (colorInText.toUpperCase() != hexColor.toUpperCase()) {
    $("#color-picker-text").val(hexColor);
}

$("#color-picker").minicolors("value", hexColor);
}

function updateUndoRedoButtons() {
    if (undoStack.length > 0) {
        $("#btn-undo").removeAttr("disabled");
    } else {
        $("#btn-undo").attr("disabled", "disabled");
    }

    if (redoStack.length > 0) {
        $("#btn-redo").removeAttr("disabled");
    } else {
        $("#btn-redo").attr("disabled", "disabled");
    }
}

function doUndo() {
    if (undoStack.length > 0) {
        var undoStep = undoStack.pop();
        var changes = {
            cells: [],
        };
        if (undoStep.shiftChange) {
            var dir = undoStep.shiftChange.dir;
            if (dir == "left") {
                gridArtwork.doShiftRight(grid);
                changes.shift = "right";
            } else if (dir == "right") {
                gridArtwork.doShiftLeft(grid);
                changes.shift = "left";
            } else if (dir == "up") {
                gridArtwork.doShiftDown(grid);
                changes.shift = "down";
            } else if (dir == "down") {
                gridArtwork.doShiftUp(grid);
                changes.shift = "up";
            }
        }
        var removedCells = undoStep.shiftChange.removedCells;
        for (var i = 0; i < removedCells.length; i++) {
            paintOnCanvas(
                removedCells[i].col,
                removedCells[i].row,
                removedCells[i].shapeName,
                removedCells[i].color
            );
            changes.cells.push({
                col: removedCells[i].col,
                row: removedCells[i].row,
                shapeName: removedCells[i].shapeName,
            });
        }
    }
}

```

```

        color: removedCells[i].color,
    });
}
} else if (undoStep.backgroundChange) {
    setBackgroundColor(undoStep.backgroundChange.oldColor);
    changes.backgroundColor = undoStep.backgroundChange.oldColor;
} else if (undoStep.workspaceChange) {
    applyWorkspaceSize(
        undoStep.workspaceChange.oldWorkspace.width,
        undoStep.workspaceChange.oldWorkspace.height,
        undoStep.workspaceChange.oldWorkspace.cellSize,
        undoStep.workspaceChange.oldWorkspace.gridThickness,
        undoStep.workspaceChange.oldWorkspace.gridColor,
        undoStep.workspaceChange.oldWorkspace.backgroundColor
    );
    changes.workspace = undoStep.workspaceChange.oldWorkspace;
} else {
    for (var i = 0; i < undoStep.cellChanges.length; i++) {
        cc = undoStep.cellChanges[i];
        paintOnCanvas(cc.col, cc.row, cc.oldShapeName, cc.oldColor);
        changes.cells.push({
            col: cc.col,
            row: cc.row,
            shapeName: cc.oldShapeName,
            color: cc.oldColor,
        });
    }
}
}

redoStack.push(undoStep);
updateUndoRedoButtons();

if (socket) {
    console.log("socket.io <- changes (undo)");
    console.log(changes);
    socket.emit("changes", changes);
}
}
}

function doRedo() {
    if (redoStack.length > 0) {
        var redoStep = redoStack.pop();
        var changes = {
            cells: [],
        };
        if (redoStep.shiftChange) {
            var dir = redoStep.shiftChange.dir;
            if (dir == "left") {
                gridArtwork.doShiftLeft(grid);
                changes.shift = "left";
            } else if (dir == "right") {
                gridArtwork.doShiftRight(grid);
                changes.shift = "right";
            } else if (dir == "up") {
                gridArtwork.doShiftUp(grid);
                changes.shift = "up";
            }
        }
    }
}

```



```

    } else if (dir == "down") {
      gridArtwork.doShiftDown(grid);
      changes.shift = "down";
    }
  } else if (redoStep.backgroundChange) {
    setBackgroundColor(redoStep.backgroundChange.newColor);
    changes.backgroundColor = redoStep.backgroundChange.newColor;
  } else if (redoStep.workspaceChange) {
    applyWorkspaceSize(
      undoStep.workspaceChange.newWorkspace.width,
      undoStep.workspaceChange.newWorkspace.height,
      undoStep.workspaceChange.newWorkspace.cellSize,
      undoStep.workspaceChange.newWorkspace.gridThickness,
      undoStep.workspaceChange.newWorkspace.gridColor,
      undoStep.workspaceChange.newWorkspace.backgroundColor
    );
    changes.workspace = undoStep.workspaceChange.newWorkspace;
  } else {
    for (var i = 0; i < redoStep.cellChanges.length; i++) {
      cc = redoStep.cellChanges[i];
      paintOnCanvas(cc.col, cc.row, cc.newShapeName, cc.newColor);
      changes.cells.push({
        col: cc.col,
        row: cc.row,
        shapeName: cc.newShapeName,
        color: cc.newColor,
      });
    }
  }
}

undoStack.push(redoStep);
updateUndoRedoButtons();

if (socket) {
  console.log("socket.io <- changes (redo)");
  console.log(changes);
  socket.emit("changes", changes);
}
}
}

function pasteSelection() {
  var pasteCells = selection.getPasteCells();
  for (var i = 0; i < pasteCells.length; i++) {
    var cc = pasteCells[i];
    storeUndoCell(cc.col, cc.row, cc.shapeName, cc.color);
    paintOnCanvas(cc.col, cc.row, cc.shapeName, cc.color);
  }
  changed = true;

  if (socket != null) {
    var changes = {
      cells: pasteCells,
    };
    console.log("socket.io <- changes (pasteSelection)");
    console.log(changes);
    socket.emit("changes", changes);
  }
}

```

```

    }
}

function setBackgroundColor(color) {
    backgroundColor = color;
    $("#canvas").css("background-color", backgroundColor);
}

function showWarningMessage(message) {
    var messageModal = $("#message-modal");
    messageModal.find("#message-text").text(message);
    messageModal.modal();
}

function showCircleLoader() {
    $("body").append(
        '<div id="spinner-fullscreen" class="spinner-filesystem-wrapper"><div class="spinner wide"></div></div>'
    );
}

function hideCircleLoader() {
    $("#spinner-fullscreen").remove();
}

function initPropertiesDialog() {
    $("#btn-properties-save").click(function () {
        var artworkName = $("#modal_artwork_name").val();
        if (artworkName === "") {
            var messageModal = $("#message-modal");
            messageModal.find("#message-text").text("Please enter artwork name");
            messageModal.modal();
            return;
        }

        $("#artwork_name").val($("#modal_artwork_name").val());
        $("#artwork_tags").val($("#modal_artwork_tags").val());
        $("#artwork_description").val($("#modal_artwork_description").val());

        $("#properties-modal").modal("hide");

        var modalAction = $("#modal_success_action").val();
        if (modalAction === "save") {
            saveArtwork();
        }
    });

    initialTagsValue = $("input[name=modal_artwork_tags]").val();

    var tags = $("#modal_artwork_tags");
    tags.tagsinput();
    tags
        .tagsinput("input")
        .typeahead({
            name: "dataset",
            remote: "/tag-typeahead?query=%QUERY",
        })
}

```

```

.bind(
  "typeahead:selected",
  $.proxy(function (obj, datum) {
    tags.tagsinput("add", datum.value);
    tags.tagsinput("input").typeahead("setQuery", "");
  }, $("input"))
);
tags.on("beforeItemAdd", function (event) {
  if (event.item.length <= 1 || event.item.length > 64) {
    $("#modal-tags-hint").show();
    event.cancel = true;
  } else {
    $("#modal-tags-hint").hide();
  }
});

$("#modal_description_label").click(function () {
  if (modalDescriptionVisible) {
    $("#modal_description_label i")
      .removeClass("icon-caret-down")
      .addClass("icon-caret-right");
    $("#modal_artwork_description").hide();
    propertiesDialog_updateDescriptionLabelPreview();
    $("#modal_description_label")
      .find(".save-dialog-expandable-content-preview")
      .show();
    modalDescriptionVisible = false;
  } else {
    $("#modal_description_label i")
      .removeClass("icon-caret-right")
      .addClass("icon-caret-down");
    $("#modal_artwork_description").show();
    $("#modal_description_label")
      .find(".save-dialog-expandable-content-preview")
      .hide();
    modalDescriptionVisible = true;
  }
});

$("#modal_tags_label").click(function () {
  if (modalTagsVisible) {
    $("#modal_tags_label i")
      .removeClass("icon-caret-down")
      .addClass("icon-caret-right");
    $("#modal_artwork_tags_field").hide();
    $("#modal-tags-hint").hide();
    propertiesDialog_updateTagsLabelPreview();
    $("#modal_tags_label")
      .find(".save-dialog-expandable-content-preview")
      .show();
    modalTagsVisible = false;
  } else {
    $("#modal_tags_label i")
      .removeClass("icon-caret-right")
      .addClass("icon-caret-down");
    $("#modal_artwork_tags_field").show();
    $("#modal_tags_label")

```

```

        .find(".save-dialog-expandable-content-preview")
        .hide();
    modalTagsVisible = true;
    }
    });

$("#modal_square_grid_special_properties_label").click(function () {
    if (modalSquareGridSpecialPropertiesVisible) {
        $("#modal_square_grid_special_properties_label i")
            .removeClass("icon-caret-down")
            .addClass("icon-caret-right");
        $(".save-dialog-special-properties-frame").hide();
        modalSquareGridSpecialPropertiesVisible = false;
    } else {
        $("#modal_square_grid_special_properties_label i")
            .removeClass("icon-caret-right")
            .addClass("icon-caret-down");
        $(".save-dialog-special-properties-frame").show();
        modalSquareGridSpecialPropertiesVisible = true;
    }
    });

$("#properties-modal").on("shown.bs.modal", function () {
    $("#modal_artwork_name").focus();
    });
}

function initShiftPanel() {
    $("#shift-toolbar-header").click(function () {
        if ($(this).attr("content-visible")) {
            $("#shift-toolbar-content").slideUp();
            $("#shift-toolbar-header i").removeClass("icon-chevron-up");
            $("#shift-toolbar-header i").addClass("icon-chevron-down");
            $(this).removeAttr("content-visible");
        } else {
            $("#shift-toolbar-content").slideDown();
            $("#shift-toolbar-header i").removeClass("icon-chevron-down");
            $("#shift-toolbar-header i").addClass("icon-chevron-up");
            $(this).attr("content-visible", "visible");
        }
    });
}

var sendShiftBySocket = function (shift) {
    if (socket) {
        var changes = {
            shift: shift,
        };

        console.log("socket.io <- changes (shiftWorkspace)");
        console.log(changes);
        socket.emit("changes", changes);
    }
};

$("#btn-shift-left").click(function () {
    removedCells = gridArtwork.doShiftLeft(grid);
    changed = true;

```

```

undoStep = new UndoStep();
undoStep.setShiftChange("left", removedCells);
undoStack.push(undoStep);
redoStack = [];
updateUndoRedoButtons();
sendShiftBySocket("left");
});

$("#btn-shift-right").click(function () {
removedCells = gridArtwork.doShiftRight(grid);
changed = true;

undoStep = new UndoStep();
undoStep.setShiftChange("right", removedCells);
undoStack.push(undoStep);
redoStack = [];
updateUndoRedoButtons();
sendShiftBySocket("right");
});

$("#btn-shift-up").click(function () {
removedCells = gridArtwork.doShiftUp(grid);
changed = true;

undoStep = new UndoStep();
undoStep.setShiftChange("up", removedCells);
undoStack.push(undoStep);
redoStack = [];
updateUndoRedoButtons();
sendShiftBySocket("up");
});

$("#btn-shift-down").click(function () {
removedCells = gridArtwork.doShiftDown(grid);
changed = true;

undoStep = new UndoStep();
undoStep.setShiftChange("down", removedCells);
undoStack.push(undoStep);
redoStack = [];
updateUndoRedoButtons();
sendShiftBySocket("down");
});
}

function applyWorkspaceSize(
newWidth,
newHeight,
newCellSize,
gridThickness,
gridColor,
backgroundColor
) {
$("#canvas").css("width", newWidth).css("height", newHeight);
grid.workspaceWidth = newWidth;
grid.workspaceHeight = newHeight;

```

```

grid.cellSize = newCellSize;
grid.gridThickness = gridThickness;
grid.gridColor = gridColor;

paper.remove();
paper = new Raphael("canvas", newWidth, newHeight);
grid.paintGrid(paper);
selection.paper = paper;

setBackgroundColors(backgroundColors);

var oldGridArtwork = gridArtwork;
gridArtwork = new GridArtwork();
for (var row = 0; row < oldGridArtwork.cells.length; row++) {
  for (var col = 0; col < oldGridArtwork.cells[row].length; col++) {
    var cell = oldGridArtwork.cells[row][col];
    if (cell) {
      var cellRect = grid.getCellRect(col, row);
      if (
        cellRect.left + cellRect.width < newWidth &&
        cellRect.top + cellRect.height < newHeight
      ) {
        paintOnCanvas(col, row, cell.shapeName, cell.color);
      }
    }
  }
}

$("#workspace-width").val(newWidth);
$("#workspace-height").val(newHeight);
$("#workspace-cell-size").val(newCellSize);
$("#workspace-grid-thickness").val(gridThickness);
$("#workspace-grid-color").minicolors("value", gridColor);
$("#workspace-background-color").minicolors("value", backgroundColor);
}

function initSizePanel() {
$("#size-toolbar-header").click(function () {
  if ($(this).attr("content-visible")) {
    $("#size-toolbar-content").slideUp();
    $("#size-toolbar-header i").removeClass("icon-chevron-up");
    $("#size-toolbar-header i").addClass("icon-chevron-down");
    $(this).removeAttr("content-visible");
  } else {
    $("#size-toolbar-content").slideDown();
    $("#size-toolbar-header i").removeClass("icon-chevron-down");
    $("#size-toolbar-header i").addClass("icon-chevron-up");
    $(this).attr("content-visible", "visible");
  }
});

$("#btn-set-workspace-size").click(function () {
  // !!!!
  var newWidth = parseInt($("#workspace-width").val(), 10);
  var newHeight = parseInt($("#workspace-height").val(), 10);
  var newCellSize = parseInt($("#workspace-cell-size").val(), 10);
  var newGridThickness = parseInt($("#workspace-grid-thickness").val(), 10);

```

```

var newGridColor = $("#workspace-grid-color").val();
var newBackgroundColor = $("#workspace-background-color").val();

undoStep = new UndoStep();
undoStep.setWorkspaceChange(
  {
    width: grid.workspaceWidth,
    height: grid.workspaceHeight,
    cellSize: grid.cellSize,
    gridThickness: grid.gridThickness,
    gridColor: grid.gridColor,
    backgroundColor: backgroundColor,
  },
  {
    width: newWidth,
    height: newHeight,
    cellSize: newCellSize,
    gridThickness: newGridThickness,
    gridColor: newGridColor,
    backgroundColor: newBackgroundColor,
  }
);
undoStack.push(undoStep);
redoStack = [];
updateUndoRedoButtons();

if (newWidth < 200 || newWidth > 4000) {
  alert("Artwork width should be between 200 and 4000 pixels.");
  return;
}

if (newHeight < 200 || newHeight > 4000) {
  alert("Artwork height should be between 200 and 4000 pixels");
  return;
}

if (newCellSize < 10 || newCellSize > 32) {
  alert("Cell size should be between 10 and 32 pixels");
  return;
}

if (newGridThickness < 1 || newGridThickness > 4) {
  alert("Grid thickness should be between 1 and 4 pixels");
  return;
}

applyWorkspaceSize(
  newWidth,
  newHeight,
  newCellSize,
  newGridThickness,
  newGridColor,
  newBackgroundColor
);

$("#workspace-size-modal").modal("hide");

```

```

if (socket) {
  var changes = {
    workspace: {
      width: newWidth,
      height: newHeight,
      cellSize: newCellSize,
      gridThickness: newGridThickness,
      gridColor: newGridColor,
      backgroundColor: newBackgroundColor,
    },
  };

  console.log("socket.io <- changes (setWorkspaceSize)");
  console.log(changes);
  socket.emit("changes", changes);
}
});
}

function initCopyPastePanel() {
  $("#copy-paste-toolbar-header").click(function () {
    if ($(this).attr("content-visible")) {
      $("#copy-paste-toolbar-content").slideUp();
      $("#copy-paste-toolbar-header i").removeClass("icon-chevron-up");
      $("#copy-paste-toolbar-header i").addClass("icon-chevron-down");
      $(this).removeAttr("content-visible");
    } else {
      $("#copy-paste-toolbar-content").slideDown();
      $("#copy-paste-toolbar-header i").removeClass("icon-chevron-down");
      $("#copy-paste-toolbar-header i").addClass("icon-chevron-up");
      $(this).attr("content-visible", "visible");
      if (!localStorage["dontShowCopyPasteMessage"]) {
        $("#copy-paste-message-modal").modal();
      }
    }
  });

  $("#btn-copy-mode").click(function (event) {
    if (mode == "copy") {
      setMode("paint");
    } else {
      setMode("copy");
    }
  });

  $("#btn-paste-mode").click(function (event) {
    if (mode == "paste") {
      setMode("paint");
    } else {
      setMode("paste");
    }
  });
}

function initDrawingToolsPanel() {
  $("#btn-pick-color").click(function () {
    setMode("pick-color");
  });
}

```



```

});

$("#btn-pencil").click(function () {
  setMode("paint");
});

$("#btn-erase").click(function () {
  setMode("erase");
});

$("#btn-flood-fill").click(function () {
  setMode("fill");
});

if (grid.getAdjacentCells && grid.isCellInsideWorkspace) {
  $("#btn-flood-fill").show();
}

if (grid.drawTools) {
  for (var toolName in grid.drawTools) {
    var tool = grid.drawTools[toolName];
    var button = $(
      `<button class="btn btn-sm btn-default painter-btn-tool btn-draw-tool" type="button" tool-
name="${toolName}" title="${tool.title}"></button>`
    );
    $("#draw-tools-bar").append(button);
  }
  $(".btn-draw-tool").click(function (event) {
    var toolName = $(this).attr("tool-name");
    setMode(toolName);
  });
}
}

function initShapesToolbar() {
  shapesToolbarGrid = gridFactory[artwork.layers[0].grid]();
  shapesToolbarGrid.cellSize = 24; // TODO toolbar cell size from grid defaults

  var shapeElements = "";
  for (var i = 0; i < shapesToolbarGrid.shapesToolbar.length; i++) {
    shapeElements +=
      '<div class="shapes-toolbar-row" id="shapes-toolbar-row-' + i + "'>';
    for (var j = 0; j < shapesToolbarGrid.shapesToolbar[i].length; j++) {
      var shapeName = shapesToolbarGrid.shapesToolbar[i][j];
      shapeElements +=
        '<span id="shape-' +
        shapeName +
        '" class="grid-shape-button" shape-name="' +
        shapeName +
        "'></span>';
    }
    shapeElements += "</div>";
  }

  $("#shapes-toolbar").html(shapeElements);

  selectedShapeName = "flat";

```

```

for (var shapeName in shapesToolbarGrid.shapes) {
  paintShapeToolButton(shapeName);
}

$(".grid-shape-button").click(function () {
  selectShape($(this).attr("shape-name"));
});

if (shapesToolbarGrid.shapesToolbar.length <= 2) {
  $("#shapes-toolbar-header i").hide();
} else {
  for (var i = 2; i < shapesToolbarGrid.shapesToolbar.length; i++) {
    $("#shapes-toolbar-row-" + i).hide();
  }

  $("#shapes-toolbar-header").click(function () {
    if ($(this).attr("content-visible")) {
      var selectedRowIndex = 1;
      for (var i = 2; i < shapesToolbarGrid.shapesToolbar.length; i++) {
        if (shapesToolbarGrid.shapesToolbar[i].includes(selectedShapeName)) {
          selectedRowIndex = i;
          break;
        }
      }
      for (var i = 1; i < shapesToolbarGrid.shapesToolbar.length; i++) {
        if (i != selectedRowIndex) {
          $("#shapes-toolbar-row-" + i).slideUp();
        }
      }
      $("#shapes-toolbar-header i").removeClass("icon-chevron-up");
      $("#shapes-toolbar-header i").addClass("icon-chevron-down");
      $(this).removeAttr("content-visible");
    } else {
      for (var i = 0; i < shapesToolbarGrid.shapesToolbar.length; i++) {
        $("#shapes-toolbar-row-" + i).slideDown();
      }
      $("#shapes-toolbar-header i").removeClass("icon-chevron-down");
      $("#shapes-toolbar-header i").addClass("icon-chevron-up");
      $(this).attr("content-visible", "visible");
    }
  });
}

function onCanvasMouseDown(event) {
  paperMouseDown = true;
  updateCellCoordiantesPanel(event);

  if (mode == "paint") {
    undoStack.push(new UndoStep());
    redoStack = [];
    updateUndoRedoButtons();
    paintOnCanvasByMouseEvent(event);
  } else if (mode == "erase") {
    undoStack.push(new UndoStep());
    redoStack = [];
    updateUndoRedoButtons();
  }
}

```

```

    eraseOnCanvasByMouseEvent(event);
} else if (mode == "pick-color") {
    pickColorByMouseEvent(event);
} else if (mode == "copy") {
    selectOnCanvasByMouseEvent(event);
} else if (mode == "paste") {
    undoStack.push(new UndoStep());
    redoStack = [];
    updateUndoRedoButtons();
    pasteSelection();
    selection.pasteFinished();
} else if (mode == "fill") {
    undoStack.push(new UndoStep());
    redoStack = [];
    updateUndoRedoButtons();
    fillAreaOnCanvasByMouseEvent(event);
} else if (grid.drawTools && grid.drawTools[mode]) {
    undoStack.push(new UndoStep());
    redoStack = [];
    updateUndoRedoButtons();
    draftDrawToolOnCanvasByMouseEvent(event);
}
}

function onCanvasMouseUp(event) {
    paperMouseDown = false;

    if (mode == "paste") {
        setMode("paint");
    } else if (grid.drawTools && grid.drawTools[mode]) {
        drawToolOnCanvas();
    }
}

function onCanvasMouseMove(event) {
    updateCellCoordinatesPanel(event);
    if (mode == "paint") {
        paintOnCanvasByMouseEvent(event);
    } else if (mode == "erase") {
        eraseOnCanvasByMouseEvent(event);
    } else if (mode == "copy") {
        selectOnCanvasByMouseEvent(event);
    } else if (mode == "paste") {
        changePastePositionByMouseEvent(event);
    } else if (grid.drawTools && grid.drawTools[mode]) {
        draftDrawToolOnCanvasByMouseEvent(event);
    }
}

function is_touch_device() {
    try {
        document.createEvent("TouchEvent");
        return true;
    } catch (e) {
        return false;
    }
}

```

```

var touchMode = "ready";
var touchDown = false;
var ongoingTouches = new Array();

function getOngoingTouch(identifier) {
  for (let i = 0; i < ongoingTouches.length; i++) {
    if (ongoingTouches[i].identifier == identifier) {
      return ongoingTouches[i];
    }
  }
}

function setOngoingTouch(identifier, pageX, pageY) {
  for (let i = 0; i < ongoingTouches.length; i++) {
    if (ongoingTouches[i].identifier == identifier) {
      ongoingTouches[i].pageX = pageX;
      ongoingTouches[i].pageY = pageY;
      return;
    }
  }
  ongoingTouches.push({
    identifier: identifier,
    pageX: pageX,
    pageY: pageY,
  });
}

function deleteOngoingTouch(identifier) {
  let index = -1;
  for (let i = 0; i < ongoingTouches.length; i++) {
    if (ongoingTouches[i].identifier == identifier) {
      index = i;
      break;
    }
  }
  ongoingTouches.splice(index, 1);
}

function onCanvasTouchStart(evt) {
  evt.preventDefault();
  let touches = evt.originalEvent.changedTouches;
  for (let i = 0; i < touches.length; i++) {
    setOngoingTouch(touches[i].identifier, touches[i].pageX, touches[i].pageY);
  }

  if (touchMode == "ready") {
    if (ongoingTouches.length == 1) {
      touchMode = "paint";
    } else if (ongoingTouches.length == 2) {
      touchMode = "pan";
    } else {
      touchMode = "invalid";
    }
  } else if (touchMode == "paint") {
    if (ongoingTouches.length == 1) {
    } else if (ongoingTouches.length == 2) {
    }
  }
}

```

```

    touchMode = "pan";
  } else {
    touchMode = "invalid";
  }
}
}

function onCanvasTouchMove(evt) {
  evt.preventDefault();

  if (touchMode == "paint") {
    let currentTouch = evt.originalEvent.changedTouches[0];
    let prevTouch = getOngoingTouch(currentTouch.identifier);
    if (!prevTouch) {
      console.log("no prev touch", currentTouch.identifier, ongoingTouches);
      return;
    }
    if (
      Math.abs(currentTouch.pageX - prevTouch.pageX) < 4 &&
      Math.abs(currentTouch.pageY - prevTouch.pageY) < 4
    ) {
      console.log("no move");
      return;
    }
    let mouseEvent = {
      pageX: currentTouch.pageX,
      pageY: currentTouch.pageY,
      which: 1,
      altKey: evt.altKey,
      ctrlKey: evt.ctrlKey,
      shiftKey: evt.shiftKey,
    };
    if (!touchDown) {
      onCanvasMouseDown(mouseEvent);
      touchDown = true;
    }
    onCanvasMouseMove(mouseEvent);
    setOngoingTouch(
      currentTouch.identifier,
      currentTouch.pageX,
      currentTouch.pageY
    );
  } else if (
    touchMode == "pan" &&
    evt.originalEvent.changedTouches.length == 2 &&
    ongoingTouches.length == 2
  ) {
    let newTouches = evt.originalEvent.changedTouches;
    let newTouch_1 = newTouches[0];
    let newTouch_2 = newTouches[1];
    let oldTouch_1 = getOngoingTouch(newTouch_1.identifier);
    let oldTouch_2 = getOngoingTouch(newTouch_2.identifier);
    let dx1 = newTouch_1.pageX - oldTouch_1.pageX;
    let dy1 = newTouch_1.pageY - oldTouch_1.pageY;
    let dx2 = newTouch_2.pageX - oldTouch_2.pageX;
    let dy2 = newTouch_2.pageY - oldTouch_2.pageY;
    let scalarMult = dx1 * dx2 + dy1 * dy2;
  }
}

```

```

if (scalarMult > 0) {
  let dx = (dx1 + dx2) / 2;
  let dy = (dy1 + dy2) / 2;
  try {
    $("#canvas").parent()[0].scrollBy(-dx, -dy);
  } catch (e) {
    console.log(e);
  }
}
setOngoingTouch(newTouch_1.identifier, newTouch_1.pageX, newTouch_1.pageY);
setOngoingTouch(newTouch_2.identifier, newTouch_2.pageX, newTouch_2.pageY);
}
}

function onCanvasTouchEnd(evt) {
  evt.preventDefault();
  let touches = evt.originalEvent.changedTouches;
  if (touchMode == "paint" && touches.length == 1) {
    if (touchDown) {
      let mouseEvent = {
        pageX: touches[0].pageX,
        pageY: touches[0].pageY,
        which: 1,
        altKey: evt.altKey,
        ctrlKey: evt.ctrlKey,
        shiftKey: evt.shiftKey,
      };
      onCanvasMouseMove(mouseEvent);
      onCanvasMouseUp(mouseEvent);
    }
  }
}

for (let i = 0; i < touches.length; i++) {
  deleteOngoingTouch(touches[i].identifier);
}

touchDown = false;
if (ongoingTouches.length == 0) {
  touchMode = "ready";
} else {
  touchMode = "invalid";
}
}

function onCanvasTouchCancel(evt) {
  onCanvasTouchEnd(evt);
}

function addCollaborator(collaborator) {
  var sid = collaborator.sid;
  var exists = false;
  for (var i = 0; i < collaboratorsOnline.length; i++) {
    if (collaboratorsOnline[i].sid == sid) {
      exists = true;
      break;
    }
  }
}

```

```

if (!exists) {
  collaboratorsOnline.push(collaborator);
}
}

function deleteCollaborator(sid) {
  var index = -1;
  for (var i = 0; i < collaboratorsOnline.length; i++) {
    if (collaboratorsOnline[i].sid == sid) {
      index = i;
      break;
    }
  }
  collaboratorsOnline.splice(index, 1);
}

function updateCollaboratorsPanel() {
  if (collaboratorsOnline.length == 0) {
    $(".group-image-users-online").hide();
    $("#call-collaborators").show();
    return;
  } else {
    $(".group-image-users-online").show();
    $("#call-collaborators").hide();
  }
  var html = "";
  for (var i = 0; i < collaboratorsOnline.length; i++) {
    var c = collaboratorsOnline[i];
    html += `<div class="user-icon" style="background-image: url(${c.user.avatar_url})"
title="${c.user.nickname}"></div>`;
  }
  $(".group-image-users-online").html(html);
}

function sendMessageToChat() {
  if (!socket) {
    return;
  }

  let text = $("#group-image-chat-input").val().trim();
  $("#group-image-chat-input").val("");
  if (text == "") {
    return;
  }

  console.log("socket.io <- add_chat_message");
  socket.emit("add_chat_message", { text: text });
}

function safe_tags(str) {
  return str.replace(/&/g, "&amp;").replace(/</g, "&lt;").replace(/>/g, "&gt;");
}

var lastChatMessageUserId = null;
var chatWindowVisible = false;

function showChatMessage(chat_message) {

```

```

let tokenPayload = JSON.parse(atob(exchangeToken.split(".")[1]));
let currentUser = tokenPayload.user;
let messageClass = "other-user";
if (chat_message.user.user_id == currentUser.user_id) {
  messageClass = "current-user";
}

let messageHtml = null;
if (lastChatMessageUserId == chat_message.user.user_id) {
  messageHtml = `
    <div class="group-image-chat-message ${messageClass}">
      <div class="chat-message-content">
        <div class="chat-message-text">
          ${chat_message.text}
        </div>
      </div>
    </div>`;
} else {
  messageHtml = `
    <div class="group-image-chat-message ${messageClass}">
      <div class="chat-message-avatar" style="background-image: url(${
        chat_message.user.avatar_url
      })"></div>
      <div class="chat-message-content">
        <div class="chat-message-
author">${safe_tags(chat_message.user.nickname)}</div>
        <div class="chat-message-text">
          ${safe_tags(chat_message.text)}
        </div>
      </div>
    </div>`;
}
$(".group-image-chat-messages-list").append(messageHtml);
$(".group-image-chat-messages-container").animate(
  { scrollTop: $(".group-image-chat-messages-list").height() },
  300
);
showChatWindow();

lastChatMessageUserId = chat_message.user.user_id;
}

function showChatWindow() {
  $(".group-image-chat-show-button")
    .html('<i class="glyphicon glyphicon-chevron-down"></i>')
    .show();
  $(".group-image-chat-messages").slideDown();
  chatWindowVisible = true;
}

function hideChatWindow() {
  $(".group-image-chat-show-button").html(
    '<i class="glyphicon glyphicon-chevron-up"></i>'
  );
  $(".group-image-chat-messages").slideUp();
  chatWindowVisible = false;
}

```



```

function toggleChatWindow() {
  if (chatWindowVisible) {
    hideChatWindow();
  } else {
    showChatWindow();
  }
}

var initComplete = false;
function initialPaintArtwork() {
  initComplete = true;
  hideCircleLoader();
  if (artwork.version.major == 1) {
    var layer = artwork.layers[0];
    for (var i = 0; i < layer.cells.length; i++) {
      paintOnCanvas(
        layer.cells[i].col,
        layer.cells[i].row,
        layer.cells[i].shape,
        layer.cells[i].color
      );
    }
  } else if (artwork.version.major == 2) {
    var layer = artwork.layers[0];
    for (var rowIndex = 0; rowIndex < layer.rows.length; rowIndex++) {
      var cellRow = layer.rows[rowIndex];
      for (var cellIndex = 0; cellIndex < cellRow.cells.length; cellIndex++) {
        var cell = cellRow.cells[cellIndex];
        paintOnCanvas(cell[0], cellRow.row, cell[1], cell[2]);
      }
    }
  }
}

function drawCollaboratorPointerToCell(user, cell) {
  var item = null;
  var pointer = $("#collaborator-pointer-" + user.user_id);
  var cellRect = grid.getCellRect(cell.col, cell.row);
  var x = cellRect.left + cellRect.width / 2;
  var y = cellRect.top + cellRect.height / 2;

  if (pointer.length == 0) {
    html = `<div class="collaborator-pointer" id="collaborator-pointer-${user.user_id}">
      <div class="wrapper">
        <div class="arrow"></div>
        <div class="nickname">${user.nickname}</div>
        <div class="avatar" style="background-image:
url(${user.avatar_url})"></div>
      </div>`;
    pointer = $(html);
    console.log(pointer);
    $("##canvas-wrapper #canvas").append(pointer);
  }

  pointer.css("left", x - 10 + "px");
}

```

```

pointer.css("top", y - 30 + "px");
pointer.show();
}

$(function () {
adjustCanvasWrapper();
$(window).resize(function () {
adjustCanvasWrapper();
});

backgroundColor = artwork.backgroundColor;
$("#canvas")
.css("background-color", backgroundColor)
.css("width", artwork.canvasSize.width)
.css("height", artwork.canvasSize.height);

paper = new Raphael(
"canvas",
artwork.canvasSize.width,
artwork.canvasSize.height
);
grid = gridFactory[artwork.layers[0].grid]();
grid.cellSize = artwork.layers[0].cellSize;
grid.workspaceWidth = artwork.canvasSize.width;
grid.workspaceHeight = artwork.canvasSize.height;
grid.gridThickness = artwork.layers[0].gridThickness
? artwork.layers[0].gridThickness
: 1;
grid.gridColor = artwork.layers[0].gridColor
? artwork.layers[0].gridColor
: "#d0d0d0";
grid.paintGrid(paper);

selection.grid = grid;
selection.paper = paper;
selection.loadFromLocalStorage();

initShapesToolbar();
updateUndoRedoButtons();

$("#workspace-width").val(artwork.canvasSize.width);
$("#workspace-height").val(artwork.canvasSize.height);
$("#workspace-cell-size").val(artwork.layers[0].cellSize);
if (artwork.layers[0].gridThickness) {
$("#workspace-grid-thickness").val(artwork.layers[0].gridThickness);
} else {
$("#workspace-grid-thickness").val("1");
}
if (artwork.layers[0].gridColor) {
$("#workspace-grid-color").val(artwork.layers[0].gridColor);
} else {
$("#workspace-grid-color").val("#d0d0d0");
}
$("#workspace-background-color").val(artwork.backgroundColor);

$("#modal_artwork_grid_visible")[0].checked = artwork["gridVisible"];
$("#modal_artwork_pixel_art")[0].checked = artwork["additionalPixelImage"];

```

```

$("#modal_transparent_background")[0].checked =
  artwork["transparentBackground"];

$("#canvas")
  .mousedown(onCanvasMouseDown)
  .mouseup(onCanvasMouseUp)
  .mousemove(onCanvasMouseMove);

if (is_touch_device()) {
  console.log("Touch present");
  $("#canvas")
    .on("touchstart", onCanvasTouchStart)
    .on("touchmove", onCanvasTouchMove)
    .on("touchend", onCanvasTouchEnd)
    .on("touchcancel", onCanvasTouchCancel);
}

$.mask.definitions["k"] = "[A-Fa-f0-9]";
$("#color-picker-text").mask("#kkkkkk");
$("#color-picker-text")
  .change(function () {
    selectColorFromText($(this).val());
  })
  .keyup(function () {
    selectColorFromText($(this).val());
  });

$("#color-picker")
  .minicolors({
    inline: true,
    control: "brightness",
    change: function (hex, opacity) {
      selectColorFromPicker(hex);
    },
  })
  .minicolors("value", selectedColor);

$("#workspace-grid-color").minicolors({
  theme: "bootstrap",
  letterSpacing: "uppercase",
});

$("#workspace-background-color").minicolors({
  theme: "bootstrap",
  letterSpacing: "uppercase",
});

$("#btn-set-background-color").click(function () {
  undoStep = new UndoStep();
  undoStep.setBackgroundChange(background-color, selected-color);
  undoStack.push(undoStep);
  redoStack = [];
  updateUndoRedoButtons();

  setBackground-color(selected-color);

  if (socket) {

```

```

var changes = {
  backgroundColor: selectedColor,
};

console.log("socket.io <- changes (setBackgroundColor)");
console.log(changes);
socket.emit("changes", changes);
}
});

$("#btn-save").click(function () {
  showPropertiesDialog("save");
});

$("#btn-workspace-size").click(function () {
  $("#workspace-size-modal").modal("show");
});

window.onbeforeunload = function () {
  if (changed) {
    return "Your drawing was changed. Do you want to leave without saving?";
  }
};

//Create color palette
var paletteHtml = "";
for (var i = 0; i < 22; i++) {
  paletteHtml += "<div></div>";
  recentColors.push("#FFFFFF");
}
$("#color-palette").html(paletteHtml);
$("#color-palette div").click(function () {
  hexColor = rgb2hex($(this).css("background-color"));
  $("#color-picker").minicolors("value", hexColor);
  //selectColor(hexColor);
});

//Fill color palette with recent colors
if (artwork.recentColors) {
  $paletteDivs = $("#color-palette div");
  for (
    var i = 0;
    i < $paletteDivs.length && i < artwork.recentColors.length;
    i++
  ) {
    $($paletteDivs[i]).css("background-color", artwork.recentColors[i]);
    recentColors[i] = artwork.recentColors[i];
  }
}
selectColor(recentColors[0]);

setMode("paint");

$("#btn-undo").click(function () {
  doUndo();
});

```

```

$("#btn-redo").click(function () {
  doRedo();
});

//Disable context menu on canvas
$("#canvas").bind("contextmenu", function (e) {
  return false;
});

$("body").keydown(function (event) {
  if (event.ctrlKey && event.keyCode == 90) {
    //Ctrl-Z
    doUndo();
  }
});

$("#copy-paste-message-modal").on("hidden.bs.modal", function () {
  if ($("#chk-dont-show-copy-paste-message")[0].checked) {
    localStorage["dontShowCopyPasteMessage"] = true;
  }
});

initPropertiesDialog();
initDrawingToolsPanel();
initShiftPanel();
initSizePanel();
initCopyPastePanel();

if (exchangeToken) {
  $(".group-image-online").show();

  $("#btn-send-message-to-chat").click(sendMessageToChat);
  $("#group-image-chat-input").on("keypress", function (e) {
    if (e.which == 13) {
      sendMessageToChat();
    }
  });
  $(".group-image-chat-show-button").click(toggleChatWindow);

  showCircleLoader();
  timeoutTaskId = setTimeout(initialPaintArtwork, 5000);

  socket = io(exchangeUrl);
  socket.on("connect", (data) => {
    console.log("socket.io => connect");
    console.log("socket.io <- login");
    socket.emit("login", { token: exchangeToken });
    $("#socketio-online").show();
    $("#call-collaborators").show();
    $("#group-image-chat-wrapper").show();
    $("#socketio-offline").hide();
  });
  socket.on("disconnect", (data) => {
    console.log("socket.io => disconnect");
    $("#socketio-online").hide();
    $("#call-collaborators").hide();
    $("#group-image-chat-wrapper").hide();
  });
}

```

```

$("#socketio-offline").show();
collaboratorsOnline = [];
updateCollaboratorsPanel();
});
socket.on("login_ok", (data) => {
  console.log("socket.io => login_ok");

  var tokenPayload = JSON.parse(atob(exchangeToken.split(".")[1]));
  console.log("socket.io <- hello");
  console.log(tokenPayload.user);
  socket.emit("hello", tokenPayload.user);

  console.log("socket.io <- who_is_here");
  socket.emit("who_is_here", {});
});
socket.on("login_fail", (data) => {
  console.log("socket.io => login_fail");
});
socket.on("you_are_first", (data) => {
  console.log("socket.io => you_are_first");
  if (!initComplete) {
    clearTimeout(timeoutTaskId);
    initialPaintArtwork();
  }
});
socket.on("hello", (data) => {
  console.log("socket.io => hello");
  console.log(data);
  addCollaborator(data);
  updateCollaboratorsPanel();
});
socket.on("bye", (data) => {
  console.log("socket.io => bye");
  console.log(data);
  var sid = data.sid;
  deleteCollaborator(sid);
  updateCollaboratorsPanel();
});
socket.on("who_is_here", (data) => {
  console.log("socket.io => who_is_here");
  var tokenPayload = JSON.parse(atob(exchangeToken.split(".")[1]));
  console.log("socket.io <- hello");
  console.log(tokenPayload.user);
  socket.emit("hello", tokenPayload.user);
});
socket.on("ask_image", (data) => {
  console.log("socket.io => ask_image");
  new_data = {
    sid: data.sid,
    image: prepareArtworkToSave(),
  };
  console.log("socket.io <- full_image");
  console.log(new_data);
  socket.emit("full_image", new_data);
});
socket.on("redirect_full_image", (data) => {
  console.log("socket.io => redirect_full_image");

```

```

if (!initComplete) {
  console.log(data);
  artwork = data;
  initialPaintArtwork();
}
});
socket.on("redirect_changes", (data) => {
  console.log("socket.io => redirect_changes");
  console.log(data);
  var user = data.user;
  var changes = data.changes;
  if (changes.cells) {
    var lastCell = null;
    for (var i = 0; i < changes.cells.length; i++) {
      var cell = changes.cells[i];
      paintOnCanvas(cell.col, cell.row, cell.shapeName, cell.color);
      lastCell = cell;
    }
    if (lastCell) {
      drawCollaboratorPointerToCell(data.user, lastCell);
    }
  }
  if (changes.backgroundColor) {
    setBackgroundColor(changes.backgroundColor);
  }
  if (changes.shift) {
    if (changes.shift == "left") {
      gridArtwork.doShiftLeft(grid);
    } else if (changes.shift == "right") {
      gridArtwork.doShiftRight(grid);
    } else if (changes.shift == "up") {
      gridArtwork.doShiftUp(grid);
    } else if (changes.shift == "down") {
      gridArtwork.doShiftDown(grid);
    }
  }
  if (changes.workspace) {
    applyWorkspaceSize(
      changes.workspace.width,
      changes.workspace.height,
      changes.workspace.cellSize,
      changes.workspace.gridThickness,
      changes.workspace.gridColor,
      changes.workspace.backgroundColor
    );
  }
});
socket.on("chat_message", (data) => {
  console.log("socket.io => chat_message");
  console.log(data);
  showChatMessage(data);
});
} else {
  initialPaintArtwork();
}
});

```

## Лістинг програми розмітки

```

<!DOCTYPE html>
<html>

<head>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Grid Paint</title>
  <link rel="apple-touch-icon" sizes="180x180" href="/apple-touch-icon.png">
  <link rel="icon" type="image/png" sizes="32x32" href="/favicon-32x32.png">
  <link rel="icon" type="image/png" sizes="16x16" href="/favicon-16x16.png">
  <link rel="manifest" href="/site.webmanifest">
  <link rel="mask-icon" href="/safari-pinned-tab.svg" color="#5bbad5">
  <meta name="msapplication-TileColor" content="#da532c">
  <meta name="theme-color" content="#e7e7e7">

  <link href="/bootstrap/css/bootstrap.css" rel="stylesheet">
  <link href="/typeahead/typeahead.css" rel="stylesheet">
  <link href="/font-awesome/css/font-awesome.min.css" rel="stylesheet">
  <link href="/css/main2.css?2019-03-31" rel="stylesheet">

  <link href="/css/jquery.minicolors.css" rel="stylesheet">
  <link href="/bootstrap-tagsinput/bootstrap-tagsinput.css" rel="stylesheet">
  <style>
    body {
      -webkit-touch-callout: none;
      -webkit-user-select: none;
      -khtml-user-select: none;
      -moz-user-select: none;
      -ms-user-select: none;
      user-select: none;
    }
  </style>

  <meta property="og:type" content="website" />

</head>

<body>

  <script src="/js/jquery-2.1.4.min.js"></script>
  <script src="/bootstrap/js/bootstrap.js"></script>
  <script src="/typeahead/typeahead.min.js"></script>

  <script src="/js/raphael-min.js"></script>
  <script src="/bootstrap-tagsinput/bootstrap-tagsinput.js"></script>
  <script src="/js/jquery.maskedinput.js"></script>
  <script src="/js/socket.io.js"></script>

```



```

    <nav class="navbar navbar-default navbar-fixed-top" role="navigation">
      <ul class="nav navbar-nav navbar-right header-user-panel nocollapse" style="font-size:
20px; float:right;">
        <li class="header-right-margin"></li>

        <li class="dropdown">
          <a href="#" class="dropdown-toggle header-avatar-dropdown-toggle"
data-toggle="dropdown">
            <div class="header-user-wrapper" style="border-
color:transparent">
              <div class="header-user" style="background-image:
url(/img/svg-buttons/empty-avatar.svg)"></div>
            </div>
          </a>
          <ul class="dropdown-menu">
            <li>
              <a href="/my-profile">
                <div class="my-profile-menu-text">My
profile</div>
                <div class="my-profile-menu-
email">max.kozubenko6@gmail.com</div>
              </a>
            </li>

            <li class="delimiter"></li>
            <li><a
              href="https://grid-
paint.com/_ah/logout?continue=https://accounts.google.com/Logout%3Fcontinue%3Dhttps://uc.appengin
e.google.com/_ah/logout%253Fcontinue%253Dhttps://google.com/url%25253Fsa%25253DD%252526q
%25253Dhttps://grid-
paint.com/%252526ust%25253D1665603412179444%252526usg%25253DAOvVaw2GRh0_ZAR7S61
BWL44gSy2%26service%3Dah">Logout</a>
            </li>
          </ul>
        </li>

      </ul>

      <button aria-controls="bs-navbar" aria-expanded="false" class="navbar-toggle collapsed"
data-target="#bs-navbar"
      data-toggle="collapse" type="button" style="margin-right: 0;">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>

      <div class="navbar-header">
        <a class="navbar-brand" href="/">
          
          Grid paint
        </a>
      </div>
    </nav>

```

```

<script>
    $.fn.flash = function (duration, iterations) {
        duration = duration || 1000; // Default to 1 second
        iterations = iterations || 1; // Default to 1 iteration
        var iterationDuration = duration / iterations;

        for (var i = 0; i < iterations; i++) {
            this.fadeOut(iterationDuration).fadeIn(iterationDuration);
        }
    }

    $(function () {
        $("input[name=q]")
            .typeahead({
                name: "dataset",
                remote: "/tag-typeahead?query=%QUERY"
            })

        $(''.blink').flash(5000, 10);
    });
</script>

<script src="/js/jquery.minicolors.js"></script>

<div class="row-fluid">
    <div class="painter-toolbar-full text-center">
        <div style="margin-bottom: 10px;">
            <button id="btn-workspace-size" class="btn btn-sm btn-default"
type="button" title="Workspace size"></i></button>
            <div class="btn-group">
                <button id="btn-undo" class="btn btn-sm" type="button"
title="Undo (Ctrl-Z)"></button>
                <button id="btn-redo" class="btn btn-sm" type="button"
title="Redo"></button>
            </div>
        </div>
        <div style="margin-bottom: 10px;" id="draw-tools-bar">
            <button id="btn-pencil" class="btn btn-sm btn-default painter-btn-tool
active" type="button"
title="Paint">
                
            </button><button id="btn-pick-color" class="btn btn-sm btn-default
painter-btn-tool" type="button"
title="Pick color">

```

```

        
    </button><button id="btn-erase" class="btn btn-sm btn-default painter-
btn-tool" type="button"
        title="Erase">
        
    </button><button id="btn-flood-fill" class="btn btn-sm btn-default
painter-btn-tool" type="button"
        title="Flood fill" style="display:none;">
        
    </button>
</div>
<div id="color-selection">
    <div id="color-picker"></div>
</div>

<div class="painter-toolbar">
    <input id="color-picker-text" />
</div>

<div class="painter-toolbar">
    <div id="selected-color"></div>
    <div id="color-palette"></div>
</div>
<!--
<div>
    <a href="#" id="btn-set-background-color">Set background color</a>
</div>
-->
<div class="painter-toolbar">
    <div class="painter-toolbar-header" id="shapes-toolbar-header"
style="cursor:pointer;">
        Shapes
        <div class="pull-right arrows-position"></div>
    </div>
    <div id="shapes-toolbar"></div>
</div>
<div class="painter-toolbar">
    <!-- Shift toolbar -->
    <div class="painter-toolbar-header" id="shift-toolbar-header"
style="cursor:pointer;">
        Shift
        <div class="pull-right arrows-position">
        </div>
    </div>
    <div id="shift-toolbar-content" style="display:none">
        <div class="painter-toolbar-container text-center">
            <button id="btn-shift-up" class="btn btn-sm"
type="button" title="Shift up"></button>
        </div>
        <div class="painter-toolbar-container text-center">
            <button id="btn-shift-left" class="btn btn-sm"
type="button" title="Shift left"></button>

```



```

    <div class="group-image-chat-messages-container">
      <div class="group-image-chat-messages-list">

        </div>
        <!--
        <div class="group-image-chat-message current-user">
          <div class="chat-message-avatar" style="background-image:
url(/images/avatar/5629499534213120.jpg)"></div>
          <div class="chat-message-content">
            <div class="chat-message-author">Pixel expert</div>
            <div class="chat-message-text">
              First message
            </div>
          </div>
        </div>
        <div class="group-image-chat-message current-user">
          <div class="chat-message-content">
            <div class="chat-message-text">
              Second message
            </div>
          </div>
        </div>

        <div class="group-image-chat-message other-user">
          <div class="chat-message-avatar" style="background-image:
url(/images/avatar/5629499534213120.jpg)"></div>
          <div class="chat-message-content">
            <div class="chat-message-author">Pixel expert</div>
            <div class="chat-message-text">
              First message
            </div>
          </div>
        </div>
        <div class="group-image-chat-message other-user">
          <div class="chat-message-content">
            <div class="chat-message-text">
              Second message
            </div>
          </div>
        </div>
      </div>
    </div>

    -->

    </div>
    <div class="group-image-chat-enter">
      <div class="input-group">
        <input type="text" class="form-control" id="group-image-chat-input"
placeholder="Type message" />
      </div>
    </div>
  </div>

  <div id="workspace-size-modal" class="modal fade" role="dialog">
    <div class="modal-dialog">
      <div class="modal-content">
        <div class="modal-header">

```

```

        <button type="button" class="close" data-dismiss="modal" aria-
hidden="true">&times;</button>
        <h4 class="modal-title">Workspace properties</h4>
    </div>
    <div class="modal-body">
        <form role="form">
            <div class="row" style="margin-bottom: 15px;">
                <div class="col-xs-3">
                    <div>Width (px)</div>
                    <div>
                        <input type="number"
class="form-control" id="workspace-width"
width">
                                name="workspace-
                                width">
                            </div>
                        </div>
                    </div>
                <div class="col-xs-3">
                    <div>Height (px)</div>
                    <div>
                        <input type="number"
class="form-control" id="workspace-height"
height">
                                name="workspace-
                                height">
                            </div>
                        </div>
                    </div>
                <div class="col-xs-6">
                    <div>Background color</div>
                    <div>
                        <input type="text" class="form-
control minicolors-input" value="#ffffff"
background-color" name="workspace-background-color">
                                id="workspace-
                                background-color" name="workspace-background-color">
                            </div>
                        </div>
                    </div>
                <div class="row">
                    <div class="col-xs-3">
                        <div>Cell size (px)</div>
                        <div>
                            <input type="number"
class="form-control" id="workspace-cell-size"
size">
                                    name="workspace-cell-
                                    size">
                                </div>
                            </div>
                        </div>
                    <div class="col-xs-3">
                        <div>Grid thickness (px)</div>
                        <div>
                            <input type="number"
class="form-control" id="workspace-grid-thickness"
thickness">
                                    name="workspace-grid-
                                    thickness">
                                </div>
                            </div>
                        </div>
                    <div class="col-xs-6">
                        <div>Grid color</div>
                        <div>

```

```

control minicolors-input" value="#d0d0d0"
color" name="workspace-grid-color">
    <input type="text" class="form-
    id="workspace-grid-
    </div>
    </div>
    </div>
    </form>
    </div>
    <div class="modal-footer">
    <button type="button" class="btn" data-
    dismiss="modal">Close</button>
    <button id="btn-set-workspace-size" type="button" class="btn
    btn-primary">Change</button>
    </div>
    </div>
    </div>
    <div id="properties-modal" class="modal fade" role="dialog">
    <div class="modal-dialog">
    <div class="modal-content">
    <div class="modal-header">
    <button type="button" class="close" data-dismiss="modal" aria-
    hidden="true">&times;</button>
    <h4 class="modal-title">Please, fill artwork properties before
    saving</h4>
    </div>
    <div class="modal-body">
    <form role="form">
    <div class="form-group">
    <label>Artwork title</label>
    <input type="text"
    name="modal_artwork_name" id="modal_artwork_name" value="qwerty"
    class="form-control" />
    </div>
    <div class="form-group">
    <label class="save-dialog-expandable-label"
    id="modal_description_label">
    <div class="save-dialog-expandable-
    label-gradient"></div>
    Artwork description <span
    content-preview"></span>
    <i class="icon icon-caret-right"></i>
    class="save-dialog-expandable-
    </label>
    <textarea name="modal_artwork_description"
    id="modal_artwork_description"
    class="form-control"></textarea>
    </div>
    <div class="form-group">
    <label class="save-dialog-expandable-label"
    id="modal_tags_label">
    <div class="save-dialog-expandable-
    label-gradient"></div>

```

```

Tags <span
class="save-dialog-expandable-
content-preview"></span>
style="display:none;">
name="modal_artwork_tags" id="modal_artwork_tags"
hint" style="display:none;">Tag length should be
class="form-group">
id="modal_square_grid_special_properties"><i class="icon icon-caret-right"></i>
Image rendering properties</label>
frame" style="display:none;">
name="modal_artwork_grid_visible" id="modal_artwork_grid_visible"
/> Visible grid
name="modal_artwork_pixel_art" id="modal_artwork_pixel_art"
/> Pixel-art image
name="modal_transparent_background" id="modal_transparent_background"
/> Transparent background
id="modal_success_action" />
</form>
</div>
<div class="modal-footer">

```



```

<button type="button" class="btn" data-
dismiss="modal">Close</button>
primary">Save artwork</button>
</div>
</div>
</div>
</div>
<div id="copy-paste-message-modal" class="modal fade" role="dialog">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<button type="button" class="close" data-dismiss="modal" aria-
hidden="true">&times;</button>
<h4 class="modal-title">Information</h4>
</div>
<div class="modal-body">
<p>
Copy/Paste mode sequence:
</p>
<ol>
<li>Push button 'Mark area' to start selecting cells;</li>
<li>Select cells on grid by pressing left mouse button on
them;</li>
<li>Unpush button 'Mark area' to fix your selection;</li>
<li>Push button 'Paste' and press left mouse button in
place, where you want to paste your
selection.</li>
</ol>
<p>
You can copy cells in one artwork and copy them to
another artwork.
</p>
</div>
<div class="modal-footer">
<label style="float:left;" class="text-muted"><input
type="checkbox"
name="chk-dont-show-copy-paste-message"
id="chk-dont-show-copy-paste-message"> Don't show
this message again</label>
<button type="button" class="btn" data-
dismiss="modal">Close</button>
</div>
</div>
</div>
</div>
<div id="message-modal" class="modal fade" role="dialog">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<button type="button" class="close" data-dismiss="modal" aria-
hidden="true">&times;</button>
<h4 class="modal-title">Warning</h4>
</div>

```



```

import javafx.scene.text.TextAlignment;

import javax.imageio.ImageIO;

public class AppController implements Initializable {

    public ScrollPane haxaScrollPane;
    public ScrollPane rectScrollPane;
    public Button chooseFile;
    public Slider lambdaSlider;
    public Rectangle precisionLevelOne;
    public Rectangle precisionLevelTwo;
    public Rectangle precisionLevelThree;
    public ScrollPane previewPane;
    public Text messageOnScreen;
    public ToggleButton langUABtn;
    public ToggleButton langENGBtn;
    public Text previewText;
    public Button readMeButton;
    public Button precisionButton;
    public Button startButton;
    public Text sliderText;

    public void onStartButtonClick(){
        rectScrollPane.setContent(null);
        haxaScrollPane.setContent(null);
        saveAsPng(previewPane, "img-preview");

        Application.image = new Image(new File("img-preview.png").toURI().toString());
        Application.imageWidth = (int) Application.image.getWidth() - 15;
        Application.imageHeight = (int) Application.image.getHeight() - 15;
        Application.setDefaultValue();

        execute();
    }

    public void onChooseFileButtonClick(){
        Application.setDefaultValue();
        Application.file =
Application.fileChooser.showOpenDialog(Application.pane.getScene().getWindow());
        if (Application.file != null){
            messageOnScreen.setVisible(false);
            System.out.println(Application.file.getPath());

            Application.image = new Image(new File(Application.file.getPath()).toURI().toString());
            Application.imageWidth = (int) Application.image.getWidth();
            Application.imageHeight = (int) Application.image.getHeight();
            previewPane.setContent(new ImageView(Application.image));
        }
        if (Application.image == null)
            messageOnScreen.setVisible(true);
    }

    public void onLambdaSliderAction(){

```

```

        Application.lambda = lambdaSlider.getValue()

hexaScrollPane.setContent(new Hexagon(lambdaSlider.getValue()).getPolygon());
    }

    public void onPrecisionButtonClick(){

        if (precisionLevelThree.getOpacity() == 1 ){
            precisionLevelThree.setOpacity(0);
            precisionLevelTwo.setOpacity(0);
            lambdaSlider.setMin(0.2);
            lambdaSlider.setMax(1);
            lambdaSlider.setMajorTickUnit(0.2);
            lambdaSlider.setMinorTickCount(1);
            lambdaSlider.setValue(0.2);
        }
        else if (precisionLevelOne.getOpacity() == 1 &&
            precisionLevelTwo.getOpacity() != 1){
            precisionLevelTwo.setOpacity(1);
            lambdaSlider.setMin(1);
            lambdaSlider.setMax(10);
            lambdaSlider.setMajorTickUnit(2);
            lambdaSlider.setMinorTickCount(1);
            lambdaSlider.setValue(1);
        }
        else if (precisionLevelTwo.getOpacity() == 1){
            precisionLevelThree.setOpacity(1);
            lambdaSlider.setMin(10);
            lambdaSlider.setMax(20);
            lambdaSlider.setMajorTickUnit(2);
            lambdaSlider.setMinorTickCount(1);
            lambdaSlider.setValue(10);
        }
    }

    public void saveAsPng(Node node, String fname) {
        saveAsPng(node, fname, new SnapshotParameters());
    }

    public void saveAsPng(Node node, String fname, SnapshotParameters ssp) {
        WritableImage image = node.snapshot(ssp, null);
        File file = new File(fname+".png");
        try {
            ImageIO.write(SwingFXUtils.fromFXImage(image, null), "png", file);
        } catch (IOException e) {
            System.out.println("Помилка збереження результату у форматі <png>");
        }
    }

    public void execute(){

        HBox hBox = new HBox();

        Application.drawRectangleRaster(Application.imageWidth, Application.imageHeight);
        saveAsPng(Application.rectangleSpace, "img-original");
        hBox.getChildren().add(new ImageView(new Image(new File("img-
original.png").toURI().toString())));
    }

```

```

        Application.setDefaultImage();
        rectScrollPane.setContent(new ImageView(new Image(new File("img-
original.png").toURI().toString())));

        Application.drawHexagonalRaster(Application.imageWidth, Application.imageHeight);
        saveAsPng(Application.workspace, "img-converted");
        hbox.getChildren().add(new ImageView(new Image(new File("img-
converted.png").toURI().toString())));
        Application.setDefaultImage();
        haxaScrollPane.setContent(new ImageView(new Image(new File("img-
converted.png").toURI().toString())));

        Application.finalImageResult.getChildren().add(hbox);

        saveAsPng(Application.finalImageResult, "img-compare");

        Application.setDefaultImage();
    }

    public void onHelpButtonClick() throws IOException, URISyntaxException {
        Desktop.getDesktop().browse(new
URI("https://github.com/Artem1018/imagConverterToHexagonalRaster/blob/main/README.md"));
    }

    public void onLangUAButtonClick(){
        langENGBtn.setSelected(false);
        langUABtn.setSelected(true);

        Application.getStage().setTitle("Конвертер зображень");

        previewText.setText("Попередній перегляд");
        precisionButton.setText("Точність");
        chooseFile.setText("Обрати зображення");
        startButton.setText("Пуск");
        sliderText.setText("Коефіцієнт розміру пікселя");
        readMeButton.setText("Довідка");
        messageOnScreen.setText("Оберіть зображення");
    }

    public void onLangENGButtonClick(){
        langUABtn.setSelected(false);
        langENGBtn.setSelected(true);

        Application.getStage().setTitle("Image converter");

        previewText.setText("Preview");
        precisionButton.setText("Accuracy");
        chooseFile.setText("Choose an image");
        startButton.setText("Start");
        sliderText.setText("Pixel size ratio");
        readMeButton.setText("Readme");

        messageOnScreen.setText("Select an image");
    }

```

```

    }

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        precisionLevelOne.setOpacity(1);
        precisionLevelTwo.setOpacity(1);
        precisionLevelThree.setOpacity(0);
    }
}

package com.example.imagconvertertohexagonalraster;

import javafx.scene.paint.Color;
import javafx.scene.shape.Polygon;

public class Rectangle {
    private final double[] points;
    private final Polygon polygon;

    Rectangle(double lambda, int shiftX, int shiftY, int cordX, int cordY){

        points = new double[]{
            10 / lambda + shiftX, 5 / lambda + shiftY,
            30 / lambda + shiftX, 5 / lambda + shiftY,    //.....
            30 / lambda + shiftX, 25 / lambda + shiftY,   //.....
            10 / lambda + shiftX, 25 / lambda + shiftY};

        polygon = new Polygon(points);
        polygon.setStroke(Color.TRANSPARENT);
        polygon.setFill(Color.TRANSPARENT);
        Application.rectangleSpace.getChildren().add(polygon);

    }
    protected int getRectangleShift(){

        return (int) (this.points[4] - this.points[0]);
    }
    public Polygon getPolygon() {
        return polygon;
    }
}

package com.example.imagconvertertohexagonalraster;

import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import java.io.File;
import java.io.IOException;

```

```

public class Application extends javafx.application.Application {
    private static final FXMLLoader fxmlloader = new
FXMLLoder(Application.class.getResource("mainScreen.fxml"));
    protected static Pane pane;
    protected static Pane workspace = new Pane();
    protected static Pane rectangleSpace = new Pane();
    protected static Pane finalImageResult = new Pane();

    protected static Image image;
    protected static final FileChooser fileChooser = new FileChooser();
    protected static File file;

    protected static double lambda = 4;
    protected static int imageWidth;
    protected static int imageHeight;

    private final Scene scene = new Scene(pane);

    private static Stage stage;
    public static Stage getStage() { return stage; }

    static {
        try {
            pane = fxmlloader.load();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void start(Stage primaryStage){
        stage = primaryStage;
        stage.setTitle("Конвертер зображень");
        stage.setScene(scene);
        stage.show();
        stage.setMinWidth(1200);
        stage.setMinHeight(440);
    }

    public static void drawHexagonalRaster(int width, int height){
        int shiftX = (int) (10/lambda);
        int shiftY = 0;
        Hexagon hexagon;

        for (int i = 1; i < height; i++) {
            for (int j = 1; j < width; j++) {
                hexagon = new Hexagon(lambda, shiftX, shiftY, j, i);

                Color aRGB = image.getPixelReader().getColor(j,i);

                hexagon.getPolygon().setFill(aRGB);

                shiftX += hexagon.getHexagonWidthForShift();
                if (j + 1 == width)
                    shiftY += hexagon.getHexagonHeightForShift();
            }
        }
    }
}

```

```

    }
    if (i%2 == 0)
        shiftX = (int) (10/lambda);
    else
        shiftX = 0;
    }
}

public static void drawRectangleRaster(int width, int height){
    int shiftX = 0;
    int shiftY = 0;
    Rectangle rectangle;

    for (int i = 1; i < height; i++) {
        for (int j = 1; j < width; j++) {
            rectangle = new Rectangle(lambda, shiftX, shiftY, j, i);
            Color aRGB = image.getPixelReader().getColor(j,i);

            rectangle.getPolygon().setFill(aRGB);
            shiftX += rectangle.getRectangleShift();
            if (j + 1 == width)
                shiftY += rectangle.getRectangleShift();
        }
        shiftX = 0;
    }
}

public static void setDefaultValue(){
    workspace.getChildren().clear();
    rectangleSpace.getChildren().clear();
    finalImageResult.getChildren().clear();
}

public static void main(String[] args) {
    launch();
}
}

package com.example.imagconvertertohexagonalraster;

import javafx.embed.swing.SwingFXUtils;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.SnapshotParameters;
import javafx.scene.control.*;
import javafx.scene.control.Button;
import javafx.scene.control.ScrollPane;
import javafx.scene.image.Image;

import java.awt.*;
import java.io.File;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;

```



```

import java.net.URL;
import java.util.ResourceBundle;

import javafx.scene.image.ImageView;
import javafx.scene.image.WritableImage;
import javafx.scene.layout.HBox;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Text;
import javafx.scene.text.TextAlignment;

import javax.imageio.ImageIO;

public class AppController implements Initializable {

    public ScrollPane haxaScrollPane;
    public ScrollPane rectScrollPane;
    public Button chooseFile;
    public Slider lambdaSlider;
    public Rectangle precisionLevelOne;
    public Rectangle precisionLevelTwo;
    public Rectangle precisionLevelThree;
    public ScrollPane previewPane;
    public Text messageOnScreen;
    public ToggleButton langUABtn;
    public ToggleButton langENGBtn;
    public Text previewText;
    public Button readMeButton;
    public Button precisionButton;
    public Button startButton;
    public Text sliderText;

    public void onStartButtonClick(){
        rectScrollPane.setContent(null);
        haxaScrollPane.setContent(null);
        saveAsPng(previewPane, "img-preview");

        Application.image = new Image(new File("img-preview.png").toURI().toString());
        Application.imageWidth = (int) Application.image.getWidth() - 15;
        Application.imageHeight = (int) Application.image.getHeight() - 15;
        Application.setDefaultLookAndFeelDecorated(true);

        execute();
    }

    public void onChooseFileButtonClick(){
        Application.setDefaultLookAndFeelDecorated(true);
        Application.file =
Application.fileChooser.showOpenDialog(Application.pane.getScene().getWindow());
        if (Application.file != null){
            messageOnScreen.setVisible(false);
            System.out.println(Application.file.getPath());

            Application.image = new Image(new File(Application.file.getPath()).toURI().toString());
            Application.imageWidth = (int) Application.image.getWidth();
            Application.imageHeight = (int) Application.image.getHeight();

```

```

        previewPane.setContent(new ImageView(Application.image));
    }
    if (Application.image == null)
        messageOnScreen.setVisible(true);
}

public void onLambdaSliderAction(){
    Application.lambda = lambdaSlider.getValue()
}

hexaScrollPane.setContent(new Hexagon(lambdaSlider.getValue()).getPolygon());
}

public void onPrecisionButtonClick(){

    if (precisionLevelThree.getOpacity() == 1 ){
        precisionLevelThree.setOpacity(0);
        precisionLevelTwo.setOpacity(0);
        lambdaSlider.setMin(0.2);
        lambdaSlider.setMax(1);
        lambdaSlider.setMajorTickUnit(0.2);
        lambdaSlider.setMinorTickCount(1);
        lambdaSlider.setValue(0.2);
    }
    else if (precisionLevelOne.getOpacity() == 1 &&
        precisionLevelTwo.getOpacity() != 1){
        precisionLevelTwo.setOpacity(1);
        lambdaSlider.setMin(1);
        lambdaSlider.setMax(10);
        lambdaSlider.setMajorTickUnit(2);
        lambdaSlider.setMinorTickCount(1);
        lambdaSlider.setValue(1);
    }
    else if (precisionLevelTwo.getOpacity() == 1){
        precisionLevelThree.setOpacity(1);
        lambdaSlider.setMin(10);
        lambdaSlider.setMax(20);
        lambdaSlider.setMajorTickUnit(2);
        lambdaSlider.setMinorTickCount(1);
        lambdaSlider.setValue(10);
    }
}

public void saveAsPng(Node node, String fname) {
    saveAsPng(node, fname, new SnapshotParameters());
}

public void saveAsPng(Node node, String fname, SnapshotParameters ssp) {
    WritableImage image = node.snapshot(ssp, null);
    File file = new File(fname+".png");
    try {
        ImageIO.write(SwingFXUtils.fromFXImage(image, null), "png", file);
    } catch (IOException e) {
        System.out.println("Помилка збереження результату у форматі <png>");
    }
}
}

```

```

public void execute(){

    HBox hBox = new HBox();

    Application.drawRectangleRaster(Application.imageWidth, Application.imageHeight);
    saveAsPng(Application.rectangleSpace, "img-original");
    hBox.getChildren().add(new ImageView(new Image(new File("img-
original.png").toURI().toString())));
    Application.setDefaultValue();
    rectScrollPane.setContent(new ImageView(new Image(new File("img-
original.png").toURI().toString())));

    Application.drawHexagonalRaster(Application.imageWidth, Application.imageHeight);
    saveAsPng(Application.workSpace, "img-converted");
    hBox.getChildren().add(new ImageView(new Image(new File("img-
converted.png").toURI().toString())));
    Application.setDefaultValue();
    haxScrollPane.setContent(new ImageView(new Image(new File("img-
converted.png").toURI().toString())));

    Application.finalImageResult.getChildren().add(hBox);

    saveAsPng(Application.finalImageResult, "img-compare");

    Application.setDefaultValue();
}

public void onHelpButtonClick() throws IOException, URISyntaxException {
    Desktop.getDesktop().browse(new
URI("https://github.com/Artem1018/imagConverterToHexagonalRaster/blob/main/README.md"));
}

public void onLangUAButtonClick(){
    langENGBtn.setSelected(false);
    langUABtn.setSelected(true);

    Application.getStage().setTitle("Конвертер зображень");

    previewText.setText("Попередній перегляд");
    precisionButton.setText("Точність");
    chooseFile.setText("Обрати зображення");
    startButton.setText("Пуск");
    sliderText.setText("Коефіцієнт розміру пікселя");
    readMeButton.setText("Довідка");
    messageOnScreen.setText("Оберіть зображення");
}

public void onLangENGButtonClick(){
    langUABtn.setSelected(false);
    langENGBtn.setSelected(true);

    Application.getStage().setTitle("Image converter");

    previewText.setText("Preview");

```

```

precisionButton.setText("Accuracy");
chooseFile.setText("Choose an image");
startButton.setText("Start");
sliderText.setText("Pixel size ratio");
readMeButton.setText("Readme");

messageOnScreen.setText("Select an image");

}

@Override
public void initialize(URL url, ResourceBundle resourceBundle) {
    precisionLevelOne.setOpacity(1);
    precisionLevelTwo.setOpacity(1);
    precisionLevelThree.setOpacity(0);
}
}

package com.example.imagconvertertohexagonalraster;

import javafx.scene.paint.Color;
import javafx.scene.shape.Polygon;

public class Rectangle {
    private final double[] points;
    private final Polygon polygon;

    Rectangle(double lambda, int shiftX, int shiftY, int cordX, int cordY){

        points = new double[]{
            10 / lambda + shiftX, 5 / lambda + shiftY,
            30 / lambda + shiftX, 5 / lambda + shiftY, //.....
            30 / lambda + shiftX, 25 / lambda + shiftY, //.....
            10 / lambda + shiftX, 25 / lambda + shiftY};

        polygon = new Polygon(points);
        polygon.setStroke(Color.TRANSPARENT);
        polygon.setFill(Color.TRANSPARENT);
        Application.rectangleSpace.getChildren().add(polygon);

    }
    protected int getRectangleShift(){

        return (int) (this.points[4] - this.points[0]);
    }
    public Polygon getPolygon() {
        return polygon;
    }
}

package com.example.imagconvertertohexagonalraster;

import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.image.Image;

```

```

import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import java.io.File;
import java.io.IOException;

public class Application extends javafx.application.Application {
    private static final FXMLLoader fxmlLoader = new
FXMLLoader(Application.class.getResource("mainScreen.fxml"));
    protected static Pane pane;
    protected static Pane workSpace = new Pane();
    protected static Pane rectangleSpace = new Pane();
    protected static Pane finalImageResult = new Pane();

    protected static Image image;
    protected static final FileChooser fileChooser = new FileChooser();
    protected static File file;

    protected static double lambda = 4;
    protected static int imageWidth;
    protected static int imageHeight;

    private final Scene scene = new Scene(pane);

    private static Stage stage;
    public static Stage getStage() { return stage; }

    static {
        try {
            pane = fxmlLoader.load();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void start(Stage primaryStage){
        stage = primaryStage;
        stage.setTitle("Конвертер изображень");
        stage.setScene(scene);
        stage.show();
        stage.setMinWidth(1200);
        stage.setMinHeight(440);

    }

    public static void drawHexagonalRaster(int width, int height){
        int shiftX = (int) (10/lambda);
        int shiftY = 0;
        Hexagon hexagon;

        for (int i = 1; i < height; i++) {
            for (int j = 1; j < width; j++) {
                hexagon = new Hexagon(lambda, shiftX, shiftY, j, i);
            }
        }
    }
}

```

```

        Color aRGB = image.getPixelReader().getColor(j,i);

        hexagon.getPolygon().setFill(aRGB);

        shiftX += hexagon.getHexagonWidthForShift();
        if (j + 1 == width)
            shiftY += hexagon.getHexagonHeightForShift();
    }
    if (i%2 == 0)
        shiftX = (int) (10/lambda);
    else
        shiftX = 0;
}
}

public static void drawRectangleRaster(int width, int height){
    int shiftX = 0;
    int shiftY = 0;
    Rectangle rectangle;

    for (int i = 1; i < height; i++) {
        for (int j = 1; j < width; j++) {
            rectangle = new Rectangle(lambda, shiftX, shiftY, j, i);
            Color aRGB = image.getPixelReader().getColor(j,i);

            rectangle.getPolygon().setFill(aRGB);
            shiftX += rectangle.getRectangleShift();
            if (j + 1 == width)
                shiftY += rectangle.getRectangleShift();
        }
        shiftX = 0;
    }
}

public static void setDefaultValue(){
    workspace.getChildren().clear();
    rectangleSpace.getChildren().clear();
    finalImageResult.getChildren().clear();
}

public static void main(String[] args) {
    launch();
}
}

```

### **Лістинг програми формування графічних примітивів і їх антиаліазингу**

```

module com.example.practicalworkscomputergraphics {
    requires javafx.controls;
    requires javafx.fxml;
    requires javafx.web;

    requires org.controlsfx.controls;
    requires com.dlsc.formsfx;
    requires validatorfx;
    requires org.kordamp.ikonli.javafx;

```

```

requires org.kordamp.bootstrapfx.core;
requires eu.hansolo.tilesfx;

opens com.example.Hexa_project to javafx.fxml;

exports com.example.Hexa_project.Another;
opens com.example.Hexa_project.Another to javafx.fxml;
exports com.example.Hexa_project;
}
    package com.example.Hexa_project;

import javafx.scene.paint.Color;
import javafx.scene.shape.Polygon;

public class Hex {
    private int x;
    private int y;
    private double _x;
    private double posX;
    private double posY;
    private static double height = 30 ;
    private static double width = height*(double) (3.0/(2.0*Math.sqrt(3.0)));

    public static void setHeight(double height) {
        Hex.height = height;
        Hex.width = height*(double) (3.0/(2.0*Math.sqrt(3.0)));
    }

    private double points [];
    private Polygon polygon;

    public double[] getPoints() {
        return points;
    }

    public static double getWidth(){
        return width;
    }
    public static double getHeight(){
        return height;
    }

    public static double strokeWidth = 1;
    public boolean filled = false;
    private static Color filledColor = Color.RED;
    private static Color fillColor = Color.WHITE;
    private static Color strokeColor = Color.BLACK;

    public static void setFilledColor(Color _filledColor) {
        filledColor = _filledColor;
    }
    public static Color getFilledColor() {
        return filledColor;
    }
}

    public static void setFillColor(Color _fillColor) {
        fillColor = _fillColor;
    }
}

```

```

}

public static void setStrokeColor(Color _strokeColor) {
    strokeColor = _strokeColor;
}

public Hex(int x_, int y_){
    this.x = x_;
    this.y = y_;
    this._x= x_;

    double posX = x * width ;
    double posY = y * height * 0.75 ;

    if(y%2 == 1){
        _x += 0.5;
        posX += width/2;
    }
    points = new double[]{
        posX,      posY - height/2,
        posX + width/2, posY - height/4,
        posX + width/2, posY + height/4,
        posX,      posY + height/2,
        posX - width/2, posY + height/4,
        posX - width/2, posY - height/4,
    };
    print();
}

public Hex(double x, double y){
    this.posx = x ;
    this.posy = y ;

    double posX = x;
    double posY = y;

    if(y%2 == 1){
        //posX += width/2;
    }
    points = new double[]{
        posX,      posY - height/2,
        posX + width/2, posY - height/4,
        posX + width/2, posY + height/4,
        posX,      posY + height/2,
        posX - width/2, posY + height/4,
        posX - width/2, posY - height/4,
    };
}

public void print() {
    polygon = new Polygon(points);
    polygon.setStroke(strokeColor);
    polygon.setStrokeWidth(strokeWidth);
    if(filled){
        polygon.setFill(filledColor);
    }
}

```



```

else {
    polygon.setFill(fillColor);
}

App.workSpace.getChildren().add(polygon);
}
}

    public void interpolationHexagonCircle() {
double tabX = (R + 3) * Hex.getWidth();
double tabY = (R + 3) * Hex.getHeight() * 0.75;
if (R % 2 == 0) {
    tabY = (R + 40) * Hex.getHeight() * 0.75;
}
double x = R * Hex.getWidth();
double y = 0;

//    Hex h = new Hex(tabX, tabY);
//    h.filled = true;
//    h.print();

double OF = 0;

while (//y <= pi/6 * Hex.getHeight()*0.75){
    Math.sqrt(x * x + y * y) / 2 > y) { // 0° до 30°

//    h = new Hex(x + tabX, y + tabY);
//    h.filled = true;
//    h.print();
//    h = new Hex(x + tabX, -y + tabY);
//    h.filled = true;
//    h.print();
//    h = new Hex(-x + tabX, -y + tabY);
//    h.filled = true;
//    h.print();
//    h = new Hex(-x + tabX, y + tabY);
//    h.filled = true;
//    h.print();

//    System.out.print("U(" + x + "," + y + ") ");
//    System.out.printf("%3f", OF);

double OFdL = Math.abs((x - 0.5 * Hex.getWidth()) * (x - 0.5 * Hex.getWidth())
    + (y + Hex.getHeight() - Hex.getHeight() / 4) * (y + Hex.getHeight() - Hex.getHeight() / 4)
    - ((double) (R * R) * Hex.getWidth() * Hex.getWidth()));

//double OFdL = Math.pow( Math.pow(x -0.5 ,2) + Math.pow(y + (3.0/(2.0*Math.sqrt(3.0))),2)
,2)- ( (double) R * Hex.getWidth() * (double)R * Hex.getWidth());
//OF - x + Math.sqrt(3) * y + 1;
double OFdR = Math.abs((x + 0.5 * Hex.getWidth()) * (x + 0.5 * Hex.getWidth())
    + (y + Hex.getHeight() - Hex.getHeight() / 4) * (y + Hex.getHeight() - Hex.getHeight() / 4)
    - ((double) (R * R) * Hex.getWidth() * Hex.getWidth()));

```

```

if (OFdR < OFdL) {
    OF = OFdR;
    x += Hex.getWidth() / 2;
    y += Hex.getHeight() - Hex.getHeight() / 4;

    currentStep = "діагональний праворуч";
    printlnTextField(currentStep);
//    System.out.print(currentStep);
//    currentStep = " ↗ ";
} else {
    OF = OFdL;
    x -= Hex.getWidth() / 2;
    y += Hex.getHeight() - Hex.getHeight() / 4;

    currentStep = "діагональний ліворуч";
    printlnTextField(currentStep);
//    System.out.print(currentStep);
//    currentStep = " ↖ ";
}
if (isSecondStep) {
    registerStepsService.addStep(previousStep + " + " + currentStep + " | 0° до 30°");
} else {
    previousStep = currentStep;
}
isSecondStep = !isSecondStep;
}

while (x >= 0 - Hex.getWidth() / 2) {

double OFdL = Math.abs(
    (x - 0.5 * Hex.getWidth()) * (x - 0.5 * Hex.getWidth())
    + (y + Hex.getHeight() - Hex.getHeight() / 4) * (y + Hex.getHeight() - Hex.getHeight() /
4)
    - ((double) (R * R) * Hex.getWidth() * Hex.getWidth()));

double OFhL = Math.abs(
    (x - 1 * Hex.getWidth()) * (x - 1 * Hex.getWidth())
    + (y) * (y)
    - ((double) (R * R) * Hex.getWidth() * Hex.getWidth()));
//Math.abs( Math.pow( Math.pow(x - 1.0 ,2) + Math.pow(y,2) ,2)- R*R);
//OF -2 * x + 1;

//    System.out.println(" " + OFhL + " " + OFdL);
if (OFhL < OFdL) {
    OF = OFhL;
    x -= Hex.getWidth();
    currentStep = "горизонтальний ліворуч";
    printlnTextField("горизонтальний ліворуч P/6_P/2 | 30° до 90°");
//    System.out.print(" горизонтальний ліворуч P/6_P/2 | 30° до 90°");
//    currentStep = " ⇐ ";
} else {
    OF = OFdL;
    x -= Hex.getWidth() / 2;
    y += Hex.getHeight() - Hex.getHeight() / 4;

    currentStep = "діагональний ліворуч";

```

```

        printlnTextField("діагонально вліво P/6_P/2 | 30° до 90°");
//      System.out.print(" діагонально вліво P/6_P/2 | 30° до 90°");
//      currentStep = "↖ ";
    }
    if (isSecondStep) {
        registerStepsService.addStep(previousStep + " + " + currentStep + " | 30° до 90°");
    } else {
        previousStep = currentStep;
    }
    isSecondStep = !isSecondStep;
}
}

```

```
package com.example.Hexa_project;
```

```

import java.util.*;
import java.util.stream.Collectors;

public class RegisterStepsService {

    private int startWithR = 2000;
    private int endWithR = 4000;

    private int currentR = startWithR;

    public Map<Integer, List<Step>> stepsStatisticMap
        = new LinkedHashMap<>(Collections.singletonMap(currentR, new ArrayList<>()));

    public RegisterStepsService() {
    }

    public void addStep(String stepName) {
        List<Step> stepList = new ArrayList<>(stepsStatisticMap.get(currentR));
        if (!stepList.isEmpty()) {
            List<Step> fileteredStepList = stepList.stream()
                .filter(step -> stepName.equals(step.name))
                .toList();
            if (!fileteredStepList.isEmpty()) {
                fileteredStepList.forEach(Step::incrementCounter);
            } else {
                stepList.add(new Step(stepName, 1));
            }
            stepsStatisticMap.put(currentR, stepList);
        } else {
            stepsStatisticMap.put(currentR, Collections.singletonList(new Step(stepName, 1)));
        }
    }

    public void printResult() {
        System.out.println("Result:");
        for (Integer radius : stepsStatisticMap.keySet()) {
            printResultByRadius(radius);
        }
    }

    public void printResultByRadius(Integer radius) {
        System.out.println();
    }
}

```

```

System.out.format("R=%4d |", radius);

System.out.println("=====
=====");
String firstSector = "0° до 30°";
String secondSector = "30° до 90°";
extracted(radius, firstSector);
System.out.println("-----
-----");
extracted(radius, secondSector);
}

private void extracted(Integer radius, String sectorName) {
    List<Step> stepListBySector = new ArrayList<>(
        stepsStatisticMap.get(radius).stream()
            .filter(o -> o.name.contains(sectorName))
            .toList()
    );
    stepListBySector.sort(Comparator.comparing(Step::getName));
    int allStepsBySector = stepListBySector.stream().mapToInt(Step::getCounter).sum();
    for (Step step : stepListBySector) {
        double perCent = ((double) step.counter / (double) allStepsBySector * 100D);
        System.out.format(" | крок: %60s | рахунок кроків: %3d | у відсотках: %2f%%", step.name,
step.counter, perCent);
        System.out.println();
    }
}

public int getStartWithR() {
    return startWithR;
}

public int getEndWithR() {
    return endWithR;
}

public int getCurrentR() {
    return currentR;
}

public void setCurrentR(int currentR) {
    this.currentR = currentR;
    stepsStatisticMap.put(currentR, new ArrayList<>());
}

private static class Step {

    String name;
    int counter;

    public Step(String name, int counter) {
        this.name = name;
        this.counter = counter;
    }

    public void incrementCounter() {
        counter++;
    }
}

```

```

    }

    public int getCounter() {
        return counter;
    }

    public String getName() {
        return name;
    }
}

}

package com.example.Hexa_project;
import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.control.SplitPane;
import javafx.scene.image.Image;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.Pane;
import javafx.scene.shape.Line;
import javafx.stage.Stage;
import java.io.IOException;

import static com.example.Hexa_project.HexConrtoller.antialiasing;
import static com.example.Hexa_project.HexConrtoller.clearFilledHexagon;

public class App extends Application {

    private static final FXMLLoader fxmlLoader = new
FXMLLoader(App.class.getResource("HexagonLineInterpolation.fxml"));
    private static SplitPane pane;
    static Pane workSpace = new Pane();
    private final Scene scene = new Scene(pane);

    public static boolean isline = true;
    public static boolean isAddline = true;
    public static boolean isInterpolation = true;
    public static boolean isAntialiasing = false;
    public static boolean isAntialiasingPro = false;

    static {
        try {
            pane = fxmlLoader.load();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        launch(args);
    }

    @Override

```

```

public void start(Stage primaryStage) {
    primaryStage.getIcons().add(new Image("hexLogo.png"));
    primaryStage.setTitle("Hexagon Screen");
    primaryStage.setScene(scene);

    primaryStage.show();
    primaryStage.setResizable(false);
    workspace.setLayoutX(20);
    workspace.setLayoutY(20);

    workspace.setOnMousePressed (new EventHandler<MouseEvent>() {
        @Override public void handle(MouseEvent event) {
            if(isline){
                System.out.println(
                    "(x: " + event.getX() + ", y: " + event.getY() + ")");

                HexConrtoller.dX = event.getX();
                HexConrtoller.dY = event.getY();
                clearFilledHexagon();
                if(isInterpolation){
                    HexConrtoller.interpolationHexagonLine();
                }
                else if(isAntialiasing){
                    HexConrtoller.antialiasing();
                }
                else if(isAntialiasingPro){
                    HexConrtoller.antialiasingPro();
                }
                //

                if(isAddline){
                    Line line= new Line(
                        0.0,
                        0.0,
                        event.getX(),
                        event.getY());
                    App.workspace.getChildren().add(line);
                }
            }
            //reporter.setText(msg);
        }
    });
    pane.getItems().add(workspace);
}

package com.example.Hexa_project;

import javafx.scene.paint.Color;
import javafx.scene.shape.Polygon;

public class Hex {
    private int x;
    private int y;
    private double _x;
    private double posx;
    private double posy;
}

```

```

private static double height = 30 ;
private static double width = height*(double) (3.0/(2.0*Math.sqrt(3.0)));

public static void setHeight(double height) {
    Hex.height = height;
    Hex.width = height*(double) (3.0/(2.0*Math.sqrt(3.0)));
}

private double points [];
private Polygon polygon;

public double[] getPoints() {
    return points;
}

public static double getWidth(){
    return width;
}
public static double getHeight(){
    return height;
}

public static double strokeWidth = 1;
public boolean filled = false;
private static Color filledColor = Color.RED;
private static Color fillColor = Color.WHITE;
private static Color strokeColor = Color.BLACK;

public static void setFilledColor(Color _filledColor) {
    filledColor = _filledColor;
}
public static Color getFilledColor() {
    return filledColor;
}

public static void setFillColor(Color _fillColor) {
    fillColor = _fillColor;
}

public static void setStrokeColor(Color _strokeColor) {
    strokeColor = _strokeColor;
}

public Hex(int x_, int y_){
    this.x = x_;
    this.y = y_;
    this._x= x_;

    double posX = x * width ;
    double posY = y * height * 0.75 ;

    if(y%2 == 1){
        _x += 0.5;
        posX += width/2;
    }
    points = new double[]{
        posX,      posY - height/2,

```

```

        posX + width/2, posY - height/4,
        posX + width/2, posY + height/4,
        posX,        posY + height/2,
        posX - width/2, posY + height/4,
        posX - width/2, posY - height/4,
    };
    print();
}
public Hex(double x, double y){
    this.posx = x ;
    this.posy = y ;

    double posX = x;
    double posY = y;

    if(y%2 == 1){
        //posX += width/2;
    }
    points = new double[]{
        posX,        posY - height/2,
        posX + width/2, posY - height/4,
        posX + width/2, posY + height/4,
        posX,        posY + height/2,
        posX - width/2, posY + height/4,
        posX - width/2, posY - height/4,
    };
}

public void print() {
    polygon = new Polygon(points);
    polygon.setStroke(strokeColor);
    polygon.setStrokeWidth(strokeWidth);
    if(filled){
        polygon.setFill(filledColor);
    }
    else {
        polygon.setFill(fillColor);
    }

    App.workspace.getChildren().add(polygon);
}
}

package com.example.Hexa_project;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;

import java.net.URL;
import java.util.ResourceBundle;

```



```

public class HexConrtoller implements Initializable {
    private double hexagonSize = 0;
    private static Color filledColor = Color.RED;
    private Color emptyColor = Color.WHITE;
    private Color strokeColor = Color.BLACK;

    public static double dX = 1000;
    public static double dY = 1000;

    private static int R = 5;

    static double screenSize = 1100;
    private static int columns = (int) (screenSize/Hex.getWidth());
    private static int lines = (int) (screenSize/Hex.getWidth());
    private static Hex[][] fieldHexagon;

    @FXML
    private Button refreshButton;
    @FXML
    private ColorPicker choserColorFilled;
    @FXML
    private ColorPicker choserColorEmpty;
    @FXML
    private ColorPicker choserColorStroke;
    @FXML
    private Slider sizeSlider;
    @FXML
    private Slider sizeSliderRadius;
    @FXML
    private TextArea textAreaInformation;
    @FXML
    private VBox vBoxRadius;
    @FXML
    private CheckBox checkBoxAddLine;
    @FXML
    private ComboBox<String> chooser;

    @FXML
    private void colorFilled(){
        filledColor = choserColorFilled.getValue();
    }
    @FXML
    private void colorEmpty(){
        emptyColor = choserColorEmpty.getValue();
    }
    @FXML

    private void colorStroke(){
        strokeColor = choserColorStroke.getValue();
    }
}

```

```

@FXML
private void onRefreshButtonClick(){
    Hex.setHeight(sizeSlider.getValue());
    R = (int) sizeSliderRadius.getValue();
    App.isAddline = checkBoxAddLine.isSelected();
    columns = (int) (screenSize*1.4/Hex.getWidth());
    lines = (int) (screenSize*1.2/Hex.getWidth());
    Hex.setFilledColor(filledColor);
    Hex.setFillColor(emptyColor);
    Hex.setStrokeColor(strokeColor);
    if(filledColor == strokeColor){
        Hex.strokeWidth = 0;
    }
    else {
        Hex.strokeWidth = 1;
    }

    createNewScreen();
    if(chooser.getValue().equals("Антиаліайзінг 12 точок")){
        checkBoxAddLine.setVisible(true);
        vboxRadius.setDisable(true);
        App.isline = true;
        App.isAntialiasing = false;
        App.isInterpolation =false;
        App.isAntialiasingPro =true;
    }
    if(chooser.getValue().equals("Антиаліайзінг 7 точок")){
        checkBoxAddLine.setVisible(true);
        vboxRadius.setDisable(true);
        App.isline = true;
        App.isAntialiasing = true;
        App.isInterpolation =false;
        App.isAntialiasingPro =false;
    }
    else if(chooser.getValue().equals("Лінія")){
        checkBoxAddLine.setVisible(true);
        vboxRadius.setDisable(true);
        App.isline = true;
        App.isAntialiasing = false;
        App.isInterpolation =true;
        App.isAntialiasingPro =false;
        //interpolationHexagonLine();
    }
    else if(chooser.getValue().equals("Коло")){
        checkBoxAddLine.setVisible(false);
        vboxRadius.setDisable(false);
        App.isline = false;
        interpolationHexagonCircle();
        App.isAntialiasing = false;
        App.isInterpolation =false;
        App.isAntialiasingPro =false;
    }
    else if(chooser.getValue().equals("Коло Pro")){
        checkBoxAddLine.setVisible(false);
        vboxRadius.setDisable(false);
    }
}

```

```

    App.isline = false;
    interpolationHexagonCirclePro();
    App.isAntialiasing = false;
    App.isInterpolation = false;
    App.isAntialiasingPro = false;
}

}

void printInTextField(String s){
    textAreaInformation.setText(s + "\n" + textAreaInformation.getText());
}

public static void interpolationHexagonLine(){
    createNewScreen();

    double x = 0;
    double y = 0;

    double diff = 0;
    double c = Math.sqrt(dX*dX+dY*dY);
    boolean cosBilshe60 = false;
    if (0.5 >= (double) (dX) / c) {
        cosBilshe60 = true;
    }
    boolean prev = true;

    while (x <= dX && y <= dY) {
        if (cosBilshe60) {
            //diff = ((double)y + (0.75)) *(double)dX - ((double)x - 0.5)* (double)dY;
            diff = ((double) y + (double) (3.0 / (2.0 * Math.sqrt(3.0)))) * (double) dX - ((double) x - 0.5) *
(double) dY;
        } else {
            diff = ((double) y + (double) (3.0 / (2.0 * Math.sqrt(3.0)))) * (double) dX - ((double) x + 0.5) *
(double) dY;
            //diff = ((double)y + (0.75)) *(double)dX - ((double)x + 0.5) * (double)dY;
        }

        Hex h = new Hex(x,y);
        h.filled = true;
        h.print();

        System.out.print("U(" + x + "," + y + ") ");
        System.out.printf("%3f" , diff );
        //if(information== null){
        //HexagonLineInterpolation.information.setText("U(" + x + "," + y + ") OF = " + diff);
        //}
        //else {
        //HexagonLineInterpolation.information.setText("U(" + x + "," + y + ") OF = " + diff +
information.getText());
        //}

        if(diff >= 0.0){

```

```

    if(cosBilshe60){
        x+= Hex.getWidth()/2;;
        y+= Hex.getHeight() - Hex.getHeight()/4;

        System.out.print(" go XY cosBilshe60");
    }
    else {
        x+=Hex.getWidth();
        System.out.print(" go X");
    }
}
else{
    if(cosBilshe60){
        x-= Hex.getWidth()/2;
        y+= Hex.getHeight() - Hex.getHeight()/4;
        System.out.print(" go XY cosBilshe60");
    }
    else {
        prev = !prev;
        System.out.print(" go +XY");
        x+= Hex.getWidth()/2;
        y+= Hex.getHeight() - Hex.getHeight()/4;
    }
}

    System.out.println();
}

}

public void interpolationHexagonCircle(){
    double tabX = (R+3) * Hex.getWidth();
    double tabY = (R + 3) * Hex.getHeight()*0.75;
    if(R%2 == 0){
        tabY = (R + 40) * Hex.getHeight()*0.75;
    }
    double x = R * Hex.getWidth();
    double y = 0;
    double pi = 3.14159265;

    Hex h = new Hex(tabX,tabY);
    h.filled = true;
    h.print();

    double OF = 0;

    while (//y <= pi/6 * Hex.getHeight()*0.75){
        Math.sqrt(x*x+y*y)/2 > y ) { // 0° до 30°

            h = new Hex(x+tabX,y+tabY);
            h.filled = true;
            h.print();
            h = new Hex(x+tabX,-y+tabY);
            h.filled = true;
            h.print();
            h = new Hex(-x+tabX,-y+tabY);

```

```

h.filled = true;
h.print();

h = new Hex(-x+tabX,y+tabY);
h.filled = true;
h.print();

System.out.print("U("+ x + "," + y + ") ");
System.out.printf("%3F" , OF);

double OFdL = Math.abs ((x-0.5*Hex.getWidth()*(x-0.5*Hex.getWidth())
+ (y + Hex.getHeight() - Hex.getHeight()/4)*(y + Hex.getHeight() - Hex.getHeight()/4)
- ((double)(R*R) * Hex.getWidth()*Hex.getWidth()));

//double OFdL = Math.pow( Math.pow(x -0.5 ,2) + Math.pow(y + (3.0/(2.0*Math.sqrt(3.0))),2)
,2)- ( (double) R * Hex.getWidth() * (double)R * Hex.getWidth());
//OF - x + Math.sqrt(3) * y + 1;
double OFdR = Math.abs ((x+0.5*Hex.getWidth()*(x+0.5*Hex.getWidth())
+ (y + Hex.getHeight() - Hex.getHeight()/4)*(y + Hex.getHeight() - Hex.getHeight()/4)
- ((double)(R*R) * Hex.getWidth()*Hex.getWidth()));
//Math.pow( Math.pow(x + 0.5 ,2) + Math.pow(y + (3.0/(2.0*Math.sqrt(3.0))),2) ,2)-
((double)R * Hex.getWidth() *(double) R * Hex.getWidth());
//OF + x + Math.sqrt(3) * y + 1;
System.out.println(" "+ OFdL + " "+ OFdR);
//if(OFdL < 0 && OFdR < 0){
//
//
// x+= Hex.getWidth()/2;
// y+= Hex.getHeight() - Hex.getHeight()/4;
// System.out.print(" діагонально вправо 0_P/6 | 0° до 30°");
//}
//else if(OFdL < 0 && OFdR > 0){
// x+= Hex.getWidth()/2;
// y+= Hex.getHeight() - Hex.getHeight()/4;
// System.out.print(" діагонально вправо 0_P/6 | 0° до 30°");
//}
//else {
// x-= Hex.getWidth()/2;
// y+= Hex.getHeight() - Hex.getHeight()/4;
//
// System.out.print(" діагонально вліво 0_P/6 | 0° до 30°");
//}

if(OFdR < OFdL){
    OF = OFdR;
    x+= Hex.getWidth()/2;
    y+= Hex.getHeight() - Hex.getHeight()/4;

    printlnTextField("діагонально вправо 0_P/6 | 0° до 30°");
    System.out.print(" діагонально вправо 0_P/6 | 0° до 30°");
}
else {
    OF = OFdL;
    x-= Hex.getWidth()/2;
    y+= Hex.getHeight() - Hex.getHeight()/4;
}

```

```

        printlnTextField("діагонально вліво 0_P/6 | 0° до 30°");
        System.out.print(" діагонально вліво 0_P/6 | 0° до 30°");
    }

    System.out.println();
}

while (x >= 0- Hex.getWidth()/2){
    //x > 0}{ // 30° до 90°
    h = new Hex(x+tabX,y+tabY);
    h.filled = true;
    h.print();

    h = new Hex(x+tabX,-y+tabY);
    h.filled = true;
    h.print();
    h = new Hex(-x+tabX,-y+tabY);
    h.filled = true;
    h.print();
    h = new Hex(-x+tabX,y+tabY);
    h.filled = true;
    h.print();
    System.out.println();
    System.out.print("U(" + x + "," + y + ") ");
    System.out.printf("%3f" , OF);

    double OFdL = Math.abs (
        (x-0.5*Hex.getWidth())*(x-0.5*Hex.getWidth())
        + (y + Hex.getHeight() - Hex.getHeight()/4)*(y + Hex.getHeight() - Hex.getHeight()/4)
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth()));

    double OFhL = Math.abs (
        (x-1*Hex.getWidth())*(x-1*Hex.getWidth())
        + (y)*(y)
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth()));
    //Math.abs( Math.pow( Math.pow(x - 1.0 ,2) + Math.pow(y,2) ,2)- R*R);
    //OF -2 * x + 1;

    System.out.println(" "+ OFhL + " "+ OFdL);
    if(OFhL < OFdL){
        OF = OFhL;
        x-= Hex.getWidth();

        printlnTextField("горизонтальний ліворуч P/6_P/2 | 30° до 90°");
        System.out.print(" горизонтальний ліворуч P/6_P/2 | 30° до 90°");
    }
    else {
        OF = OFdL;
        x-= Hex.getWidth()/2;
        y+= Hex.getHeight() - Hex.getHeight()/4;

        printlnTextField("діагонально вліво P/6_P/2 | 30° до 90°");
        System.out.print(" діагонально вліво P/6_P/2 | 30° до 90°");
    }
}

```

```

    }
    public class HexTest {

        @Test
        public void testHexInitialization() {
            int x = 2;
            int y = 3;

            Hex hex = new Hex(x, y);

            double[] expectedPoints = {
                x * Hex.getWidth(), y * Hex.getHeight() * 0.75 - Hex.getHeight() / 2,
                x * Hex.getWidth() + Hex.getWidth() / 2, y * Hex.getHeight() * 0.75 - Hex.getHeight() /
4,
                x * Hex.getWidth() + Hex.getWidth() / 2, y * Hex.getHeight() * 0.75 + Hex.getHeight() /
4,
                x * Hex.getWidth(), y * Hex.getHeight() * 0.75 + Hex.getHeight() / 2,
                x * Hex.getWidth() - Hex.getWidth() / 2, y * Hex.getHeight() * 0.75 + Hex.getHeight() /
4,
                x * Hex.getWidth() - Hex.getWidth() / 2, y * Hex.getHeight() * 0.75 - Hex.getHeight() / 4
            };

            double[] actualPoints = hex.getPoints();

            assertEquals(expectedPoints, actualPoints, 0.0001);
        }

        @Test
        public void testSetFilledColor() {
            Color expectedColor = Color.BLUE;

            Hex.setFilledColor(expectedColor);
            Color actualColor = Hex.getFilledColor();

            assertEquals(expectedColor, actualColor);
        }

        @Test
        public void testSetFillColor() {
            Color expectedColor = Color.YELLOW;

            Hex.setFillColor(expectedColor);
            Color actualColor = Hex.getFillColor();

            assertEquals(expectedColor, actualColor);
        }

        @Test
        public void testSetStrokeColor() {
            Color expectedColor = Color.GREEN;

            Hex.setStrokeColor(expectedColor);
            Color actualColor = Hex.getStrokeColor();

            assertEquals(expectedColor, actualColor);
        }
    }
}

```