Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Методи та програмні засоби формування графічних зображень на гексогональному растрі»

Виконав: студент 2-го курсу, групи 1ПІ-21м спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)

Козубенко М. В.
(прізвище та ініціали)

Керівник: д.т.н., проф., завідувач каф. ПЗ

Романюк О. Н.
(прізвище та ініціали)

«14» 12 2022 р.

Опонент: к.т.н., доцент каф. ЗІ

Дудатьєв А. В.
(прізвище та ініціали)

«14» 12 2022 р.

Допущено до захисту

Завідувач кафедри ПЗ

д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)

«14» 12 2022 р.

Вінниця ВНТУ - 2022 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
« 16 » вересня 2022 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Козубенку Максиму Володимировичу

1. Тема роботи – розробка методів та програмних засобів формування графічних зображень на гексогональному растрі.

Керівник роботи: Романюк Олександр Никифорович, д.т.н., завідувач кафедри ПЗ, затверджені наказом вищого навчального закладу від « 15 » вересня 2022 р. № 205-А.

2. Строк подання студентом роботи- 9 грудня 2022 р.

3. Вихідні дані до роботи: Тип растру - гексагональний; тип графічних примітивів – відрізок прямої, коло; метод інтерполяції - метод оцінювальної функції; кольоровий режим – TrueColor.

4. Зміст текстової частини: вступ, аналіз використання та формування зображень на гексагональному растрі, методи та програмні засоби формування відрізків прямих на гексогональному растрі, метод та програмні засоби формування кола на гексогональному растрі, програмна реалізація методів і програмних засобів формування зображень на гексогональному растрі, економічна частина, висновки, список використаних джерел, додатки.

5. Перелік графічного матеріалу: галузі застосування методів і засобів формування гексагонального растру, програмні засоби для формування лінії та кола на гексагональному растрі.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Романюк О. Н., д.т.н., завідувач кафедри ПЗ	16.09.22	16.12.22
5	Глущенко Л. Д., к.е.н., доцент кафедри ЕПВМ	28.11.22	16.12.22

7. Дата видачі завдання 16 вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз використання та формування зображень на гексагональному растрі	18.09.2022-27.09.2022	Вик.
2	Методи та програмні засоби формування відрізків прямих на гексогональному растрі	28.09.2022-15.10.2022	Вик.
3	Метод та програмні засоби формування кола на гексогональному растрі	16.10.2022-7.11.2022	Вик.
4	Програмна реалізація методів і програмних засобів формування зображень на гексогональному растрі	8.11.2022-27.11.2022	Вик.
5.	Економічна частина	28.11.2022-10.12.2022	Вик.

Студент _____ **Козубенко М. В.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Романюк О. Н.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.5

Козубенко М. В. Методи та програмні засоби формування графічних зображень на гексогональному растрі. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2022. 274 с.

На укр. мові. Бібліогр.: 31 назв; рис.: 64; табл.: 8.

Метою роботи є підвищення реалістичності формування графічних зображень за рахунок використання гексагонального растру.

Проведено аналіз галузей використання гексагонального растру та його особливостей.

Модифіковано метод оцінювальної функції для колового інтерполювання, який відрізняється від відомого використанням нових функціональних залежностей, що дозволило підвищити реалістичність за рахунок використання гексагональної моделі піксела.

Вперше запропоновано метод формування векторів на гексагональному растрі, особливість якого полягає у використання комбінованих переміщень, що дозволило до двох разів підвищити продуктивність лінійного інтерполювання.

Практичне значення отриманих результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботи теоретичних положень запропоновано алгоритми та розроблено програмні засоби для формування графічних зображень у комп’ютерних систем високореалістичної візуалізаціїу даний проект, і окупність проекту при різних інвестиціях та попиту.

Ключові слова: гексагональний растр, формування зображень, графічні примітиви, формування відрізків прямих і кіл, метод оцінювальної функції , високо реалістична комп’ютерна графіка.

ABSTRACT

Kozubenko M. V. Methods and software tools for creating graphic images on a hexagonal grid. Master's thesis on the specialty 121 - software engineering, educational program - software engineering.Vinnytsia: VNTU, 2022. 274 p.

In Ukrainian speech Bibliography: 31 titles; Fig.: 64; tab.: 8.

The purpose of the work is to increase the realism of the formation of graphic images due to the use of a hexagonal raster.

An analysis of the fields of use of the hexagonal raster and its features has been carried out.

The method of the evaluation function for circular interpolation has been modified, which differs from the known one by the use of new functional dependencies, which made it possible to increase realism due to the use of a hexagonal pixel model.

For the first time, a method of forming vectors on a hexagonal grid was proposed, the feature of which is the use of combined movements, which made it possible to increase the productivity of linear interpolation up to two times.

The practical significance of the obtained results is that, on the basis of the theoretical provisions obtained in the master's qualification work, algorithms were proposed and software tools were developed for the formation of graphic images in computer systems of highly realistic visualization

Keywords: hexagonal raster, image formation, graphic primitives, formation of line segments and circles, method of evaluation function, highly realistic computer graphics.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Роль комп’ютерної графіки, як однієї з основних забезпечуючих підсистем обчислювальної техніки постійно зростає, оскільки вона дозволяє в умовах сучасного рівня розвитку комп’ютерної техніки реалізувати найбільш прийнятну й звичну для користувача технологію подання інформації. Основною задачею комп’ютерної графіки є синтез зображень з високим ступенем реалістичності. Як показує практика, робота з реалістичними зображеннями у реальному часі можлива тільки за наявності спеціалізованих програмних і апаратних засобів.

Сьогодні в багатьох галузях обчислювальної техніки використовують гексагональний растр, який дає можливість підвищити розподільну здатність екранів, розширити напрямки руху між точками, згладити крокові траєкторії при формуванні графічних примітивів.

Для гексагона характерні рефлекційна симетрія та шестизв’язність. Розбиття гексагонального пікселя на субпікселі дає можливість надлишковості передачі кольору, що в свою чергу дозволяє розташувати елементи відтворення основних кольорів таким чином, щоб значно збільшити якість відтворення за рахунок більшого спектра передачі кольору.

При розробці алгоритмів формування графічних зображень на гексагональному растрі використовують його властивості. Дискретне подання кривих на гексагональному растрі краще передає форму та наближує розміри об’єкта до реальних у 80-85% випадках. Не дивлячись на історично зумовлену поширеність прямокутного растру, для задач, які вимагають високої точності обчислень на зображенні, доцільно використовувати дискретизацію на гексагональному растрі, що та потребує розроблення нових та модифікації наявних алгоритмів оброблення зображень.

Флексографічний друк, який використовує гексагональний растр, відноситься до інноваційної технології та забезпечує значне збільшення оптичної щільності та інтенсивності кольору відповідно до стандартних фарб. Іншими перевагами є високі реологічні властивості, багатократно покращена якість друку

та в значній мірі більш висока продуктивність. Сильною стороною є забезпечення високоякісного друку незважаючи на тонкий шар фарби.

Використання гексагонального растру забезпечує мінімальний тиск на матеріал та заміщує робочу площину без розривів і накладань

Одну з перших цифрових камер з матрицею, виготовленою з гексагональних пікселів, випустила компанія Fuji Photo Film . Така матриця мала назву Super CCD Honeycomb і створена з метою збільшення загальної площі фотодіодів на матриці, що дозволяє підвищити чутливість і розширити діапазон фото сенсорів.

За рахунок такої топології площа матриці використовується з більшою ефективністю – здійснюється захоплення більшої кількості світла на одиницю поверхні, і тому відображається більш ширший динамічний діапазон. Сенсори з гексагональними елементами дають кращі результати по горизонтальній і вертикальній розгортці, до якої найбільш чутливий зір людини.

Модуль фотоелектричного протеза складається з гексагональних пікселів шириною 70 мкм. Легкість імплантації цих бездротових і модульних гексагональних масивів, в поєднанні з їх високою роздільною здатністю, дає змогу функціонального відновлення зору у пацієнтів, які втратили зір з причин дегенерації сітківки

Компанія SONY запатентувала технологію розробки гексагональної матриці пікселів, де кожен піксель складається з набору шестигранних елементів, що відтворюють кожен свій колір.

Переваги такої технології - це поєднання фільтрів CMY і RGB, що значно збільшує характеристики спектральної чутливості фільтра. Це поєднання може покращити відтворюваність кольорів, щоб успішно виконувати обробку зображень. Фільтр CMY має більшу пропускну здатність, ніж фільтр RGB, тому він більш “насичений”, незалежно від технічної реалізації

Сьогодні цілий ряд ігор “стратегій”, розроблених для смартфонів, використовують шестикутні ігрові карти: UniWar – розробник TBS Games, Catan – розробник гексагональному растру рефлекційна симетрія, а, також, здатність

рівностороннього шестикутника без розривів і накладань заповнювати площину, дозволяє робити ігрові карти малого розміру що підходить до екранів смартфонів.

Компанія Samsung для екранів своїх смартфонів і планшетів розробила нову структуру пікселів, що використовується в новій технології AMOLED-екранів.

Широке використання гексагонального растру знайшло в різних комп'ютерних іграх - USM, Conquest! Medieval Realms – розробник Slitherine, Eastern Front: Conflict-series – розробник Joni Nuutinen.

Широке використання гексагонального растру в різних галузях передбачає необхідність розробки методів і алгоритмів формування графічних примітивів і графічного редактора для формування зображень. **Зв'язок роботи з науковими програмами, планами, темами**.

Робота виконувалась згідно плану виконання наукових досліджень на кафедрі програмного забезпечення

Мета та завдання дослідження. Метою роботи є підвищення реалістичності формування графічних зображень за рахунок використання гексагонального растру.

Основними задачами дослідження є:

провести аналіз галузей використання гексагонального растру;

запропонувати нові:

методи формування примітивів на гексагональному растрі;

метод підвищення продуктивності формування векторів

розробити програмні компоненти для систем візуалізації на основі запропонованих методів;

провести експериментальні дослідження розроблених засобів текстурування.

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

Предмет дослідження – методи та засоби формування графічних примітивів на гексагональному растрі

Методи дослідження. У процесі досліджень використовувались: теорія чисел і чисельних методів, методи аналітичної геометрії для розробки методів формування графічних примітивів; теорія алгоритмів для розробки графічного редактора; комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів:

1.Модифіковано метод оцінювальної функції для колового інтерполювання, який відрізняється від відомого використанням нових функціональних залежностей, що дозволило підвищити реалістичність за рахунок використання гексагональної моделі пікселя.

2. Вперше запропоновано метод формування векторів на гексагональному растрі, особливість якого полягає у використання комбінованих переміщень, що дозволило до двох разів підвищити продуктивність лінійного інтерполювання.

Практичне значення отриманих результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботи теоретичних положень запропоновано алгоритми та розроблено програмні засоби для формування графічних зображень у комп'ютерних систем високо реалістичної візуалізації.

Особистий внесок здобувача. Усі наукові результати, викладені у кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: [4] – модифікована архітектура веб-додатків, [5] – переваги використання гексагонального растру в картографії, [16] – програмна реалізація додатку для формування зображень на гексагональному растрі.

[16] – **Апробація матеріалів дисертації.** Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на Міжнародній практичної Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ;. 9-10 листопада 2019; XV міжнародній науково-практичній конференції Інформаційні технології і автоматизація – 2022. Одеса, 20-21 жовтня 2022 р.; The 8th International scientific and practical conference "Modern research in world science" (October 29-31, 2022) SPC "Sci-conf.com.ua", Lviv, Ukraine. 2022;

Міжнародній науково-практичній Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ;. 29 листопада 2022 р.; L науково-технічній конференції підрозділів ВНТУ, Вінниця, 10-12 березня 2021 р

Публікації. Основні результати опубліковано в 5 наукових працях у матеріалах конференцій.

1 АНАЛІЗ ВИКОРИСТАННЯ ТА ФОРМУВАННЯ ЗОБРАЖЕНЬ НА ГЕКСАГОНАЛЬНОМУ РАСТРІ

1.1 Особливості гексогонального растру

Гексагональний растр це набір правильних шестикутників, які розміщені один біля одного та утворюють растр з гексів (рис. 1.1).

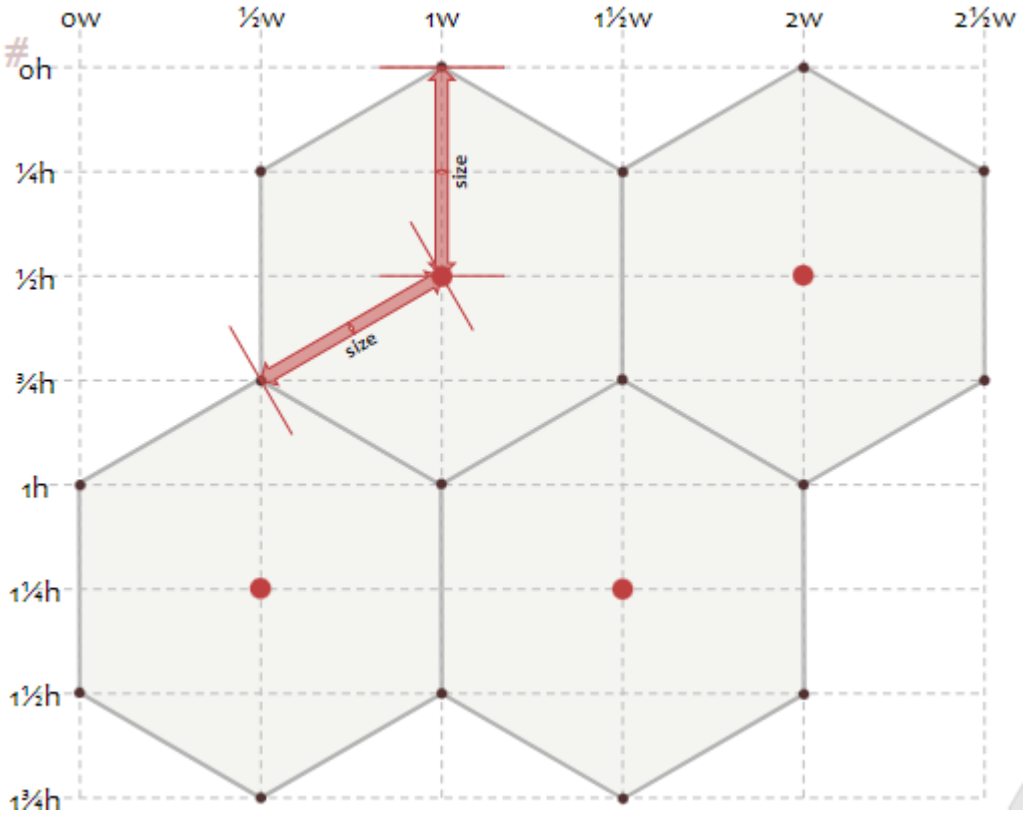


Рисунок 1.1 - Гексагональний растр

Гексагональний растр має рефлекційну симетрію, як і квадратний растр [15]. Але у гексагонального растра три лінії рефлекційної симетрії, в той час як в квадратного растра дві .

Гексагональний растр не є достатньо поширеним растром ніж квадратний, але має ряд переваг:

- більша щільність гратки - завдяки цій перевазі, гексагональний растр точніше передає форму та кращий реалізм переданого зображення приблизно на 80-85%;
- відсутність невикористаної площі - гексагональний растр немає вільної, невикористаної площі для формування зображень відносно наприклад кола, які будуть мати невикористану площу;
- шляхи пересування та найближче сусідство - кожен шестикутник має шість суміжних шестикутників у симетрично еквівалентних положеннях. Навпаки, прямокутна сітка має два різних типи найближчих сусідів: ортогональні сусіди, які мають спільний край, і діагональні сусіди, які мають спільний кут;
- краще покриття вигнутих об’єктів - на прикладі футбольного м’ячу можна побачити чітке покриття вигнутої фігури.

Також у гексагольного растру існують певні недоліки відносно квадратного растру.

- є шість сусідніх сусідів замість восьми з квадратом (якщо враховувати кути). Це зменшить точність аналізу підключення;
- найголовніше, що не можна розділити шестикутники на збільшення або зменшення масштабу вашої вибірки за допомогою шестикутника (з квадратом його легко об’єднати і розділити на нові квадрати). Тому квадрати краще підходять для ієрархічного аналізу.

Отже, гексагональний растр має багато переваг над квадратним растром, але має більшу складність апаратної частини.

1.2 Формування гексогонального растру та складових пікселів

Гексагон - шестикутна фігура, у якої всі сторони рівні. Програмне формування гексагону на квадратному растрі є досить простим. У правильному шестикутнику внутрішні кути дорівнюють 120°. Є шість «клинів», кожен з яких є

рівностороннім трикутником з кутами 60° всередині. Кожен кут знаходиться на однаковій відстані від центру (рис. 1.2).

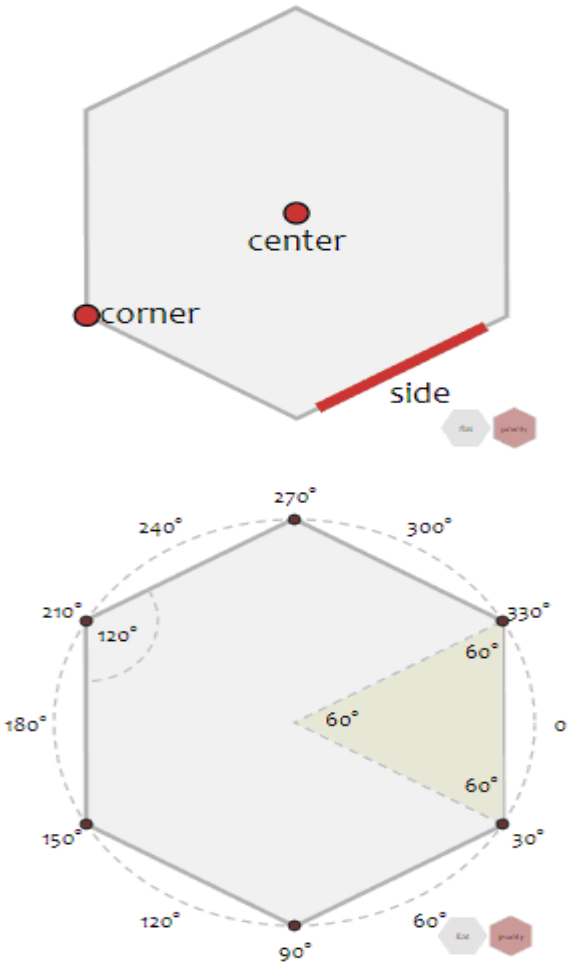


Рисунок 1.2 –Зображення гексагону

Для формування кута гекса потрібно використати формулу:
60◦ * номер кута(0 - 5) - 30◦. Далі отримані дані використовуються у формулі кута радіуса описаного кола, PI / 180 * кут гекса. Часто використовують функцію, яка буде малювати гекс з заданими параметрами: координати центри, довжина і кути, center.x + size * cos(кута радіуса описаного кола), center.y + розмір * sin(кута радіуса описаного кола).

Для формування гексагонального растру потрібно розташувати гексагони у певному порядку, щоб утворилась гексагональна сітка (рис. 1.3).

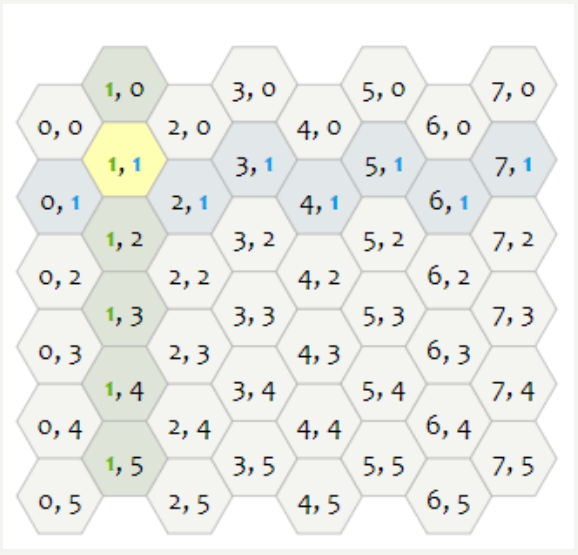


Рисунок 1.3 - Гексагональна сітка (вертикальне розміщення)

Найпоширенішим підходом є зміщення кожного другого стовпця чи рядка. Є можливість розміщувати непарні і парні стовпці/рядки, тому горизонтальний і вертикальний шестикутники мають два варіанти.

Ще один варіант оцінити шестигранну сітку — побачити, що є *три* основні осі, на відміну від *двох* , які є в квадратної сітки. У них є елегантна симетрія.

Використаємо кубічну сітку і виріжемо діагональну площину в (рис. 4). Це ідея досить ефективна із алгоритмами шістнадцяткової сітки:x + y + z = 0.

- Тривимірні декартові координати виконують стандартні векторні операції: операції додавання/віднімання координати, множення/ділення на скаляр тощо. Є можливість повторно використовувати ці операції з гексагональними сітками. Координати зсуву не підтримують ці операції;
- Тривимірні декартові координати використовують існуючі алгоритми; обертання, відображення, малювання ліній, перетворення на/з екранних координат тощо. Доступна адаптація цих алгоритмів для роботи на шестикутних сітках (рис. 1.4).

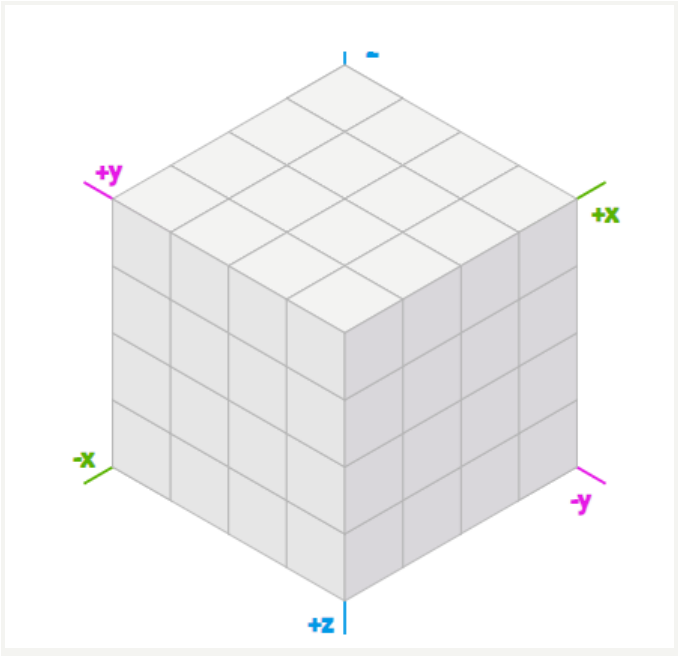


Рисунок 1.4 - Кубічна сітка

Розглянемо, як координати куба працюють на шестигранній сітці. Вибір шестигранників виділить координати куба, що відповідають трьом осям (рис. 1.5).

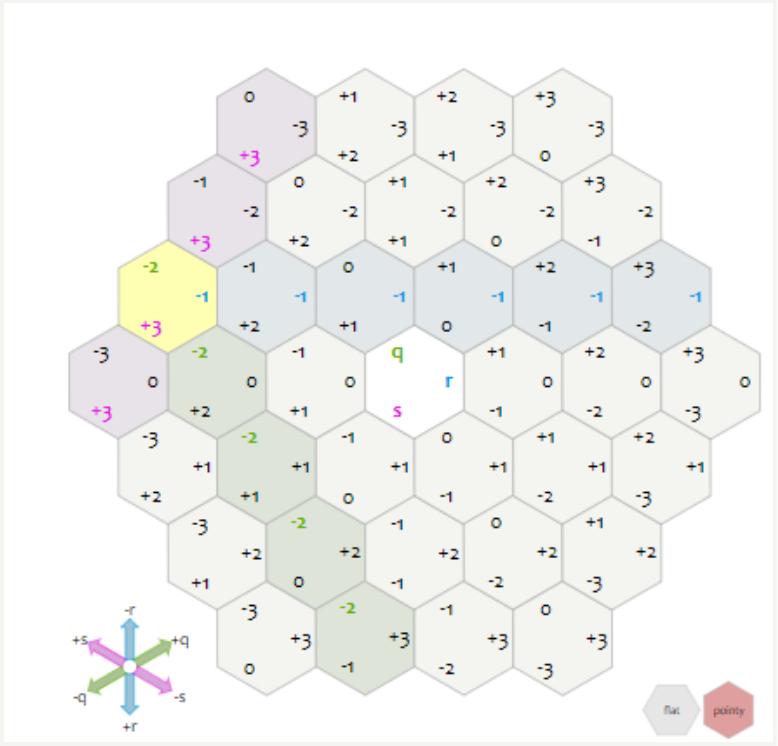


Рисунок 1.5 - Координати куба

- Кожний напрямок на кубічній сітці відповідає *лінії* на шестигранній сітці. Виділемо шістнадцятковий код за допомогою r0, 1, 2, 3, щоб побачити зв'язки. Ряд позначено синім кольором. Те саме для q(зеленого) і s(фіолетового);
- Кожен напрямок на шестигранній сітці є комбінацією *двох* напрямків на кубічній сітці. Наприклад, північний захід на шістнадцятковій сітці лежить між + 100 -r, тому кожен крок на північний захід передбачає додавання 1 до 100 віднімання 1 від r. Було використано цю власність у розділі сусідів.

Координати куба є розумним вибором для шістнадцяткової системи координат. Обмеження полягає в тому, що алгоритми повинні це зберегти. Обмеження також гарантує наявність канонічної координати для кожного гексу.q + r + s = 0

Осьова система координат, яку іноді називають «трапецієподібною», «косою» або «скошеною», є такою самою, як і система куба**,** за винятком того, що не зберігається координата s (рис. 1.6).

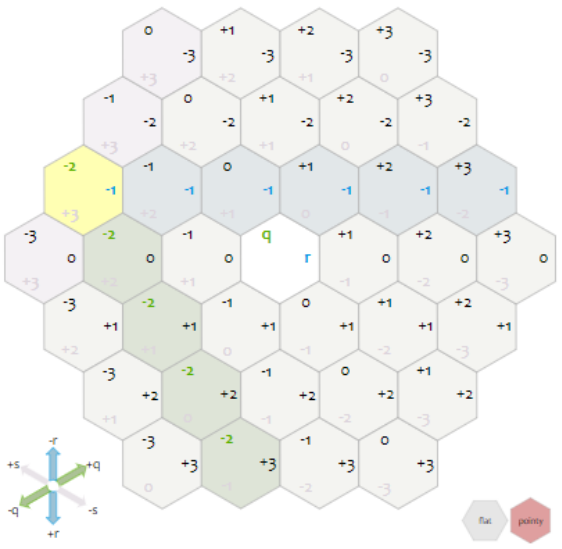


Рисунок 1.6 - Осьова система координат

Оскільки є обмеження, можуть бути виконані обчислення, коли це потрібно. q + r + s = 0s = -q-r

Осьова/кубічна система дозволяє додавати, віднімати, множити та ділити за допомогою шістнадцяткових координат. Зміщені системи координат не дозволяють цього і це частково робить алгоритми простішими з осьовими/кубічними координатами.

Хоча рекомендуються осьові/кубічні координати, якщо дотримуватися зсувних координат(рис. 1.7).

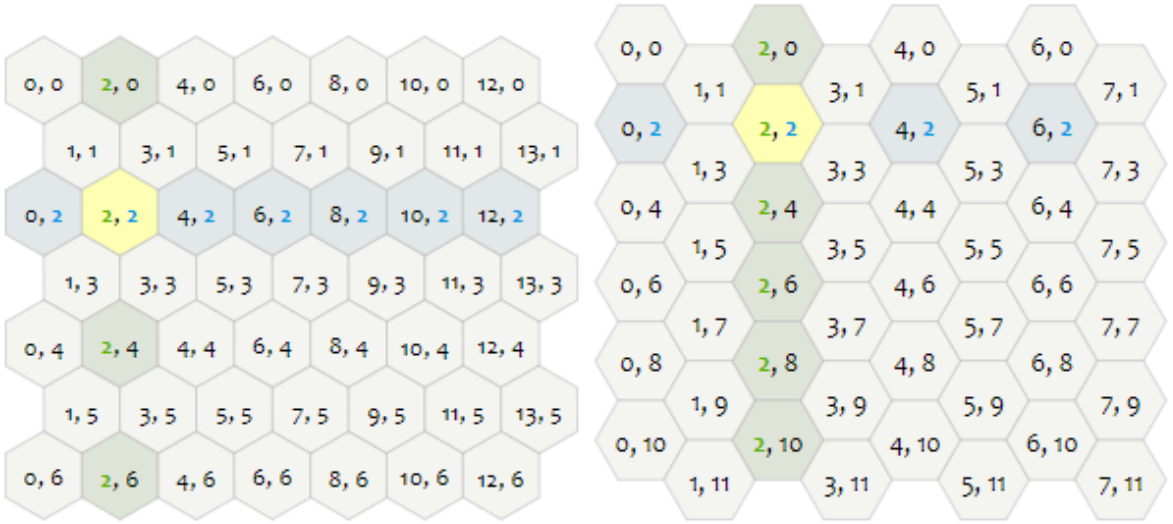


Рисунок 1.7 - Подвоєний варіант

Це полегшує реалізацію багатьох алгоритмів. Замість чергування подвоєні координати *подвоюють* горизонтальний, або вертикальний розмір кроку. Він має обмеження. У горизонтальному (загострений верхній шестигранник) розкладці збільшує стовпець на 2 у кожному шестиграннику; у вертикальній (плоский верхній шестигранник) розкладці він збільшує рядок на 2 у кожному шестиграннику. Це дозволяє використовувати проміжні значення для гексів, які знаходяться на півдорозі між ними:(col + row) % 2 == 0

Таблиця 1 Порівняння різних підходів для побудови гексагональної сітки

	Зсув	Подвоєний	Осьовий	Куб
Точкове обертання	парний відр.	Подвійна ширина	осьовий	куб
Плоске обертання	парне	подвійна висота		
Інші ротації	ні		так	
Векторні операції (віднімання, масштабування, додавання)	ні	так	так	так
Зберігання масиву	прямокутний	ні	ромб	ні
Хеш-сховище	Будь-якої форми			
Гексагональна симетрія	ні	ні	ні	так
Легкі алгоритми	мало	деякі	більшість	більшість

1.3 Використання гексогонального растру в різних галузях

Гексагональний растр на даний момент вже має широку сферу використання. Можливо скоро можна буде побачити сенсорні екрани, які будуть мати гексагональну сітку, і набагато чіткіше зображення ніж у квадратній сітці.

Також використання гексагональної сітки пов’язано з потребами

користувачів. Квадратна сітка набагато дешевша у виготовленні апаратури та програмного забезпечення [21].

До перевага прямокутного растру можна віднести меншу затратність при обчисленнях, універсальність використання, ефективність заповненості растра. Серед недоліків можна виділити незадовільну якість формування графічних зображень. З метою її підвищення у пристроях відображення використовують гексагональний растр [11].

Застосування даного растру у іграх. Можна зустріти гексагональний растр у іграх, здебільшого, якщо гра має жанр стратегії, де кожен гекс містить площу, де може бути персонаж, юніт гри, або також певна область, з ресурсами наприклад. Чому у стратегіях використовують гекс, як найменша область карти? Тому, що гексагональний растр набагато кращий для відображення площин [8]. Використання даної сітки було використано у грі Цивілізація (рис. 1.8).

Рисунок 1.8 - Гра «Цивілізація»

Також використання гексагонального растру у грі має певну перевагу для

переміщення персонажу по клітинках, оскільки гекс має шість сторін, то у рух може здійснюватись у 6 сторін, і це важливо для анімації.

Шестикутники використовувалися в деяких настільних і комп'ютерних іграх, оскільки ігри менш спотворюють відстані, ніж квадратні сітки. Це частково тому, що кожен шестикутник має більше недіагональних сусідів, ніж квадрат. Шестикутники мають приємний вигляд і зустрічаються в природі (наприклад, стільники).

Якщо взяти наприклад квадрат, який має тільки 4 сторони, то пересування по діагоналі не буде мати гарної анімації, і це б знизило рівень гри серед користувачів (рис. 1.9).

Рисунок 1.9 - Гра на квадратній сітці

Ще гексагональний растр знайшов застосування у побудові гідрологічно-коректної цифрової моделі рельєфу [5]. Це дозволяє побудувати грунтовий покрив на гексі, який буде містити більшість чіткі межі, відображати більш чітку нерівну поверхність (рис. 1.10), що дасть змогу чіткіше працювати з приладами, які шукають поклади ресурсів, що збільшить ефективність добування, і в результаті дохід для підприємства.

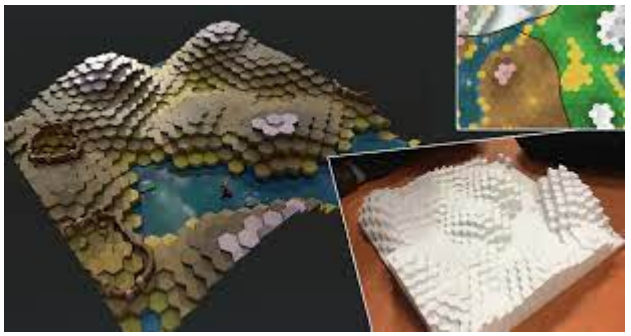


Рисунок 1.10 - 3D модель поверхні на гексагональному растрі

UBER використовуємо систему сіток, щоб розділити події на гексагональні ділянки, іншими словами, комірки. Точки даних розбиті на шестикутники та можуть бути записані за допомогою шестикутних даних. Наприклад, компанія розраховує на підвищення цін, вимірюючи попит і пропозицію в шестикутниках у кожному місті, яке обслуговується. Ці шестикутники є основою для аналізу ринку Uber.

Шестикутники були важливим вибором, оскільки люди в місті часто перебувають у русі, а шестикутники мінімізують помилку квантування, яка виникає, коли користувачі рухаються містом. Шестикутники також дозволяють легко апроксимувати радіуси, як у цьому прикладі за допомогою Elasticsearch (рис. 1.11).

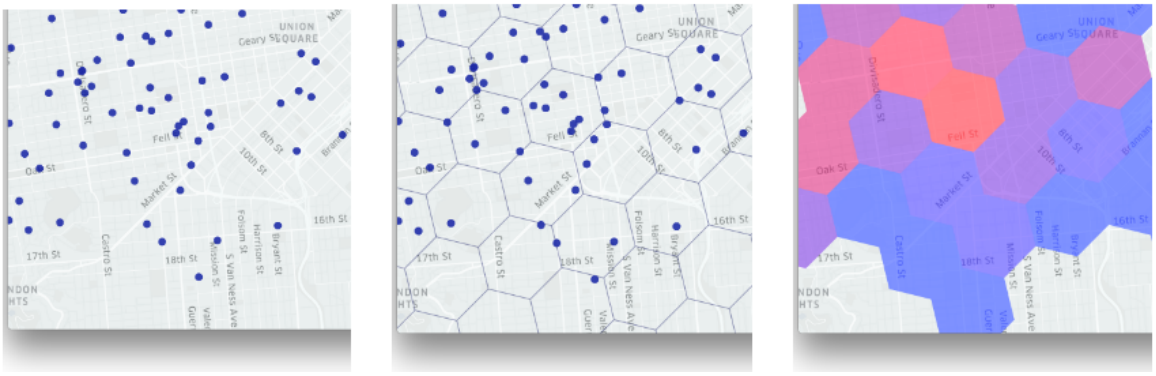


Рисунок 1.11 - Uber використовує гекси для аналізу ринку

Гексагональний растр дає більш чітку інформацію про прибуток і загруженість певних районів, отже після аналізу даних можна сформувати більш кращі рішення для збільшення прибутку, оптимізації витрат палива, визначити кількість авто на район, найчастіші зупинки, більш перспективні райони, які наймають uber частіше, сформувати відгуки по районах, і дізнатись, чому райони мають різні відгуки і тд.

Також дана сітка використовуються для аналізу економічних, соціальних та інших показників на державному рівні у Великій Британії (рис. 1.12).

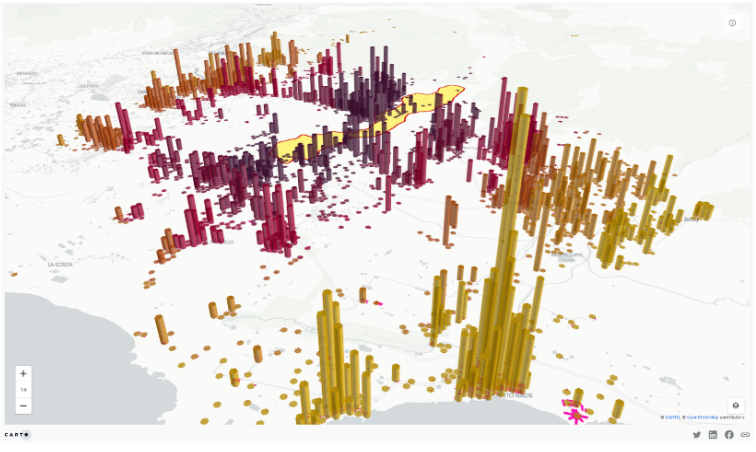


Рисунок 1.12 - Аналіз різних економічних та соціальних показників у Великій Британії

Отже, було отримано деяку інформацію про те, як використання регулярних зон може бути корисним. Основна їх перевага полягає в тому, що регулярні зони викладені мозаїкою формують регулярну безперервну сітку. Тільки дві інші форми здатні на це; квадрати і трикутники. Однак шестикутники мають низку переваг перед цими іншими формами:

- Відстань між центроїдом шестикутника до всіх сусідніх центроїдів однакова в усіх напрямках;
- Відсутність гострих кутів у правильному шестикутнику означає, що жодні ділянки фігури не є викидами в будь-якому напрямку;
- Усі сусідні шестикутники мають однакове просторове відношення до центрального шестикутника, що робить процес просторового запиту та з'єднання більш простим;
- На відміну від квадратних сіток, геометрія шестикутників добре структурована для представлення кривих географічних об'єктів, які рідко мають перпендикулярну форму, наприклад річок і доріг;
- «М'якша» форма шестикутника порівняно з квадратом означає, що він краще демонструє поступові просторові зміни.

Завдяки цим перевагам, Велика Британія використовує такий підхід для аналізу показників у державі. Згідно статистики, країна має одні з найвищих економічних показників, в особливості ВВП.

Незважаючи на те, що існують переваги шестикутників для картографування, регулярні сітки та шестикутники не завжди є оптимальним рішенням для вашої карти. Вирішуючи, чи використовувати шестикутну сітку, слід враховувати деякі ситуації:

- Втрата необроблених даних. Агрегування даних у шестикутну сітку з необробленої форми означає, що можна втратити необроблені значення даних. Блок перепису матиме точну кількість мешканців, які були обстежені, тоді як трансформації до іншої географії будуть змодельовані;
- Граничні ефекти. Гексагональна карта прибережної території може

показати, що щільність населення уздовж узбережжя низька. Однак шестикутна комірка насправді може містити великі площі моря, а щільність населення може бути дуже високою. Це можна пом'якшити, обрізавши шестикутну сітку на суші або використовуючи компактну візуалізацію з шестикутниками з вищою роздільною здатністю, розташованими вздовж узбережжя;

- Точність даних. Підрахування кількості університетів, представлених у вигляді балів, у кожній клітинці шестикутника. Університети часто займають великі території; може статися так, що точкова функція розташована в одній шестикутній клітинці, але насправді університет розташовується в кількох клітинках. Важливо враховувати, наскільки точними є ваші вихідні дані під час виконання цих агрегацій;
- Розмір комірки. Розмір ваших шестикутників має бути природно прив'язаний до історії, яка містить інформацію. До яких запитань, відповідей і, зрештою, рішень, користувачі можуть прийти користуючись вашою картою? Наприклад, якщо читач карти цікавиться причинно-наслідковими зв'язками аварій на певному перехресті вулиць, немає сенсу пропонувати йому сітку з клітинками шириною 1000 метрів.

Флексографічний друк, який використовує гексагональний растр забезпечує значне збільшення оптичної щільності та інтенсивності кольору відповідно до стандартних фарб [14]. Іншими перевагами є високі реологічні властивості, багатократно покращена якість друку та в значній мірі більш висока продуктивність. Сильною стороною є забезпечення високоякісного друку незважаючи на тонкий шар фарби.

Використання гексагонального растру забезпечує мінімальний тиск на матеріал та замощує робочу площину без розривів і накладань

Одну з перших цифрових камер з матрицею, виготовленою з гексагональних пікселів, випустила компанія Fuji Photo Film . Така матриця створена з метою

збільшення загальної площі фотодіодів на матриці, що дозволяє підвищити чутливість і розширити діапазон фотосенсорів.

За рахунок такої топології площа матриці використовується з більшою ефективністю – здійснюється захоплення більшої кількості світла на одиницю поверхні, і тому відображається більш ширший динамічний діапазон. Сенсори з гексагональними елементами дають кращі результати по горизонтальній і вертикальній розгортці, до якої найбільш чутливий зір людини.

Модуль фотоелектриого протеза складається з гексагональних пікселів шириною 70 мкм. Легкість імплантації цих бездротових і модульних гексагональних масивів, в поєднанні з їх високою роздільною здатністю, дає змогу функціонального відновлення зору у пацієнтів, які втратили зір з причин дегенерації сітківки

Компанія SONY запатентувала технологію розробки гексагональної матриці пікселів, де кожен піксель складається з набору шестикутних елементів, що відтворюють кожен свій колір.

Переваги такої технології - це поєднання фільтрів CMY і RGB, що значно збільшує характеристики спектральної чутливості фільтра. Це поєднання може покращити відтворюваність кольорів, щоб успішно виконувати обробку зображень. Фільтр CMY має більшу пропускну здатність, ніж фільтр RGB, тому він більш “насичений”, не залежно від технічної реалізації

Компанія Samsung для екранів своїх сматфонів і планшетів розробила нову структуру пікселів, що використовується в новій технології AMOLED-екранів.

Широке використання гексагонального растру в різних USM, Conquest! Medieval Realms – розробник Slitherine, Eastern Front: Conflict-series – розробник Joni Nuutinen.

Властива галузях передбачає необхідність розробки методів і алгоритмів формування графічних примітивів і графічного редактора для формування зображень.

Як результат, можна підвести, що гексагональний растр є доволі поширеним, і в більшості випадків має переваги у використанні відносно

квадратної сітки.

Широке використання гексагонального растру в різних галузях передбачає необхідність розробки методів і алгоритмів формування графічних примітивів і графічного редактора для формування зображень.

Висновки

1. Розглянуто використання гексагонального растру у різних галузях діяльності людини.
2. Проаналізовано особливості гексагонального растру.
3. Широке використання гексагонального растру в різних галузях передбачає необхідність розробки методів і алгоритмів формування графічних примітивів і графічного редактора для формування зображень

2 МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ ФОРМУВАННЯ ВІДРІЗКІВ ПРЯМИХ НА ГЕКСОГОНАЛЬНОМУ РАСТРІ

2.1 Особливості формування відрізків прямих на прямокутному растрі

Існує декілька алгоритмів для побудови відрізків прямих на піксельному растрі, маючи тільки початкову та кінцеву точки.

Алгоритм DDA-лінії формує відрізок прямий між двома заданими точками, використовуючи обчислення з дійсними числами [3]. Абревіатура DDA у назві цього алгоритму машинної графіки походить від англ. Digital Differential Analyzer (цифровий диференціальний аналізатор) - обчислювальний пристрій, який раніше застосовувався для генерації векторів. Незважаючи на те, що зараз цей алгоритм практично не застосовується, він дозволяє зрозуміти особливості, що зустрічаються при растеризації відрізка та способи їх вирішення.

У будь-якій двовимірній площині, якщо з'єднати дві точки (x0, y0) і (x1, y1), отримаємо відрізок. Але у випадку комп'ютерної графіки не можливо з'єднати будь-які дві координатні точки, Для цього необхідно обчислити координати проміжних точок і додати для кожної проміжної точки піксель потрібного кольору за допомогою таких функцій, як putpixel(x). , y, K) у C, де (x, y) є координатою, а K позначає деякий колір.

Для використання графічних функцій наш системний вихідний екран розглядається як система координат, де координата верхнього лівого кута дорівнює (0, 0), і коли виконується рух вниз, то ордината y збільшується, а коли рухаємося праворуч, ордината x зростає для будь-якої точки (x, y). Тепер для генерації будь-якого відрізка лінії потрібні проміжні точки, і для їх обчислення можна використати базовий алгоритм, який називається алгоритмом генерації лінії DDA (цифровий диференціальний аналізатор).

Переваги алгоритму DDA:

- Це простий і легкий у реалізації алгоритм;
- Це дозволяє вилучити використання кількох операцій, які мають високу складність у часі;

- Це швидше, ніж пряме використання рівняння лінії, оскільки воно не використовує множення з плаваючою комою та обчислює точки на лінії.

Недоліки алгоритму DDA:

- передбачає округлення та використання арифметики з плаваючою комою, тому має високу часову складність;
- має низьку точність визначення кінцевої точки;
- Через обмежену точність обчислення з плаваючою комою алгоритм має відносно високу похибку.

Алгоритм Брезенхема (англ. Bresenham's line algorithm) - це алгоритм, що визначає, які точки двовимірного растру потрібно зафарбувати, щоб отримати близьке наближення відрізка прямої лінії між двома заданими точками. Алгорим був розроблений Джеком Е. Брезенхемом (Jack E. Bresenham) у компанії IBM у 1962 році. Алгоритм широко використовується, зокрема, для формування зображення відрізків прямих на екрані комп'ютера .

Алгоритм Брезенхема використовує лише цілі значення, цілочисельні порівняння та додавання [6]. Це робить алгоритм Брезенхема більш ефективним, швидким і легшим для обчислення, ніж алгоритм DDA.

Алгоритм Брезенхема забезпечує точніші результати, ніж алгоритм DDA, завдяки використанню цілочисельних арифметичних обчислень.

2.2 Метод та програмний засіб формування відрізків прямих на гексагональному растрі

Відрізки прямих - це один з основних способів, яким можна користуватись від стартових ескізів до фінішної графічної роботи. Вектор може передавати об’єм, а також відображає певні форми. Розрізняють такі типи ліній:

- Лінія різних товщин - це лінії, які можуть передавати перспективу, плановість. Мають один і той самий фон, коли товщина лінії може змінюватись по певному алгоритму. Лінії різної товщини дуже добре використовуються, коли потрібно сконцентрувати увагу на певному елементів композиції;
- Лінія однієї товщини - це лінії, які добре окреслюються межу предмету, і лежать в площині, не виходячи за межі цієї площини. Лінія може мати заданий певний колір, тон. Завдяки зміні тону, лінії можуть сприйматись по-різному;
- Лінії з зміною - це лінії, які підкреслюють динаміку, то потовщуються, змінюючи форму.

Відрізки прямих у сукупності графічних примітивів мають найбільшу питому вагу. В зв’язку з цим, при розробці графічних пристроїв алгоритмам лінійної інтерполяції приділяють особливу увагу.

В описі алгоритму малювання ліній Брезенхема один аспект залишився неосвітленим - що, якщо нам потрібно знайти кожен шестикутник, що перетинається з лінією [19]. Ви можете побачити різницю між алгоритмом малювання лінії та алгоритмом «прямої видимості» на рисунку 2.1, де шістнадцятковий (4,0) відвідується лише в останньому випадку.

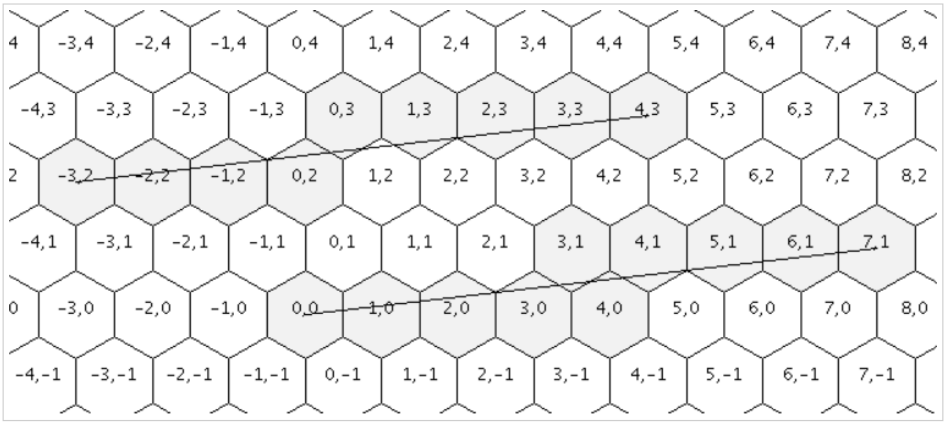


Рисунок 2.1 - Варіанти формування відрізків на гексогональному растрі

Використання такого алгоритму може бути корисним, особливо у ігровій індустрії, де використовується гексагональний растр, і лінія повинна містити всі гекси, які потрапляють під час малювання відрізка.

Алгоритм малювання відрізка. Спочатку обчислюємо N як шістнадцяткову відстань між кінцевими точками. Тут існує декілька варіантів залежно від побудови гексагонального растру .

Розглянемо обчислення дистанції на координатах 3D-куба. Оскільки шестикутні координати куба базуються на координатах 3D-куба, можливо адаптувати обчислення відстані для роботи на гексагональних сітках. Кожен шестикутник відповідає кубу в 3d просторі. Сусідні шестикутники знаходяться на відстані 1 один від одного в шестигранній сітці, але на відстані 2 один від одного в сітці куба [10]. На кожні 2 кроки кубічної сітки потрібен лише 1 крок шестигранної сітки. У сітці тривимірного куба відстані дорівнюють abs(dx) + abs(dy) + abs(dz). Відстань у шестигранній сітці вдвічі менша (рис. 2.2).

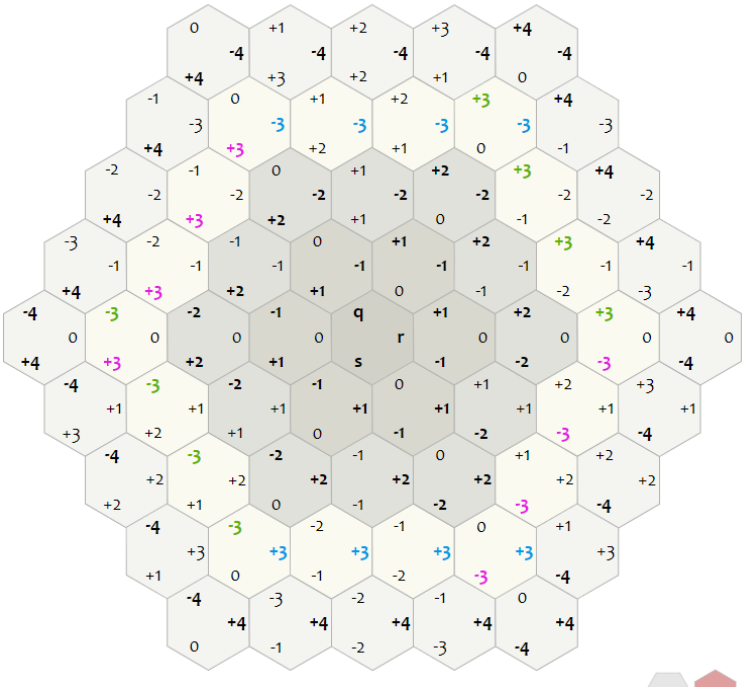


Рисунок 2.2 - відстань між гексами у гексагональній сітці

Розглянемо обчислення дистанції на координатах осі.

В осьовій системі третя координата неявна. Завжди можемо перетворити аксіал у куб, щоб обчислити відстань. Існує багато різних способів запису шістнадцяткової відстані в осьових координатах. Незалежно від способу запису, осьова шістнадцяткова відстань є похідною від відстані Манхеттена на кубах. Наприклад, «різниця відмінностей», описана тут, є результатом запису a.q + a.r - b.q - b.r як a.q - b.q + a.r - b.r і використання форми «max» замість форми «поділити на два» cube distance. Усі вони еквівалентні, коли ви бачите зв’язок із координатами куба [22].

Дізнавшись відстань між обраними гексами, може бути виконананий наступний етап розрахунків.

Рівномірно відбираємо N+1 точку між точками A і B [13]. Використовуючи лінійну інтерполяцію, кожна точка буде A + (B - A) * 1,0/N * i для значень i від 0 до N включно. На діаграмі ці вибіркові точки є темно-синіми точками. Це призводить до координат з плаваючою комою. Перетворемо кожну точку вибірки (float) назад у hex (int) (рис. 2.3).

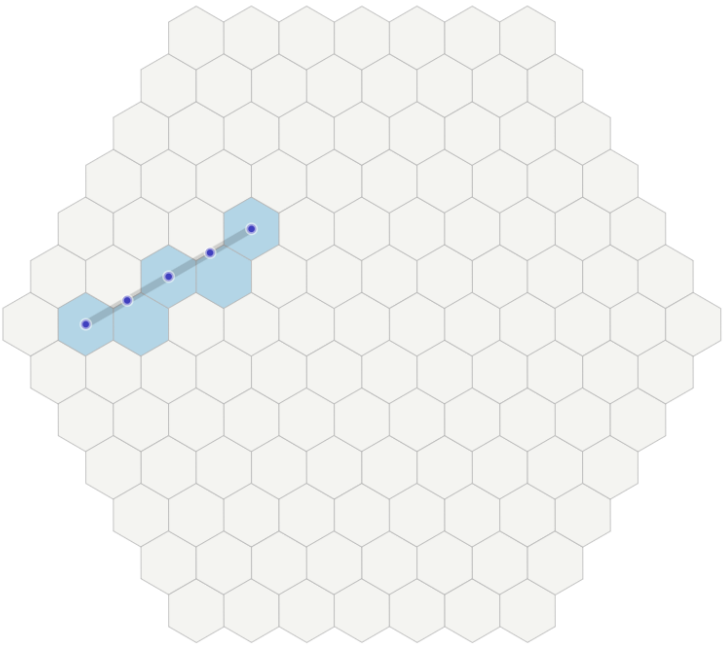


Рисунок 2.3 - Відстань між гексами та тонування гексів

За допомогою видозміненого алгоритму Брезенхема на координатах 3D-куба, було намальовано лінії між двома обраними гексами, які в свою чергу після обчислення були зафарбовані в певний колір, і утворили лінію.

Формування крокової траєкторії на гексагональному растрі можна підвищити за умови формування не елементарних кроків, а комбінації кроків, наприклад, в кожному такті двох кроків.

При інтерполяції відрізка прямої, яка задана приростами Δ по осях координат X, Y, загальна формула буде мати такий вигляд:

$$OF_i = \frac{y_i}{x_i} - \frac{\Delta y}{\Delta x} = \frac{y_i * \Delta x - x_i * \Delta y}{x_i * \Delta x} \quad (2.1)$$

Щоб визначити, по якій з осей OX чи OY робити наступний крок інтерполяції, знаходять знак OF_i. Покрокове формування відрізка буде відбуватися таким чином, як це зображено на рисунку 2.4.

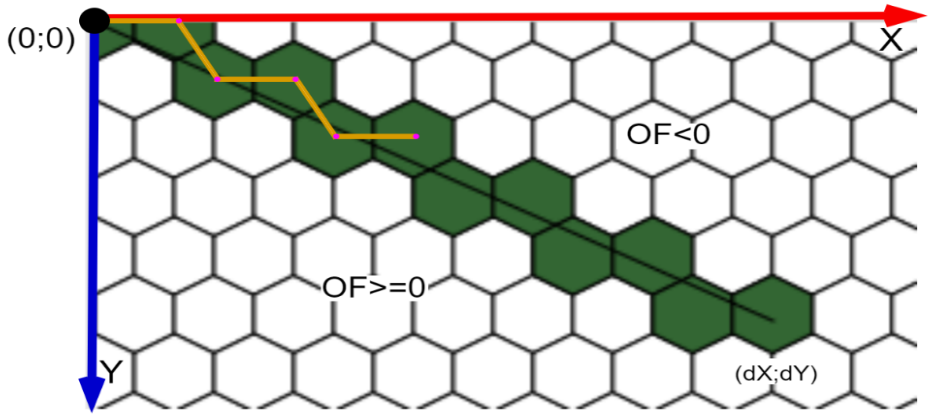


Рисунок 2.4 – Покрокове формування відрізка прямої

Якщо точка траєкторії знаходиться вище відрізка прямої b, то оцінювальна функція $OF_i >= 0$, і наступний крок треба робити по осі OX. Якщо точка траєкторії при інтерполяції знаходиться нижче прямої b, то $OF_i <= 0$ і наступний крок треба робити по обох осях, тобто виконується діагональний крок.

Оскільки в формулі:

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

знаменник дробу $x_i * \Delta x$ не впливає на знак оцінювальної функції, то можливо ввести формулу $OF_i = y_i * \Delta x - x_i * \Delta y$ для пришвидшення обрахунку.

Таким чином необхідно знайти нове значення оцінювальної функції при виконанні кроку по осі OX. При формування крокової траєкторії значення координат по OY не зміниться, а значення координат по OX зросте на 1, тобто, $x_{i+1} = x_i + 1$.

Для пришвидшення виконання даного обчислення можливо внести таку формулу:

$$OF_{i+1} = y_i * \Delta x - (x_i + 1) * \Delta y = y_i * \Delta x - x_i * \Delta y - \Delta y = OF_i - \Delta y \quad (2.3)$$

Аналогічно обрахується нове значення оцінювальної функції при виконанні кроку по обох осях OX і OY. Значення координат по осі OX збільшиться на 1 , а по осі OY на 1.

Для розрахунку значення координати по осі OY слід виконати такі дії. Згідно рисунку 2.5 можливо визначити співвідношення x та y .

$$sin60^\circ = \frac{1}{2} * x; \quad (2.4)$$

$$\frac{\sqrt{3}}{2} = \frac{1}{2} * x; \quad (2.5)$$

$$x = \frac{1}{\sqrt{3}}; \quad (2.6)$$

$$x = 2 * y; (2.7)$$

Таким чином отримане співвідношення $x = 2 * y$.

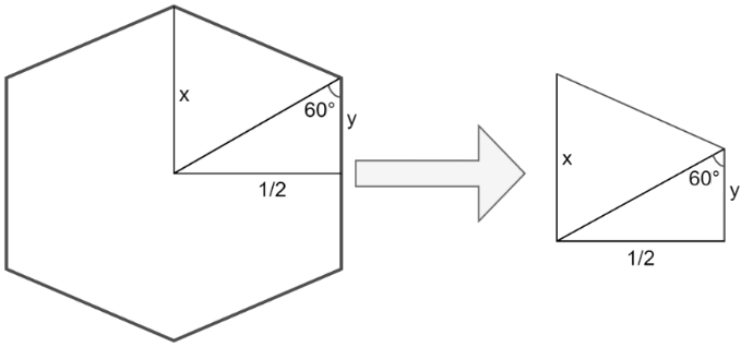


Рисунок 2.5 – Визначення співвідношення x та y

Оскільки $y = \frac{1}{2\sqrt{3}}$, то можна стверджувати що нове значення OY можна визначити як $OY = 3y = \frac{1}{2\sqrt{3}}$. Крок визначення y приведений на рисунку 2.6.

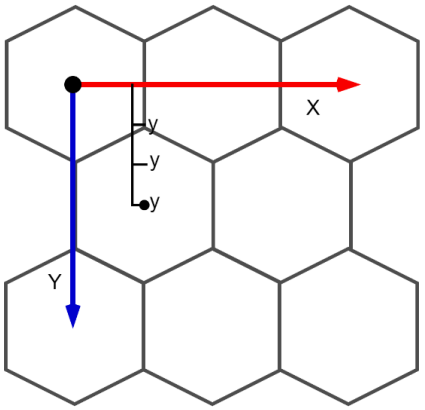


Рисунок 2.6 – Встановлення позиції центра гексагона відносно OY

Значення координат крокової траєкторії по осі OY розраховується виходячи з того, що на гексагональному растрі відстань між центрами сусідніх пікселів рівна одиниці. Таким чином, при виконання діагонального кроку координату по OX на 1, а по осі OY на 1.

З урахуванням останнього, нове значення оцінювальної функції знаходять за формулою

$$OF_{i+1} = (y_i + \frac{3}{2\sqrt{3}}) * (\Delta x + \frac{1}{2}) * \Delta y = y_i * \Delta x - x_i * \Delta y - \frac{\Delta y}{2} \quad (2.8)$$

$$OF_i + \frac{3*\Delta x}{2\sqrt{3}} - \frac{\Delta y}{2} \quad (2.9)$$

.

Отримані формули та розрахунки справедливі для інтерполяції відрізків прямих з кутами нахилу від 0° до 60° по відношенню до осі OX.

При формуванні відрізків прямих з кутами нахилу від 60° до 90° елементарні кроки інтерполяції по осі OX не виконуються. В цьому випадку кожен крок виконується по обох осях, тобто діагонально, як це зображено на рисунку 2.7.

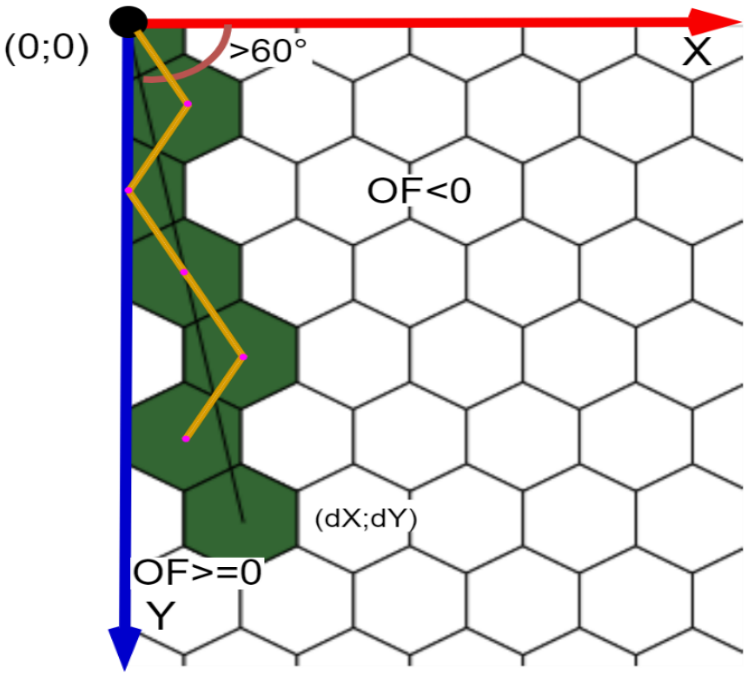


Рисунок 2.7 – Формування відрізків прямих з кутами нахилу від 60° до 90°

Покрокове формування відрізка прямої для кутів нахилу відрізків прямих в діапазоні від 60° до 90° буде обраховуватися за новими правилами. Таким чином для $OF_i >= 0$ при виконанні діагонального кроку координата по OX буде змінюватись за формулою $x_{i+1} = x_i + \frac{1}{2}$, а по OY – $y_{i+1} = y_i + \frac{3}{2\sqrt{3}}$.

Нова оцінювальна функція:

$$OF_{i+1} = (y_i + \frac{3}{2\sqrt{3}}) * \Delta x - (x_i + \frac{1}{2}) * \Delta y =$$
$$= y_i * \Delta x + \frac{3\Delta x}{2\sqrt{3}} - x_i * \Delta y - \frac{\Delta y}{2} \quad (2.10)$$

$$OF_i + \frac{3*\Delta x}{2\sqrt{3}} - \frac{\Delta y}{2} \quad (2.11)$$

Для $OF_i < 0$ при виконанні діагонального кроку координата по OX буде змінюватись за формулою $x_{i+1} = x_i - \frac{1}{2}$, а по OY – $y_{i+1} = y_i + \frac{3}{2\sqrt{3}}$.

Оцінювальна функція для даного випадку:

$$OF_{i+1} = (y_i + \frac{3}{2\sqrt{3}}) * \Delta x - (x_i - \frac{1}{2}) * \Delta y =$$
$$= y_i * \Delta x + \frac{3\Delta x}{2\sqrt{3}} - x_i * \Delta y + \frac{\Delta y}{2} \quad (2.12)$$

$$OF_i + \frac{3*\Delta x}{2\sqrt{3}} + \frac{\Delta y}{2} \quad (2.13)$$

Для програмної реалізації даного методу актуальним є розробка алгоритму для формування лінії на імітованому екрані застосунку «Hexagon Screen».

Отриманий результат формування алгоритму за описаною процедурою формування відрізку на гексагональному екрані зображено на рисунку 2.12.

Програмна реалізація надає можливість формувати гексагон різного розміру. Таким чином для перевірки вірності виконання алгоритму пропонується

використати таку координату, що буде легко перевірити виконуючи власні розрахунки. Для зручності розгляду пікселів користувач може збільшити гексагон до 100 приведених пікселів на екрані користувача.

Інші налаштування залишаються у позиціях, що виставлені «по замовчуванню». Застосувавши такі налаштування можливо чітко відслідкувати роботу алгоритму. Користувач за потреби може встановити нове положення кінцевого пікселя.

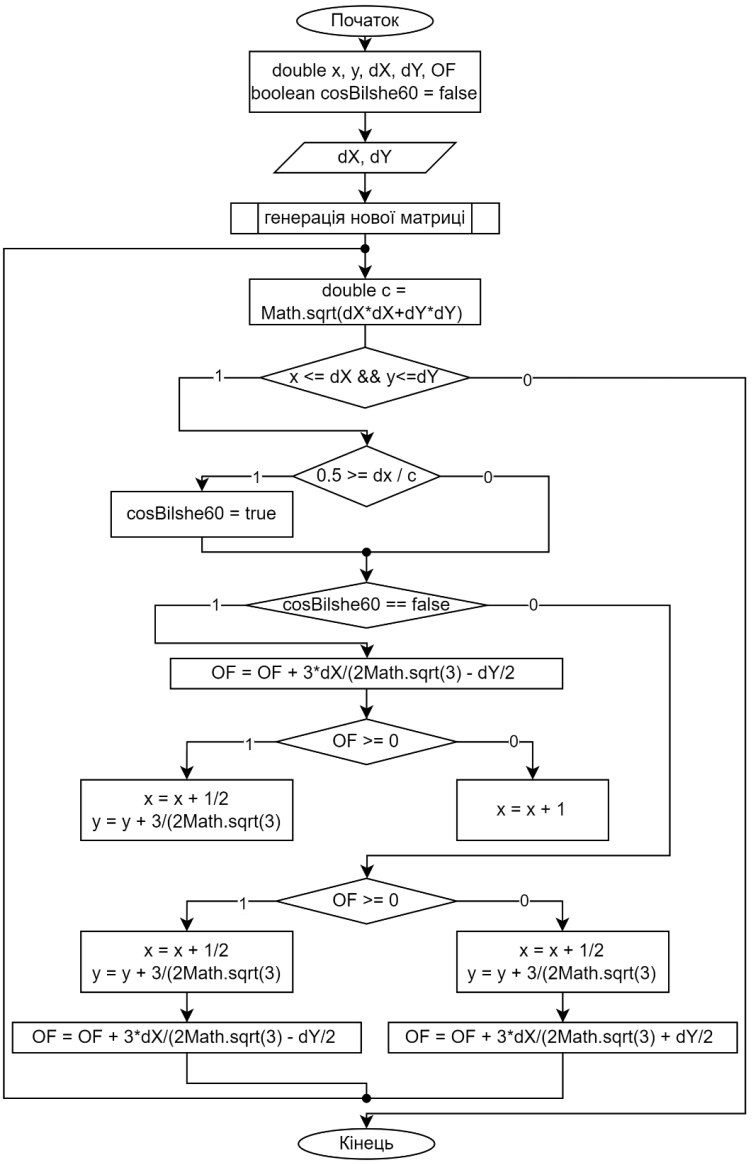


Рисунок 2.8 – Алгоритм формування відрізка

Програмний застосунок «Hexagon Screen» виконує такий алгоритм дій, що зображено на рисунку 2.8. У результаті введення координати [7;4] у програму як кінцевої для формування відрізку користувач отримує результат, що зображено на рисунку 2.9.

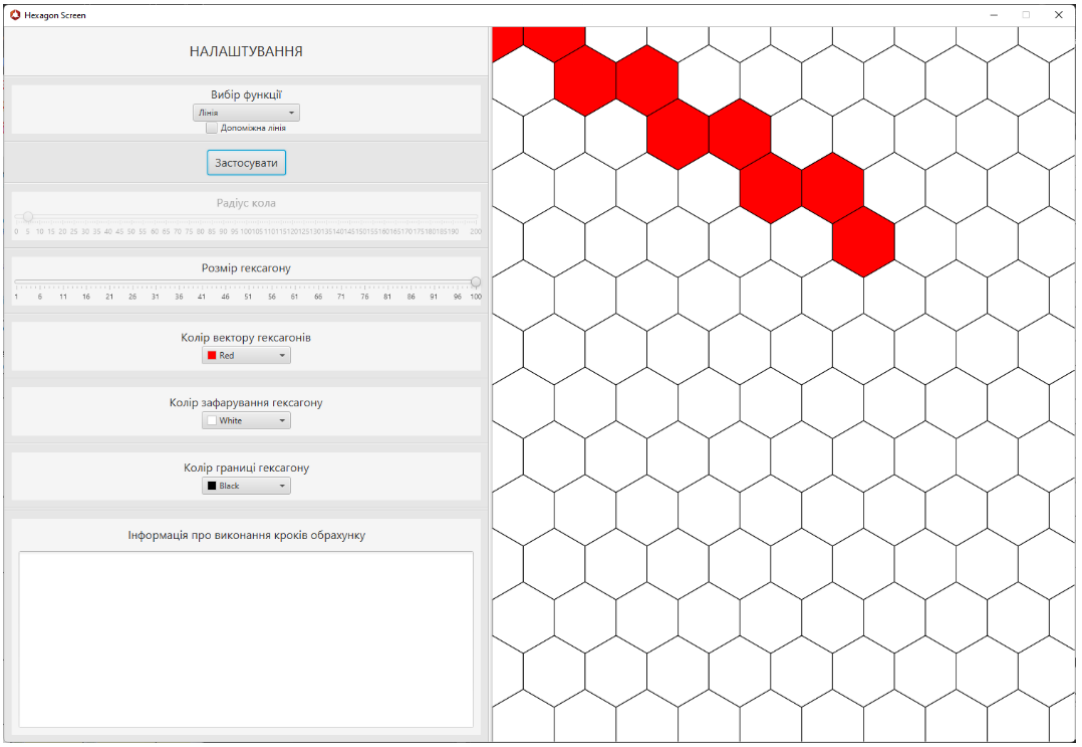


Рисунок 2.9 – Результат виконання програми

Для перевірки максимально можливого результату імітації гексагонального пікселя на екрані користувач може встановити відповідний параметр налаштувань на мінімальне значення. Для того, щоб полегшити взаємодію із користувачем було вирішено зробити імітований екран інтерактивним. Схема алгоритму взаємодії із користувачем зображено на рисунку 2.10.

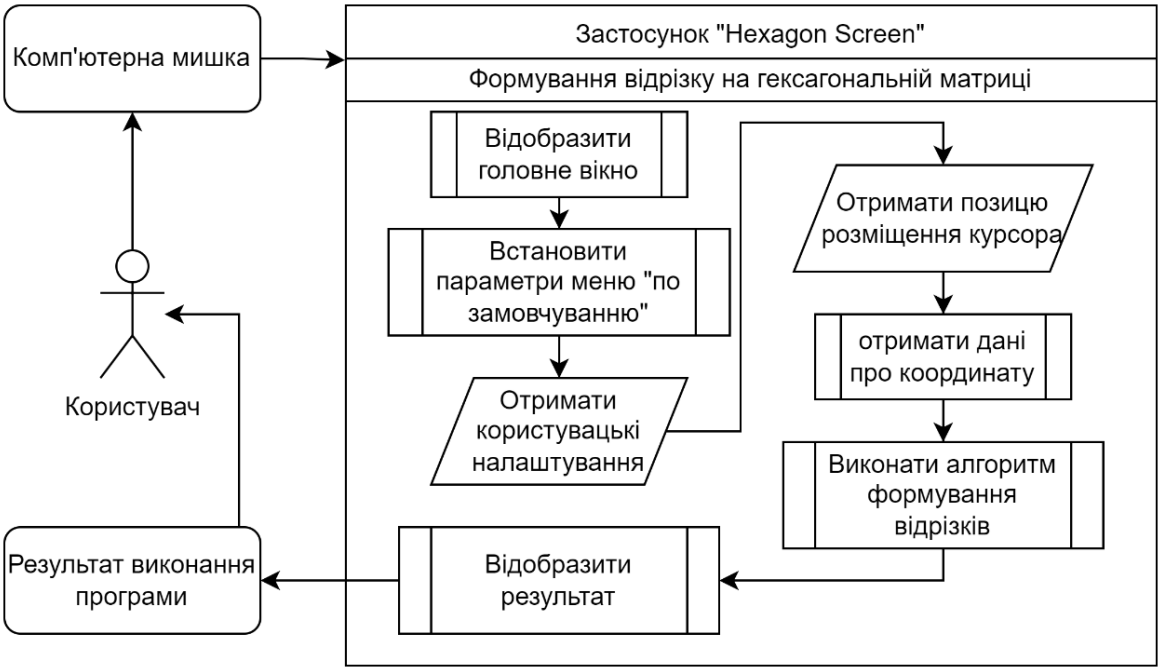


Рисунок 2.10 – Алгоритм взаємодії із користувачем

Таким чином користувач має змогу обирати кінцеву точку на екрані самостійно за допомогою натискання клавіші мишки на гексагони робочого поля. Після виконання даної дії програма створює новий відрізок на основі отримання розташування кінцевої координати пікселя. У результаті програма відображає імітовану матрицю на екрані, як це зображено на рисунку 2.11.

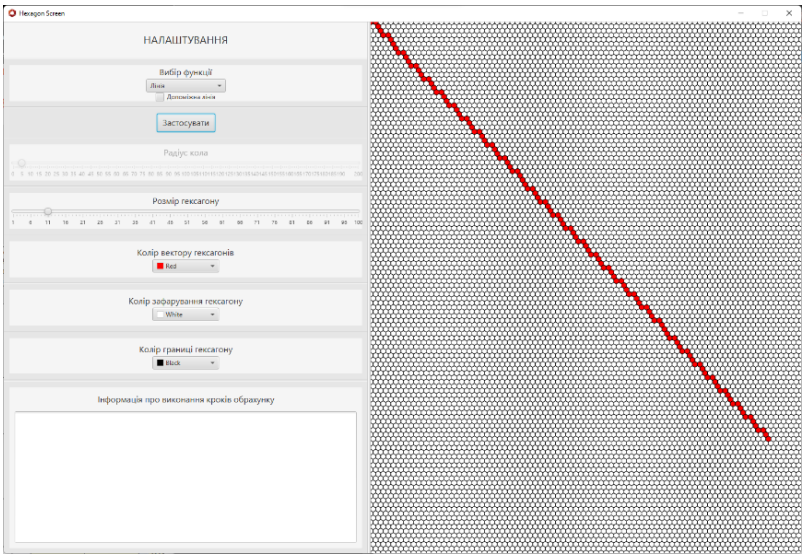


Рисунок 2.11 – Формування відрізка зі розміром гексагону в 11 пікселів

Для зручності можливо встановити такі характеристики відображення гексагонів, що задовольнить потреби користувача. Для прикладу можливо відобразити ситуацію, як зображено на рисунку 2.12, коли потрібно отримати матрицю із розміром гексагону відносно екрану, яким користується користувач, у три піксела та білий екран із зафарбуванням відрізка у синій колір.

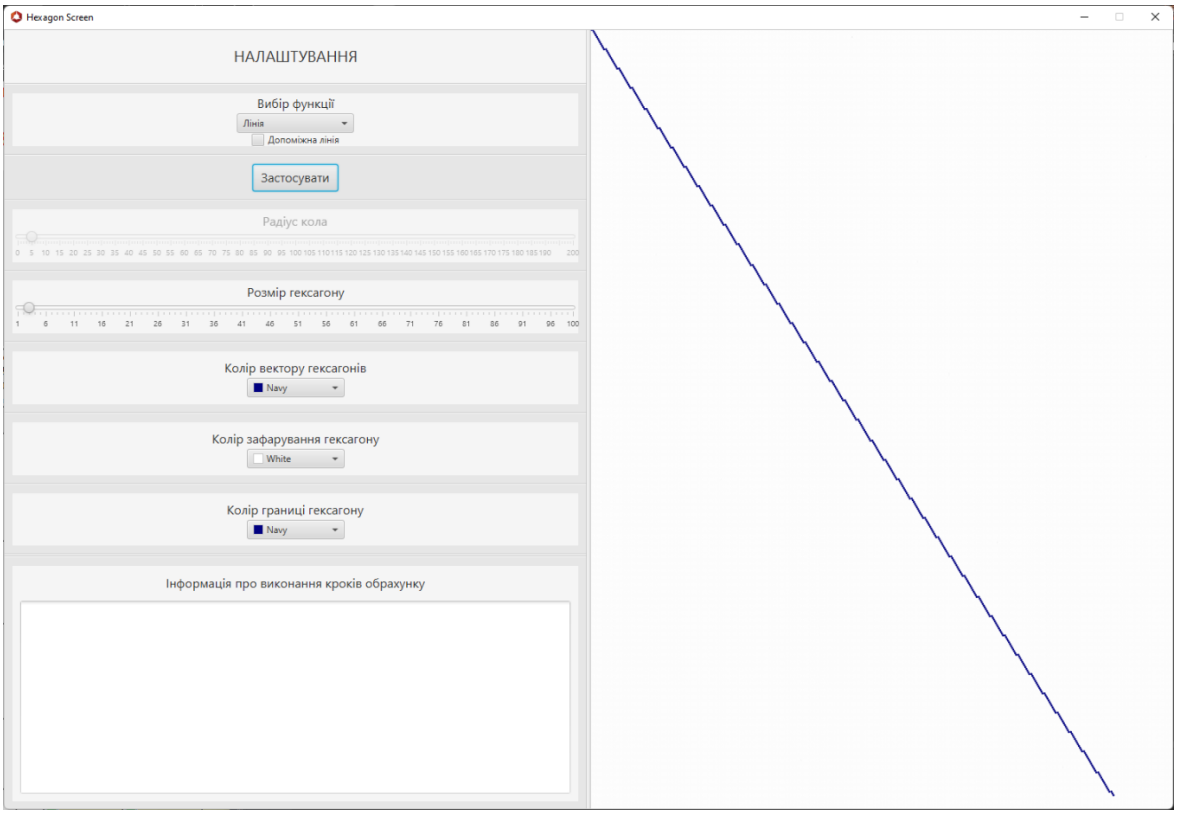


Рисунок 2.12 – Формування відрізка зі розміром гексагону в 3 піксела

Таким чином отримано програмний модуль застосунка «Hexagon Screen», що надає можливість формувати відрізки на основі імітованої матриці гексагонального растру за використання звичної декартової системи відліку [17].

2.3 Формування відрізків прямих на гексогональному растрі комбінованими приростами

Знайдемо вирази для формування комбінованого приросту, який включає горизонтальний та діагональний крок (рис. 2.13).

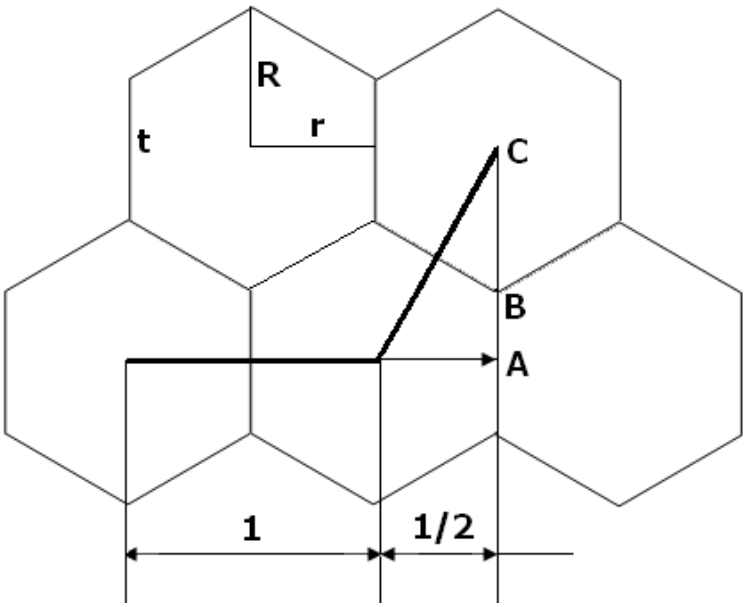


Рисунок 2.13 – Формування крокових переміщень

Відомо, що $R = 2 \div \sqrt{3} \times r$.

З рисунку видно, що $r = \frac{1}{2}$. Тому $R = \frac{1}{\sqrt{3}}$

де, $t = R = \frac{1}{\sqrt{3}}$.

Знайдемо АС

$$AC = BA + BC = R + \frac{t}{2} = \frac{1}{\sqrt{3}} + \frac{1}{2\sqrt{3}} = \frac{3}{2\sqrt{3}} = \frac{\sqrt{3}}{2} \qquad (2.14)$$

На рис. 2.14 зображено вектора, який в точці B знаходиться на одній відстані від точок гексагональної гратки. В цьому випадки в використання точок C і D є рівноцінним. Якщо ця умова буде порушення, то вибирається та піксел, який є найближчим до вектора.

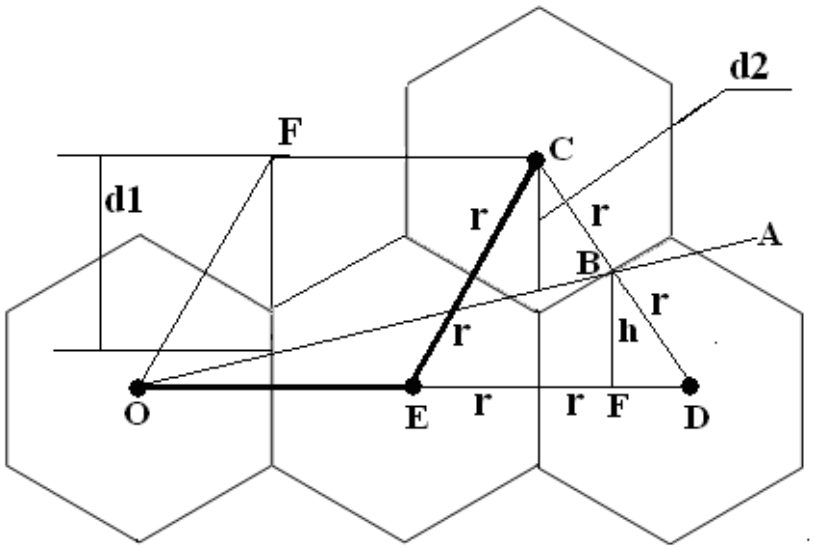


Рисунок 2.14 – Визначення ординарної похибки

Проаналізуємо ординатні похибки $d1^{3}\,d2$. У трикутнику ΔECD сторони рівні. В цьому випадку $< NDE = 60^{\circ}$ Знаходимо h

$$h = BD \cdot sin60^{\circ} = r \cdot sin60^{\circ} = \frac{1}{2} \cdot sin60^{\circ} = \frac{\sqrt{3}}{4} = 0.433 \qquad (2.15)$$

де $h < d2.\ d2 < R + \frac{t}{2} = \frac{3R}{2}$, що менше висоти діагонального переміщення.

Знайдемо кут відрізка прямої (рис. 2.15), який включає горизонтальне та сусіднього з ним діагональне переміщення

$$arctg(AN \div \frac{1}{2}) = arctg\frac{\sqrt{3}}{3} = 30^{\circ} \qquad (2.16)$$

Якщо комбіноване переміщення включає два діагональні кроки, то

$$arctg(AN \div \frac{1}{2}) = arctg\frac{2\sqrt{3}}{2} = 60^{\circ} \qquad (2.17)$$

Отримані вирази є умовою поділу координатного простору. Тому можна костатувати, що на ділянці А1 генерація двох сусідніх діагональних переміщень неможлива. На ділянці А2 неможлива генерація двох горизонтальних переміщень.

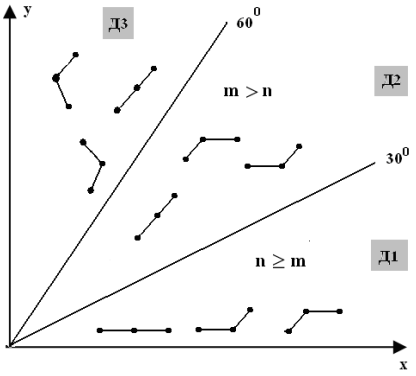


Рисунок 2.15 – Типи допустимих сполучень крокових переміщень

Введемо поняття лівого та правого діагональних переміщень (рис. 2.16).

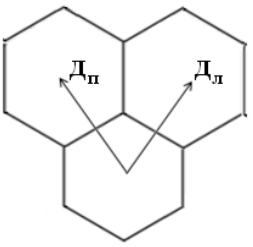


Рисунок 2.16. – Діагональні крок

Для Ае

$$\sigma = \sigma + \frac{1}{2},\ \Delta\sigma = \Delta\sigma + \frac{\sqrt{3}}{2}. \quad (2.18)$$

Для Аї

$$\sigma = \sigma - \frac{1}{2},\ \Delta\sigma = \Delta\sigma + \frac{\sqrt{3}}{2}. \quad (2.19)$$

Доведемо, що для векторів з діапазону ділянки А3 (від 60^0 до 90^0) генерація кроків Дп неможливе.

На рис. 2.17 зображено випадок генерації двох діагональних переміщень

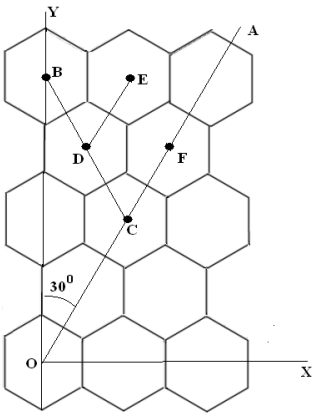


Рисунок 2.17- Формування діагональних переміщень

Нехай вектор перетинає точку С. Кут ХОА дорівнює 30^0 . З точки С, яка розташована на однаковій відстані від точок D і F, можливо виконати переміщення DC і CF. Якщо реалізовано переміщення DC, то з переміщень DE і DE треба реалізувати переміщення DE, який ближче до вектора. Якщо це не так, В протилежному випадку за вектором буде сформовано дві точки, що відхиляються від вектора. Це недопустимо.. При переміщенні променя ОА до осі ОУ зазначений випадок погіршується з точки зору похибки. Отже, при генерації векторів з діапазону від 60^0 до 90^0 реалізація двох суміжних діагональних переміщень недопустимо.

.

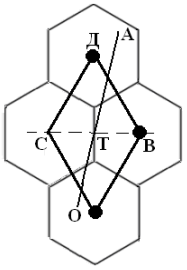


Рисунок 2.18 – Приклад формування діагональних кроків

Як видно з рисунку 2.18 видно, що максимум ординарного відхилення не більший $R + \frac{t}{2} = \frac{3R}{2}$, тобто висоти діагонального кроку.

Знайдемо значення оцінювальної функції для ділянки Аї . При формування двох горизонтальних кроків

$$OF_{i+1} = y_i\Delta x - \Delta y(x_i + 2) = y_i\Delta x - x_i\Delta y - 2\Delta y = OF_i - 2\Delta y \quad (2.20)$$

При формуванні горизонтального та діагонального кроків

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

$$= y_i\Delta x - x_i\Delta y + \frac{\sqrt{3}}{2}\Delta x - \frac{3}{2}\Delta y =$$

$$= OF_i + \frac{\sqrt{3}}{2}\Delta x - \frac{3}{2}\Delta y = (\frac{1}{2} + \frac{1}{16} + \frac{1}{64})\Delta x - (1 + \frac{1}{2})\Delta y = \quad (2.21)$$

$$= \frac{37}{64}\Delta x - \Delta y - \frac{1}{2}\Delta y = \frac{32\Delta x + 4\Delta x + \Delta x}{64} - \frac{2\Delta y + \Delta y}{2}.$$

Розглянемо ділянку А2.

Для випадку двох діагональних кроків

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

$$= OF_i + \sqrt{3}\Delta x - \Delta y = OF_i + \frac{8-1}{4}\Delta x - \Delta y = OF_i + \frac{8\Delta x - \Delta x}{4} - \Delta y. \quad (2.22)$$

Розглянемо ділянку А3.

Для переміщення, яке включає лівий і правий діагональні кроки.

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

$$= OF_i + \sqrt{3}\Delta x - \Delta y = OF_i + \frac{8-1}{4}\Delta x - \Delta y = OF_i + \frac{8\Delta x - \Delta x}{4} - \Delta y. \quad (2.23)$$

При генерації вектора елементарними кроками для ділянки А3 інтерполяція закінчується після формування всіх елементарних кроків по більшій координаті.

Якщо більший приріст $\Delta x = v$, то після реалізації горизонтального переміщення (рис. 2.19, а) цей значення зменшують на 1, а після діагонального – на 0,5. Коли поточний операнд дорівнює 0, то інтерполювання відрізка прямої закінчують.

При використанні подвійних переміщень в якості від'ємника вибирають значення 2 або 1,5 для двох горизонтальних і горизонтального та діагонального переміщень (рис. 2.19, а). Коли $v = 0.$, то інтерполювання завершують. Якщо більший операнд непарний (результат віднімання від'ємний), то в комбінованому переміщенні останній крок не виконують.

Для $A2$ (рис. 2.19, б) і $A1\ \Delta y > \Delta x.$ У цьому випадку мають місце такі цифрові сегменти : два діагональні переміщення ; горизонтальне і діагональне переміщення. Коли виконуються два діагональні переміщення від v віднімають 1.

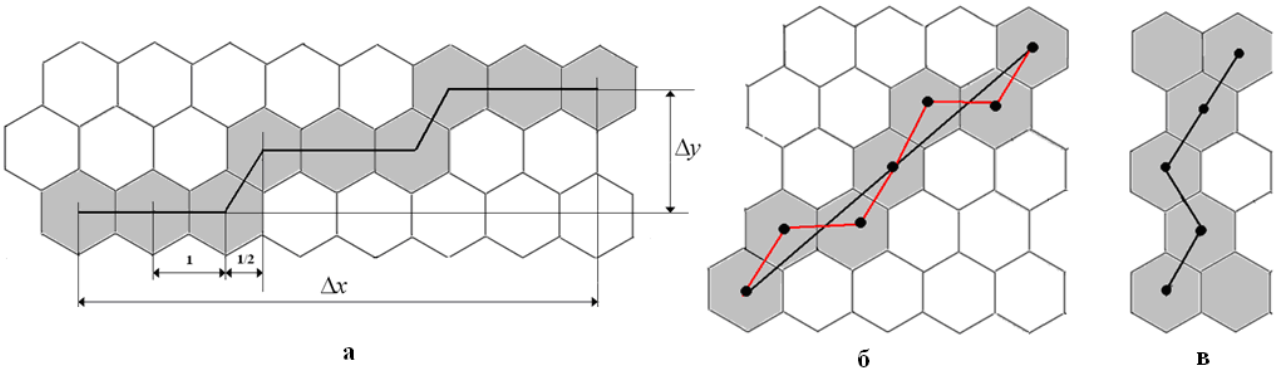


Рисунок 2.19. – Приклади формування векторів в Д1, Д2, Д3

Для $A3\ \Delta y > \Delta x$, тому $v = \Delta y$.

Слід зазначити, що формування відрізків прямих на гексагональному растрі більш трудомісткіше порівняно з прямокутним растром.

Висновки

1. Розглянуто особливості формування відрізків прямих.

2. Запропоновано алгоритм формування векторів на гексагональному растрі за методом оцінювальної функції

3. Вперше запропоновано метод формування векторів на гексагональному растрі, особливість якого полягає у використання комбінованих переміщень, що дозволило до двох разів підвищити продуктивність лінійного інтерполювання.

3 МЕТОД ТА ПРОГРАМНІ ЗАСОБИ ФОРМУВАННЯ КОЛА НА ГЕКСОГОНАЛЬНОМУ РАСТРІ

3.1 Особливості формування кола на прямокутному растрі

Коло - це геометрична фігура, яка є замкнутою. Включає набір точок, які мають однакову відстань до центру, що є радіусом.

Алгоритм малювання кола Bresenham’s Circle Drawing Algorithm — це алгоритм малювання кола, який вибирає найближчу позицію пікселя для завершення дуги.

Було розглянуто формування кола на прямокутному растрі

Формування кола є складнішим ніж для лінії, і використовує алгоритм Брезенхема для побудови кола (рис. 3.1).

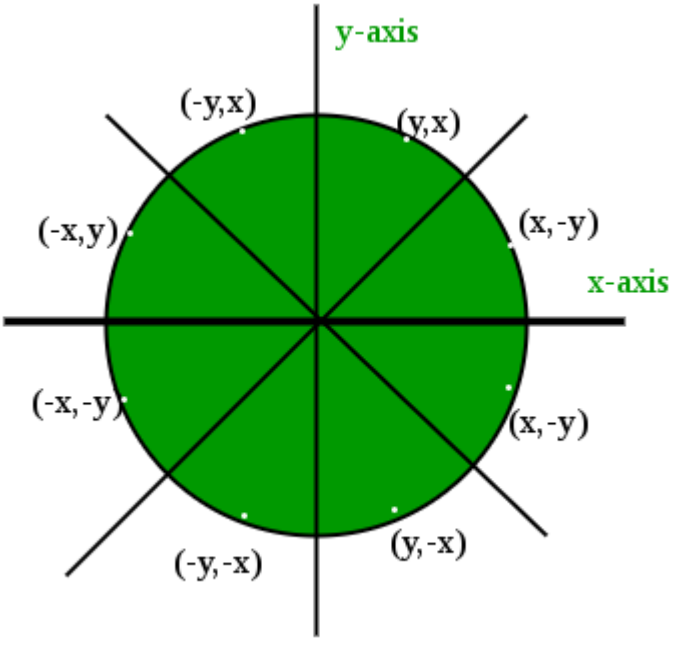


Рисунок 3.1 - Коло на піксельному растрі

Особливість цього алгоритму полягає в тому, що він використовує лише цілочисельну арифметику, що робить його значно швидшим, ніж інші алгоритми, які використовують арифметику з плаваючою комою в класичних процесорах.

Один з найбільш ефективних алгоритмів генерації окружності належить Брезенхему. Необхідно згенерувати тільки одну восьму частину кола [1]. Решта її частини можна отримати послідовними відображеннями. Якщо згенерований перший октант (від 0 до 45 ° проти годинникової стрілки), то другий октант можна отримати дзеркальним відображенням відносно прямої y = x, що дає в сукупності перший квадрант. Перший квадрант відбивається щодо прямий x = 0 для отримання відповідної частини кола в другому квадраті. Верхнє півколо відбивається відносно прямої y = 0 для завершення побудови [1].

Було розглянуто першу чверть кола з центром у початку координат. Якщо робота алгоритму починається в точці (0, r), то коло за годинниковою стрілкою в першому квадранті породжує монотонно спадаючу функцію аргументу x. Аналогічно, якщо початковою точкою є (R, 0), то при формуванні кола проти годинникової стрілки це буде монотонно спадна функція аргументу y. У даному випадку генерація за годинниковою стрілкою відбувається, починаючи з точки (0, R) [2]. Передбачається, що центр кола та початкова точка точно знаходяться в точках сітки.

Існує лише три варіанти вибору наступного пікселя, який найкраще наближає коло. Це означає виконання елементарного кроку в одному з таких напрямків відносно рухомого пікселя: горизонтально вправо, по діагоналі (вниз і вправо) і вертикально вниз. На рисунку 3.2 зображено нижче ці напрямки.

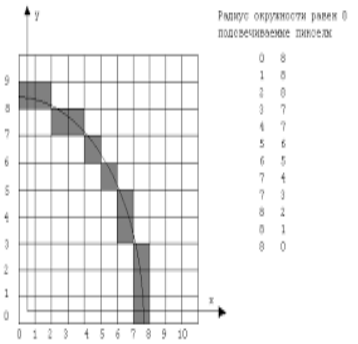


Рисунок 3.2 - Варіанти виконання елементарних кроків

Відповідно до алгоритму Брезенхема для кола вибирається піксель, для якого квадрат відстані між одним із цих положень і колом є мінімальним. Розрахунок можна спростити, якщо врахувати, що навколо точки можливі тільки п'ять типів перетину кола і сітки.

Різниця між квадратами відстаней від центру кола до діагонального пікселя (x_{i+1}, y_{i-1}) і від центру до точки на колі r^2 дорівнює:

$$D_i = x_{i+1}^2 + y_{i-1}^2 - R^2 \quad (3.1)$$

Як і у випадку з сегментом, було використано тільки знак цієї різниці, а не її значення, щоб вибрати відповідний піксель. Якщо $D_i < 0$, то точка знаходиться всередині кола. Зрозуміло, що в цій ситуації варто вибрати піксель з координатами(x_{i+1}, y_{i-1}). Для цього спочатку було розгялнуто випадок 1 і перевірено різницю між квадратами відстаней від кола до пікселів у горизонтальному та діагональному напрямках:

$$d = r_i - s_i = \left|x_{i+1}^2 + y_i^2 - R^2\right| - \left|x_{i+1}^2 + y_{i-1}^2 - R^2\right| \quad (3.2)$$

При $D < 0$ – відстань від кола до діагонального пікселя більша, ніж до горизонтального. І навпаки, якщо $D < 0$, то відстань до горизонтального пікселя більша. Таким чином, при $D < 0$ обирається R_i, а при $D > 0$ – S_i.

У випадку, коли $d = 0$, тобто коли відстань від кола до обох пікселів однакова, обирається горизонтальний крок. Кількість обчислень, необхідних для оцінки величини d, можна скоротити, якщо врахувати, що у випадку *1(3.3)*.

$$x_{i+1}^2 + y_i^2 - R^2 \geq 0;\ x_{i+1}^2 + y_{i-1}^2 - R^2 < 0 \quad (3.3)$$

Це зумовлено тим, що діагональний піксел (x_{i+1}, y_{i-1}) завжди лежить всередині кола, а горизонтальний (x_{i+1}, y_i) – поза ним. Таким чином, d можна обчислити таким чином:

$$d = r_i - s_i = \left|x_{i+1}^2 + y_i^2 - R^2\right| - \left|x_{i+1}^2 + y_{i-1}^2 - R^2\right| \quad (3.4)$$

Коли $D < 0$, то відстань від кола до діагонального пікселя більше, ніж до горизонтального. І навпаки, якщо $D > 0$, то відстань до горизонтального пікселя більша. Таким чином обираєтьсяR_i або – $D > 0 - S_i$. У випадку, коли $d = 0$, тобто коли відстань від кола до обох пікселів однакова, обирається горизонтальний крок. Кількість обчислень, необхідних для оцінки кількості d, можна зменшити, якщо врахувати , що у випадку *1*.

$$x_{i+1}^2 + y_i^2 - R^2 \geq 0;\ x_{i+1}^2 + y_{i-1}^2 - R^2 < 0 \quad (3.5)$$

Це пов'язано з тим, що діагональний піксель завжди знаходиться всередині кола (x_{i+1}, y_{i-1}), а горизонтальний (x_{i+1}, y_i) - поза ним. Таким чином, можна розрахувати такти чином:

$$d = x_{i+1}^2 + y_i^2 - R^2 + x_{i+1}^2 + y_{i-1}^2 - R^2 \quad (3.6)$$

Доповнення до повного квадрату члена y_i^2 та за допомогою додавання і віднімання$- 2y_i + 1$ дає

$$d = 2(x_{i+1}^2 + y_{i-1}^2 - R^2) + 2y_{i-1} \quad (3.7)$$

Можна записати дану формулу так:

$$d = 2(D_i + y_i) - 1 \text{ (3.8)}$$

Було розглянуто випадок *2* (рис.3.2) і зауважено, що тут повинен бути обраним горизонтальний піксель (x_{i+1}, y_i), тому що y є монотонно спадною функцією. Перевірка компонентів d показує, що

$$x_{i+1}^2 + y_i^2 - R^2 < 0 \text{ та } x_{i+1}^2 + y_{i-1}^2 - R^2 < 0 \text{ (3.9)}$$

Оскільки у випадку *2* горизонтальний та діагональний пікселі лежать всередині кола. Якщо $d < 0$, і при використанні того ж самого критерію, що і у випадку *1*, вибирається піксель (x_{i+1}, y_i).

Якщо $D_i > 0$, діагональна точка (x_{i+1}, y_{i-1}) знаходиться за межами дуги кола. Отже, це випадки 3 і 4. У цій ситуації необхідно вибрати пікселі (x_{i+1}, y_{i-1}) або (x_i, y_{i-1}). Як і в попередньому випадку, критерій відбору можна отримати, спочатку розглянувши випадок 3 і перевіривши квадрат різниці відстаней від кола до діагональних і вертикальних пікселів.

$$d = r_i - s_i = \left|x_{i+1}^2 + y_i^2 - R^2\right| - \left|x_{i+1}^2 + y_{i-1}^2 - R^2\right| \text{(3.10)}$$

Для випадку 4 - $x_{i+1}^2 + y_{i-1}^2 - R^2 > 0$ і $x_i^2 + y_{i-1}^2 - R^2 > 0$. Тобто, обидва пікселі знаходяться поза колом. Отже, якщо $d > 0$, то використовується критерії, розроблені для випадку 3. В результаті, потрібно вибирати пікселі з координатами (x_i, y_{i-1}).

Випадок 5. Це відбувається, коли діагональні пікселі (x_{i+1}, y_{i-1}) лежать на колі дорівнюють $d = 0$. Перевірка компонента d показує, що $x_{i+1}^2 + y_{i-1}^2 - R^2 > 0$ і $x_i^2 + y_{i-1}^2 - R^2 = 0$. Тобто для випадку, коли $d > 0$, формуються діагональні пікселі S_i з координатами (x_{i+1}, y_{i-1}).

На рис. 3.3 зображено граф-схема алгоритму формування кола на прямокутному растрі.

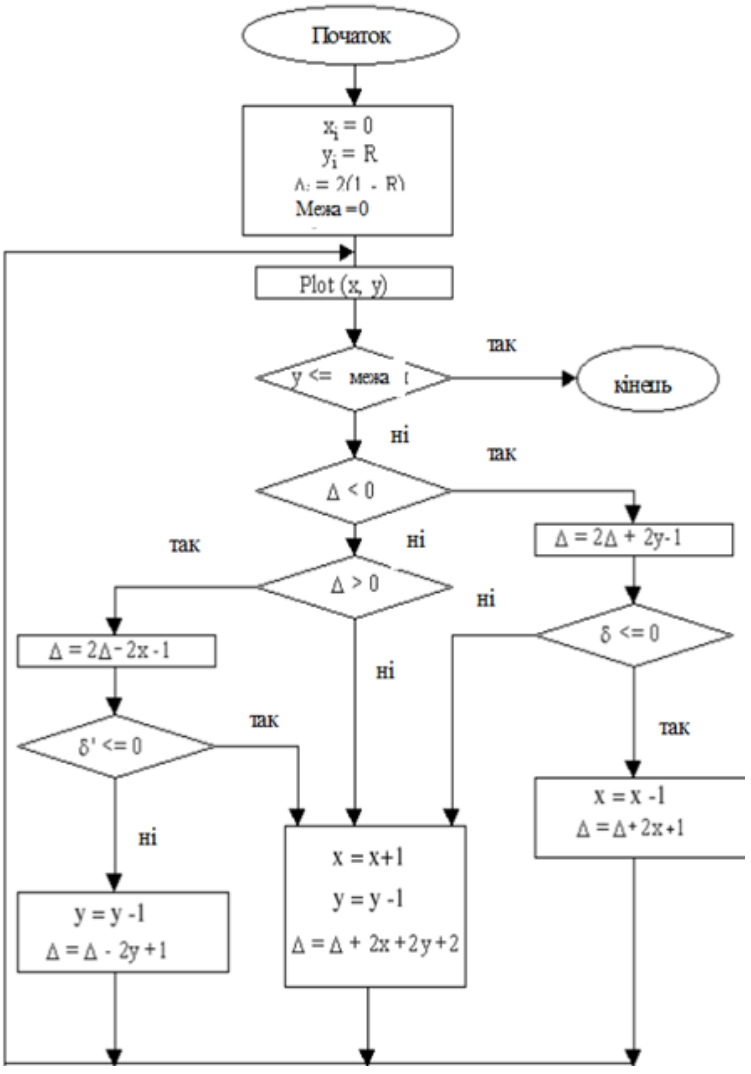


Рисунок 3.3 - Граф-схема алгоритму формування кола на прямокутному растрі

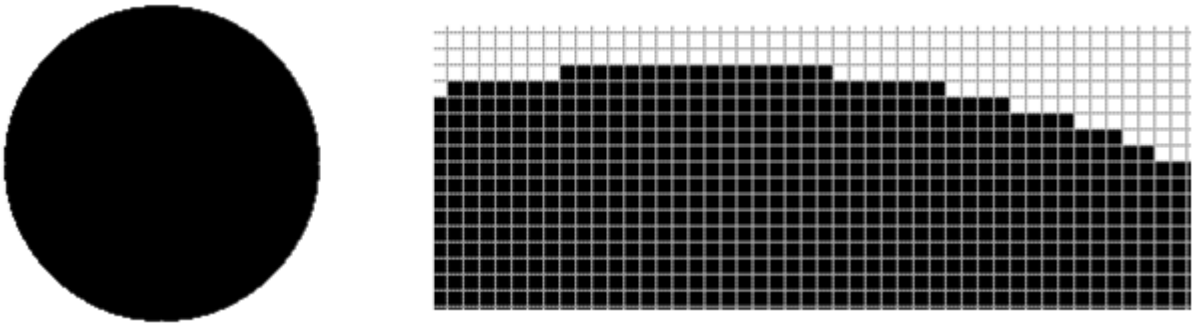


Рисунок 3.4 – Приклад формування кола

3.2 Формування кола на гексагональному растрі

Було розглянуто координати гексагонального растру, взаємозв’язки між гексами (рис. 3.5).

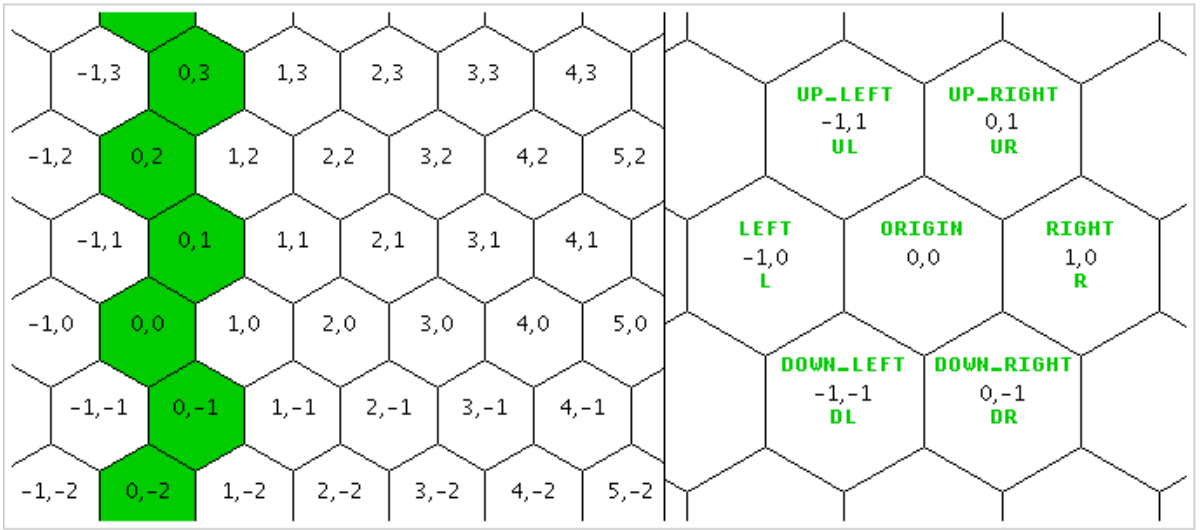


Рисунок 3.5 - Координати гексагонального растру

Гексагональні сітки мають різний тип симетрії, ніж прямокутні.

У 1-му квадранті умова $0°\leq\alpha<60°$ визначає горизонтально зміщену лінію, а $60°\leq\alpha<90°$ визначає вертикально зміщену лінію (для якої будемо вибирати між UL і UR шістнадцятками під час малювання вгору) (рис. 3.6) .

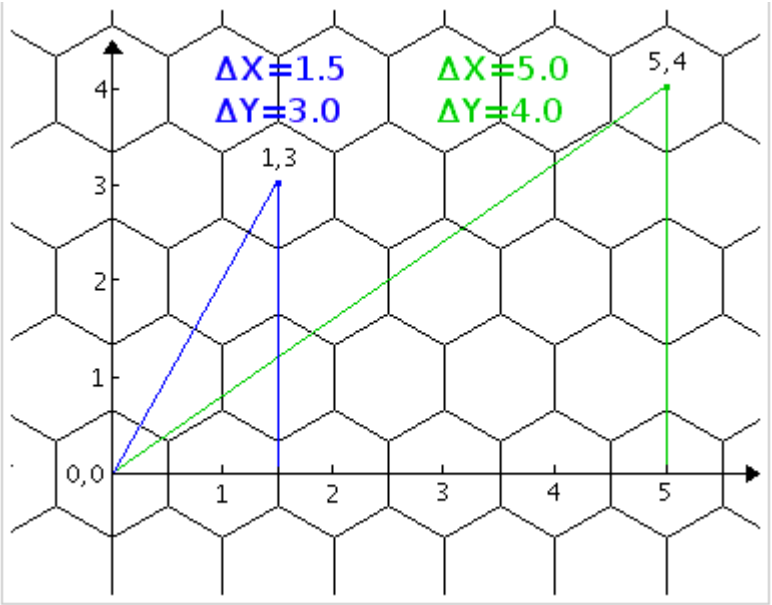


Рисунок 3.6 - Зміщення по гексагональному растрі

На зображенні вище вісь X і Y мають різні масштаби. Використовуючи ці одиниці вимірювання, можна перевірити, чи лінія горизонтально зміщена [12]. Якщо (i_0, j_0) є початковим гексом у гексагональних координатах, а (i_1, j_1)-кінцевим гексом, то лінія є горизонтально зміщеною, якщо $\Delta Y < 2\Delta X$, де $\Delta Y = (i_1 - j_0)$, де ΔX — декартова відстань між початковим і кінцевим гексами. Було прийнято до уваги, що ΔY і 2*ΔX завжди є цілими числами. Синя лінія є прикладом межі між горизонтально та вертикально зміщеними лініями (тобто α=60°).

Розглянемо обчислення 2*ΔX:

$2*\Delta X = 2*((i_1, j_1)-(i_0, j_0))+\text{abs}((i_1, j_1 \bmod 2)-\text{abs}((i_0, j_0 \bmod 2),$

де (i_0, j_0) і (i_1, j_1) — початкова і кінцева шістнадцяткові координати в шестикутних координатах, abs() повертає абсолютне значення.

Тепер, враховуючи початкову та кінцеву точки в шестикутних координатах, було визначено, чи є лінія горизонтально зміщеною, використовуючи лише цілочисельну математику без ділення.

Просторове співвідношення між довільною точкою (x, y) і колом радіусом r з центром у початку координат обчислюється на гексагональній сітці, використовуючи рівняння (3.11)

$$f(x,\ y)\ =\ (\frac{\sqrt{3x}}{2})^{\ 2}\ +\ (y\ +\ \frac{x}{2})^{\ 2}\ -\ r^{2} \quad (3.11)$$

Нехай (x, y) буде поточним пікселем кола з (0, r) як координати його початкової точки. Наступними двома можливими пікселями є (x + 1, y) і (x + 1, y − 1), як показано на рисунку 3.7. Було розглянуто координати середньої точки (так звані mp1 у рисунку 3.7) (x + 1, y − (1 / 2)), на півдорозі між пікселем (x + 1, y) і (x + 1, y − 1).

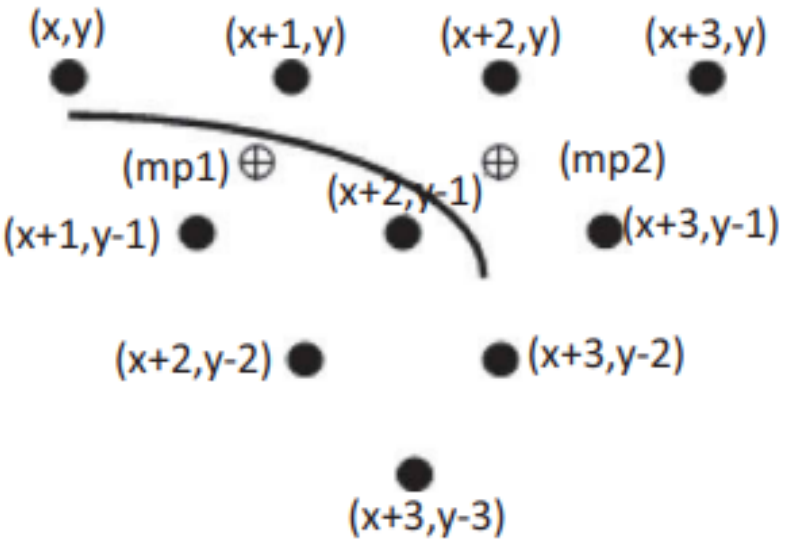


Рисунок 3.7 - Зсув по гексах для кола

На основі рішення вибирається наступний можливий піксель, який обирається через параметр p , який визначається формулою (3.12)

$$p\ =\ f(x\ +\ 1, y\ -\ \frac{1}{2})\ =\ 4(\frac{\sqrt{3}}{2}\ (x\ +\ 1))^{2} \qquad (3.12)$$

Початкове значення параметра отримують шляхом підстановки точки (0, r) у рівнянні (2), яка виробляє p = 3.

Випадок 1: якщо p < 0, то піксель (x + 1, y) є ближче до дуги кола, ніж піксель (x + 1,y − 1). Наступний можливий піксель: (x + 2, y) або (x + 2,y − 1). Щоб визначити, який із двох пікселів ближче до дуги кола, визначемо, чи є дуга кола вище середньої точка між двома пікселями (називається mp2 у рисунку *). З метою аналізу параметра p у рівнянні (3.13) враховуємо умову mp2.

$$p = p - 8x - 4y - 10 \qquad (3.13)$$

Отже побудова кола на гексагональному растрі залежить від того, з яким октаном проводяться операції, в яких межах, і відносно того, де лежить точка, вище кривої, чи нижче (m1, m2), в залежності від цього будуть використовуватись різні алгоритми.

Варто побудувати коло тільки в одному октані, тоді буде можливо перенести коло в інший октант.

За умови формування частини кола в діапазоні від 0° до 30° крок по пікселях може виконуватися за двома напрямами: діагонально праворуч і діагонально ліворуч, як це зображено на рисунку 3.8.

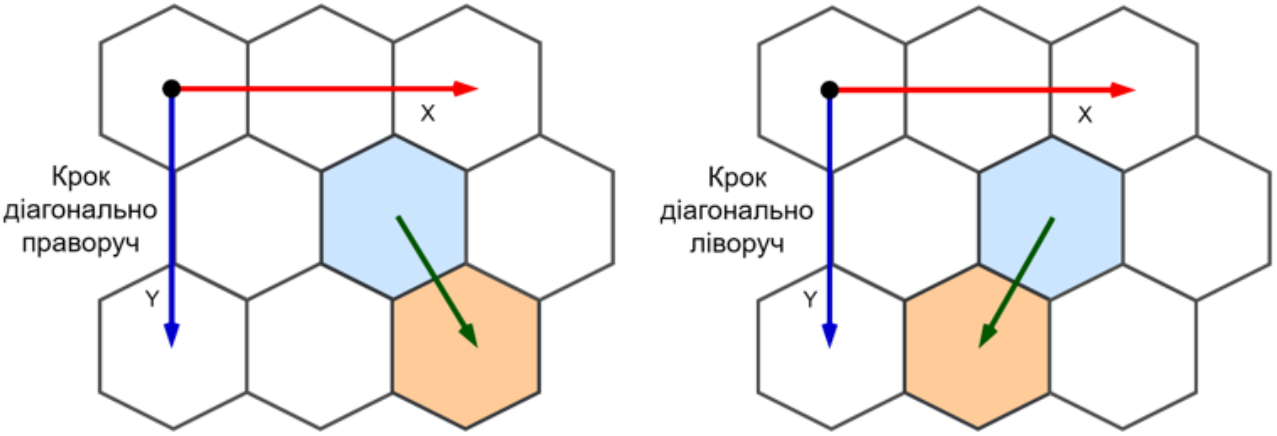


Рисунок 3.8 – Можливі кроки при формуванні сектора від 0° до 30°

За умови формування частини кола в діапазоні від 30° до 90° крок по пікселях може виконуватися за двома напрямами: крок ліворуч і діагонально ліворуч, як це зображено на рисунку 3.9.

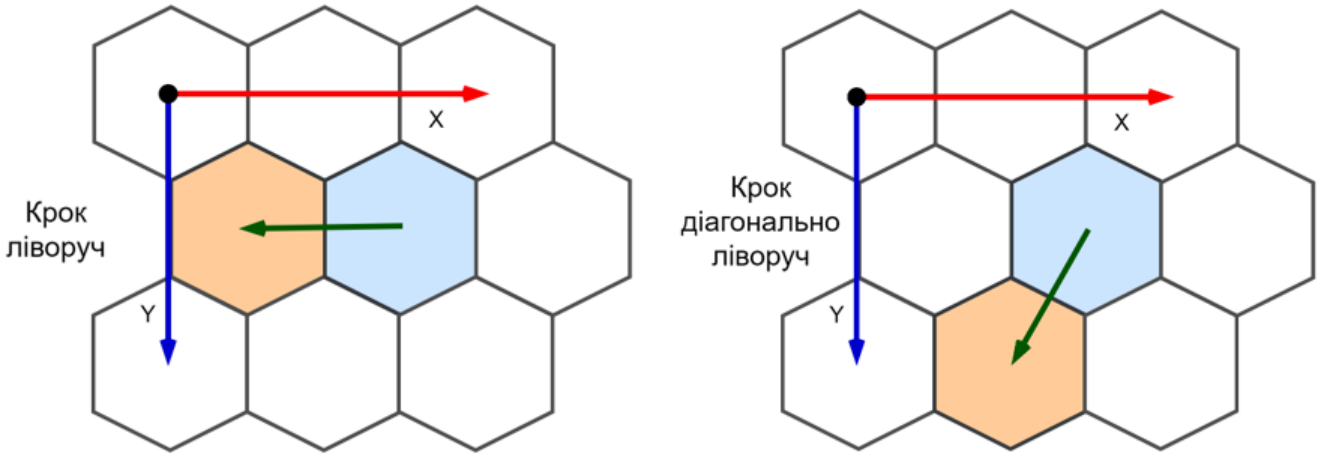


Рисунок 3.9 – Можливі кроки при формуванні сектора від 30° до 90°

На кожному кроці інтерполяції, доцільно вибирати той піксел, для якого є мінімальним значення квадрату відстані між пікселем і точкою на колі.

Для виконання обрахунків сектора 0° до 30° використовується дві формули:

$$I_{\downarrow}DL = \left| \left(\left| (x \right]_{\downarrow} i - 1/2) \right]^{\uparrow} 2 + \left(\left[y_{\downarrow} i + 3/(2\sqrt{3}) \right]^{\uparrow} 2 \right) - R^{\uparrow} 2 \right|$$

Для виконання обрахунків сектора 30° до 90° використовується такі дві формули:

$$I_{\downarrow}HL = \left| \left(\left| (x \right]_{\downarrow} i - 1) \right]^{\uparrow} 2 + \left[y_{\downarrow} i \right]^{\uparrow} 2 \right) - R^{\uparrow} 2 \right|$$

$$I_{\downarrow}DL = \left| \left(\left| (x \right]_{\downarrow} i - 1/2) \right]^{\uparrow} 2 + \left(\left[y_{\downarrow} i - 3/(2\sqrt{3}) \right]^{\uparrow} 2 \right) - R^{\uparrow} 2 \right| .$$

Залежно від результату порівняння оцінювальних функцій обирається такий крок інтерполяції, який відповідає найменшому значенню оцінювальних функцій.

Таким чином отримується рішення задачі формування кола на гексагональному растрі [9]. Коло формується в одній чверті та відображається дзеркально відносно центру кола, базової вісі абсис і ординат.

При формуванні кола із параметрами імітованого гексагонального пікселя можливо отримати відповідне зображення. Таким чином задавши радіус кола у 5 гексагонів і встановивши розмір гексагону в 90 пікселів квадратного екрану користувача можливо отримати коло, що зображено на рисунку 3.10.

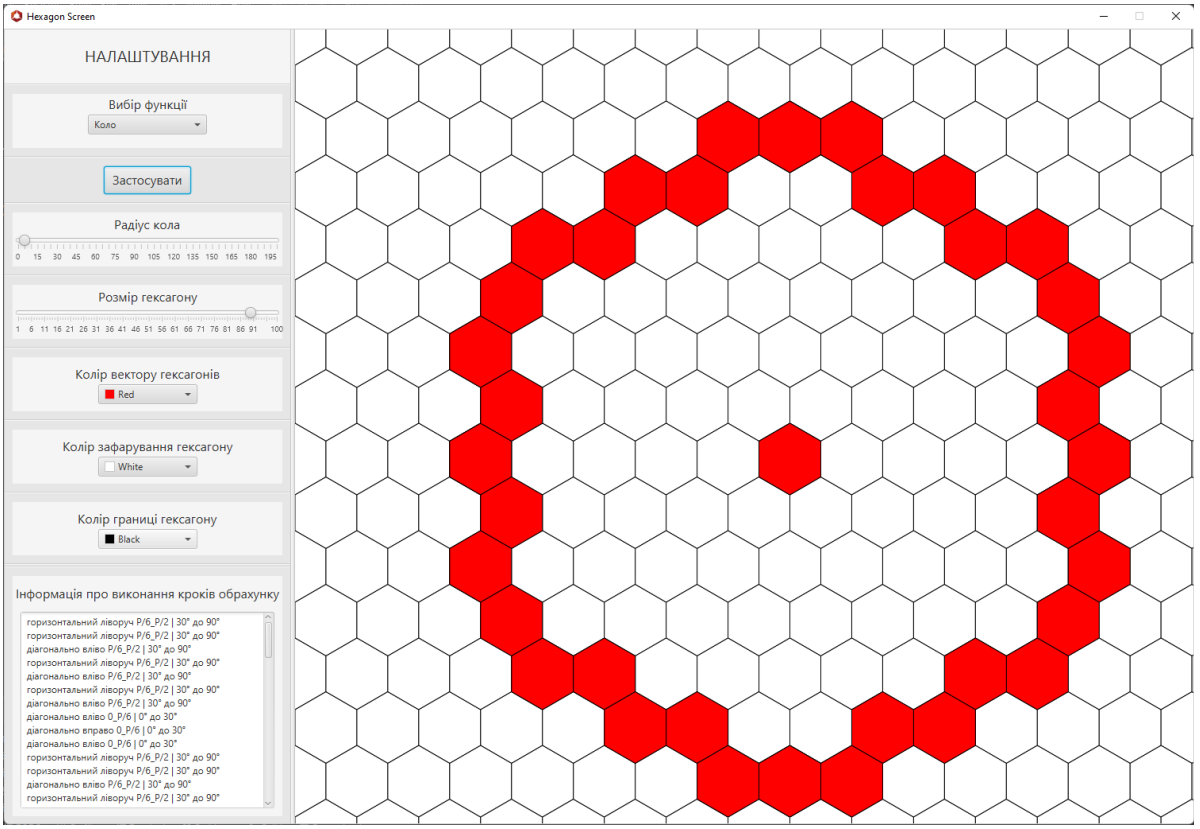


Рисунок 3.10 – Результат формування кола у великому масштабі

За умови використання даного методу формування кола можливо досягти результату, коли око майже не буде сприймати дискретність гексагонального пікселя. Таке коло та налаштування, що забезпечують формування картинки, зображено на рисунку 3.11.

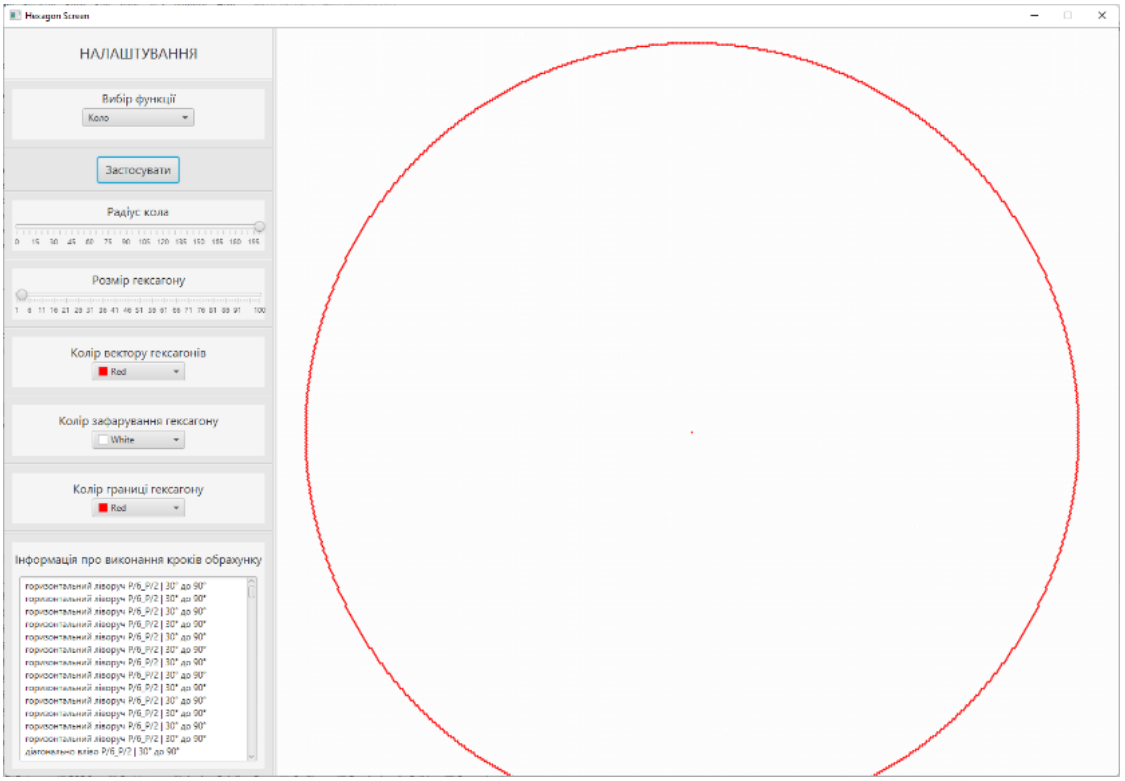


Рисунок 3.11 – Результат формування кола у малому масштабі гексагону

Таким чином, розроблено програмний модуль», що надає можливість формувати коло на основі імітованої матриці гексагонального растру за використання звичної декартової системи відліку. Така реалізація дає можливість для подальшого проведення дослідження із гексагональними матрицями довільного розміру пікселя та налаштувань параметрів кола.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ і ПРОГРАМНИХ ЗАСОБІВ ФОРМУВАННЯ ЗОБРАЖЕНЬ НА ГЕКСОГОНАЛЬНОМУ РАСТРІ

4.1 Обґрунтування вибору засобів реалізації

Використовуючи теоретичні результати магістерської роботи було створено програмний додаток, який реалізує програмні методи та програмні засоби для створення редактора, який має гексагональний растр (рис. 4.1).

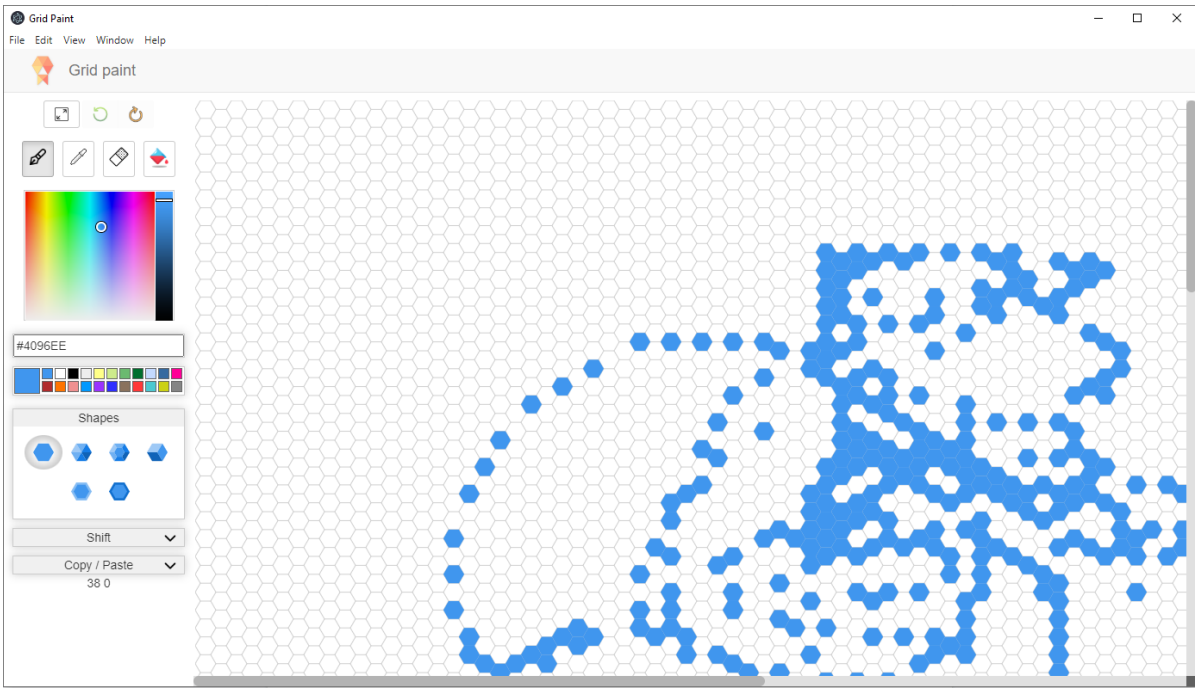


Рисунок 4.1 - Редактор на гексагональному растрі

Розробка додатку є дуже складним процесом. Але завдяки компонентному та модульному підходові, підтримка, розширення та розробка додатку стає простішою, а архітектура простішою.

Для створення гексагонального редактора було використано технологію Electron (рис. 4.2), яка дозволяє створювати програмні додатки на мультиплатформу, незалежно від типу ядра та операційної системи [23].

Рисунок 4.2 - Іконка технології Electron

Electron - це фреймворк для розробки десктопних програм з використанням HTML, CSS і JavaScript. У двійковий код Electron вже вбудовані Chromium і Node.js. Це дозволяє підтримувати тільки JavaScript код і створювати кросплатформні програми, які працюватимуть як на Windows, так і на mac OS і Linux.

Дана технологія є відносно новою на ринку, але уже має велику популярність серед програмістів. Підготовка середовища займає доволі таки мало часу, і встановлюється за допомогою NPM (Node Package Manager) [4]. Також завдяки NPM є можливість встановлювати багато бібліотек, і швидко використовувати їх, що зменшує час на розробку. Також є змога згенерувати додаток на мобільні телефони з операційними системами IOS та Android, але для цього потрібно внести незначні зміни, які будуть становити приблизно 3-5% від всієї кодової бази додатку, що робить додаток зручним. Electron має такі переваги:

- Кросплатформність (Windows, Linux, MacOS);
- Багатий набір як вбудованих, так і сторонніх компонентів;
- Кастомізованість компонентів;
- Наявність мови розмітки для опису інтерфейсу;

- Хороша підтримка.

Багато компаній використовує дану технологію, такі як:

- Slack;
- Skype;
- Discord;
- VSCode;
- Atom;
- Postman;
- Insomnia.

Дана технологія має недолік. Це використання більше ресурсів (10-15%) ніж додаток наприклад на C# (Windows Forms), але наразі потужності комп’ютерів дуже високі, тому цей недолік не помітний для користувача. З іншої точки зору, швидкість розробки є достатньо високою.

JavaScript® (часто просто JS) – це легковажна, інтерпретована або JIT-компілювана, об'єктно-орієнтована мова з функціями першого класу. Найширше застосування знаходить мову сценаріїв веб-сторінок, але також використовується і в інших програмних продуктах, наприклад node.js або Apache CouchDB. JavaScript це прототипно-орієнтована, мультипарадигменна мова з динамічною типізацією, яка підтримує об'єктно-орієнтовану, імперативну та декларативну (наприклад, функціональне програмування) стилі програмування 20.

Було обрано мову програмування JavaScript, яка на даний момент є найбільш поширеною мовою програмування у світі, завдяки своїм особливостям (рис. 4.3).

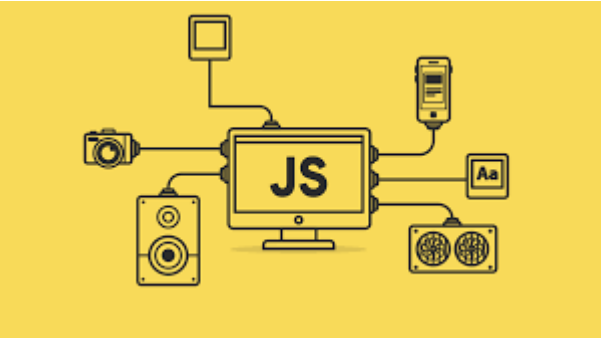


Рисунок 4.3 - JavaScript

До переваг розробки на мові програмування JavaScript можна віднести такі пункти:

- швидкість розробки;
- зручність розгортання середовища програмування;
- мнлжина бібліотек. які пришвидшують розробку, завдяки технології NodeJs, та Node Package Manager;
- можливість писати JavaScript на мові програмування TypeScript, яка є строго типізованою, відносно JavaScript, який в свою чергу є динамічно типізований;
- Web Api - технологія, яка дає змогу використовувати однопоточну мову програмування, як багатопоточну, в особливості завдяки Event Loop;
- використовувати canvas, який дає змогу малювати;
- функціональна парадигма;
- кросплатформеність завдяки певним технологіям (мобільний додаток, веб-додаток, додаток на пк).

Завдяки перевагам даної мови програмування даний додаток було створено за досить короткий час, всього 2 місяці, тільки одним розробником.

Також було використано HTML, який дозволяє робити розмітку блоків, структурувати візуальні блоки, давати певний доступ до обробки даних, використовувати дата-атрибути, для надання додаткової функціональності. Дана

мова розмітки є основним будівельним блоком Веба, і всіх додатків, які використовують мову розмітки. HTML використовуються також для відображення текста, малюнків, вхідних полів та інших елементів. HTML 5 також дозволяє писати теги семантично, що краще для пошукових систем, і надає можливість працювати з мультимедіа та звуком, має покращений accessibility. Завантажується першим та потім завантажує інші файли, які підключені у певному порядку. При використані Electron, будує структуру вікна, приблизно таким самим шляхом, як у браузері, при цьому передоводячи html в інший формат файлу.

GitHub — один з найбільших веб сервісів для спільної розробки програмного забезпечення. Існують безкоштовні та платні тарифні плани користування сайтом. Базується на системі керування версіями Git і розроблений на Ruby on Rails і Erlang компанією GitHub, Inc (раніше Logical Awesome) [18]. Сервіс безкоштовний для проектів з відкритим вихідним кодом, з наданням користувачам усіх своїх можливостей (включаючи SSL), а для окремих індивідуальних проектів пропонуються різні платні тарифні плани. 21 вересня 2011 року кількість користувачів стала більшою за мільйон.

Також для полегшення розробки було застосовано GITHUB репозиторій, який дозволяє зберігати проект на віддаленому сервері без усіх залежностей, які можна встановити, коли завантажуєш проект (рис. 4.4). Репозиторій корисний, як і для команди розробників та і для одного розробника. Завдяки репозиторію створюються різні копію файлів, з якими легко працювати. Кожен commit має свої зміни, і до них можна повернутися в будь-який час.

Рисунок 4.4 - GitHub репозитарій

CSS - використовується для наданню вузлам HTML певної стилізації та покращує вигляд. Також дає доступ до обрабки певних елементів. Це також не є мовою програмування, а лише мова таблиці стилів. На справді це дає можливість застосовувати певні стилі до певних елементів, які в свою чергу дають можливість також створити ієрархію стилів та перевизначити один одного. Даний файл підключаєтсья до html файлу, який буде підгружати стилі, після формування усіх вузлів у html (рис. 4.5).

Рисунок - 4.5 CSS

4.2 Функціональність додатку

Даний додаток підтримує функціональність простого редактора. В даному редакторі полотно є гексагональною сіткою (рис. 4.6).

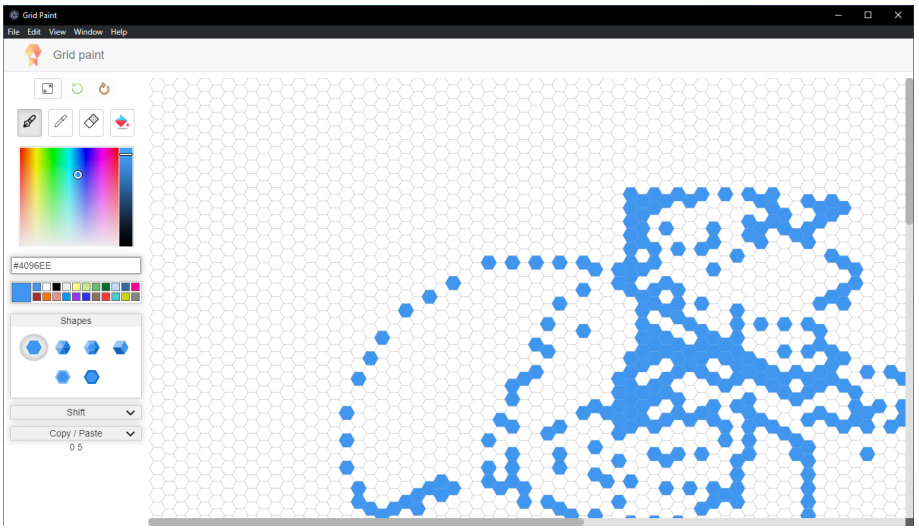


Рисунок 4.6 - Додаток “Grid Paint”

Додаток має бокове меню, де можна побачити основну функціональність програми (рис. 4.7).

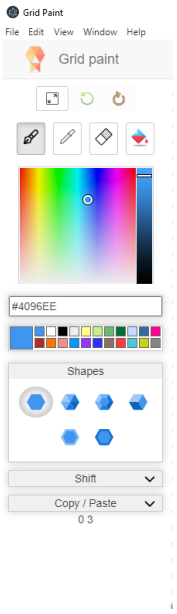


Рисунок 4.7 - Бокове меню у графічному редакторі

У верхньому меню є три кнопки (рис. 4.8), які відповідають за розмір гексагональної сітки, а також розмір гекса, колір меж гексів і т.д . Наступна кнопка це відміна останніх змін. Також є кнопка, яка дозволяє повернутися до останніх

змін.

Рисунок 4.8 - Меню керування гексагональним растром

Кнопка реверсивного повертання, або відновлення останніх змін і повернення до попередніх змін, контролюється програмним забезпеченням та обмежується статичною довжиною масиву для збереження змін на гексагональному растрі. Статична довжина масиву дорівнює 10. Було намальовано 6 гексів на гексагональному растрі, з різними алгоритмами заливки гекса (рис. 4.9).

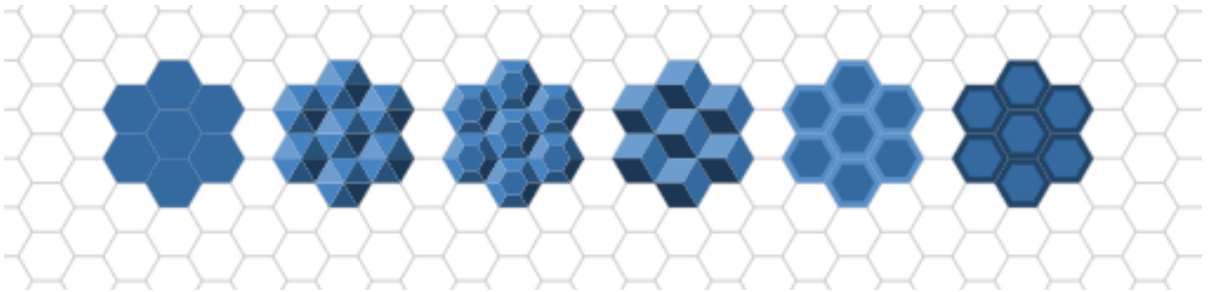


Рисунок 4.9 - 6 гексів з різними заливками

Далі було тричі натиснуто на кнопку відміни останніх дій, що має видалити останні зміни, і , як результат, було видалено три останніх малих гекса з останнього великого гекса, при умові, що кожний гекс був замальований окремо (рис. 4.10).

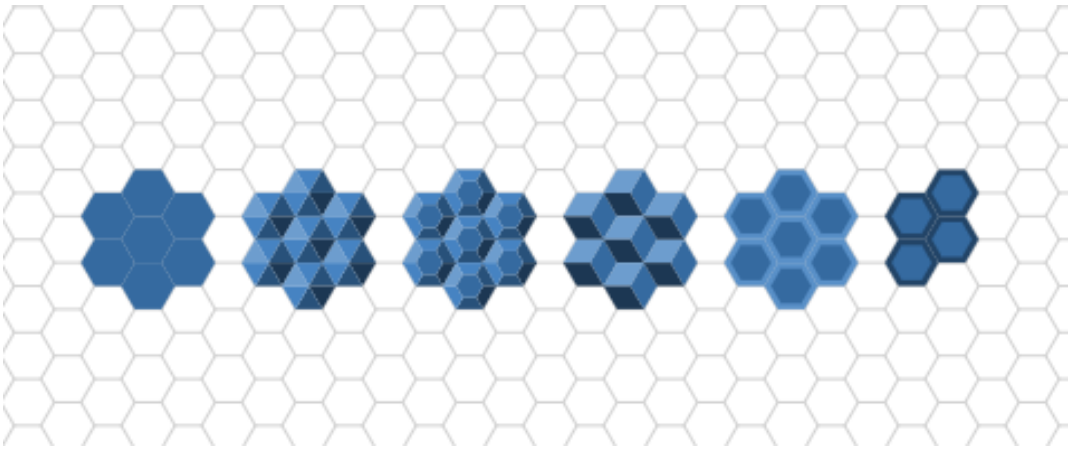


Рисунок 4.10 - Відміна останніх змін тричі

Також додаток має можливість повертати зміни цілих ліній, фігур. Якщо гекси замальовувати не окремо, а групами, то алгоритм буде сприймати це як один крок і записувати до масиву об’єкт з інформацією (рис. 4.11).

Рисунок 4.11 - Відміна останніх змін тричі (замальовування групами)

Якщо відкрити меню керуванням розмірами гексагонального растру, то буде відображено вікно з поточними властивостями, які можна буде змінити у рамках обмежень (рис. 4.12).

Рисунок 4.12 - Меню керування властивостями робочої зони

Програмний додаток також містить певний інструментарій: пензлик, піпетка, гумка для стирання, а також заливка (рис. 4.13) [16]. Даний інструментарій призначений для роботи на гексагональному растрі. Пензлик відповідає за замальовування гексів, на яких був курсор. Піпетка дозволяє отримати колір, натиснувши на певний гекс. Гумка відміняє всі стилі на для гексів, тобто повертає в початковий стан. Заливка працює тільки тоді, коли здійснюється натиснення на гекс, який оточений усіма залитими гексами, на будь-якій дистанції.

Рисунок 4.13 - інструментарій

При використанні пензлика, іконка була замальована в сірий колір. Існує два режими зафарбовування. Перший режим дозволяє кліком зафарбовувати один гекс, а другий режим при натисканні лівої кнопки миші проводити по гексам, і в результаті, дані гекси будуть зафарбовані (рис. 4.14).

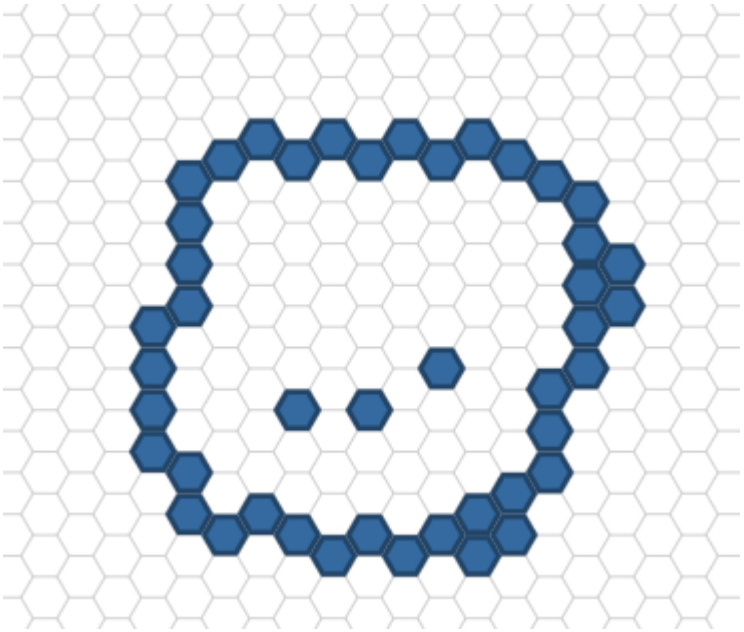


Рисунок 4.14 - Функціональність пензлика, і два режими зафарбовування

Піпетка - надає можливість обрати колір певного гекса та встановити цей колір для пензлика. Після цього пензлик буде обрано та буде застосовано даний колір (рис. 4.15).

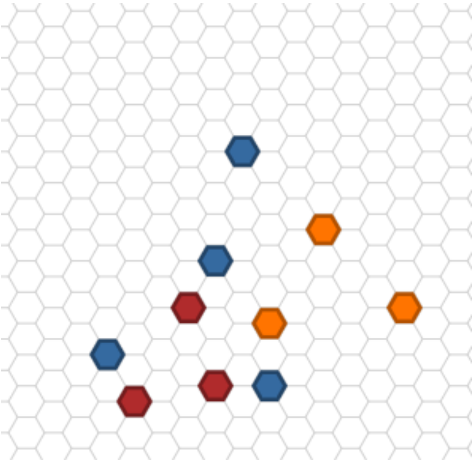


Рисунок 4.15 - Функціональність піпетки

Гумка дає можливість забирати певний колір з гекса, та залишати базовий

колір гексагонального растру (рис. 4.16).

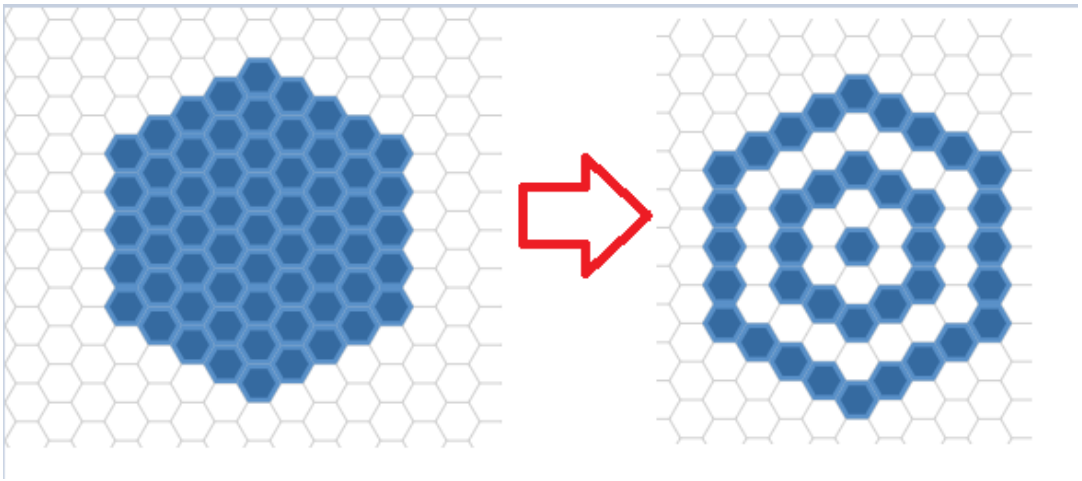


Рисунок 4.16 - Функціональність гумки

Для обрання певного кольору, щоб замальовувати гекси, було створено меню, де є кілька варіантів вибрати колір. Перший - це коли користувач здатний обрати, просто навівши на картинці курсор на певний колір. Наступний варіант ввести у поле шістнадцятковий код обраного кольору. І останній варіант це просто обрати колір з палітри основних кольорів (рис. 4.17).

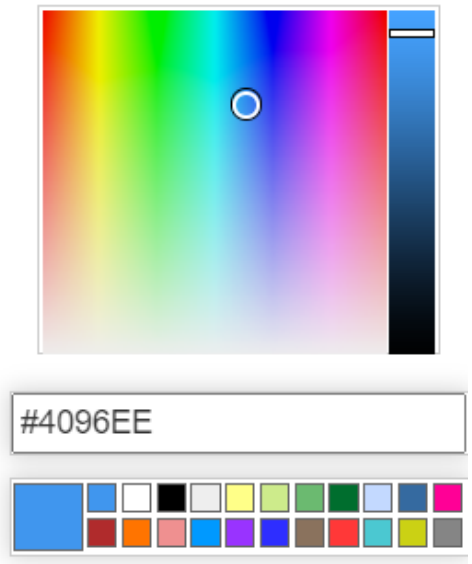


Рисунок 4.17 - Меню обрання кольору

Також додаток дає змогу обрати спосіб замалювання гекса, який дає можливість покращити малювання у редакторі (рис. 4.18).

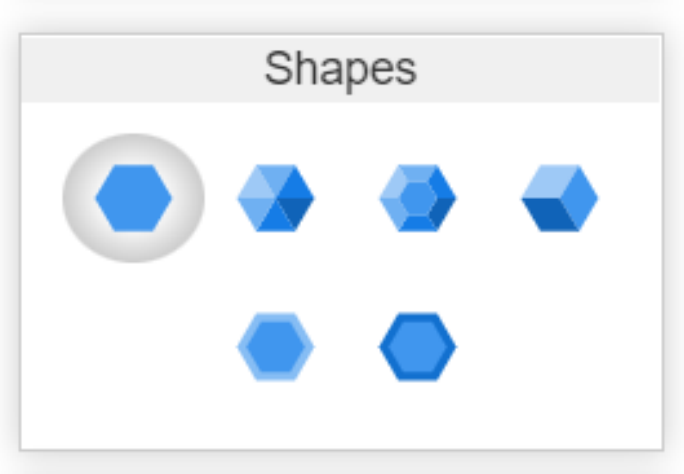


Рисунок 4.18 - Моделі замалювання гексів

Зміна положення малюнку або всіх залитих кольором гексів може здійснюватись через меню. Якщо відкрити випадаюче меню та натиснути на певні клавіші, то зображення буде змінювати своє положення. (рис. 4.19)

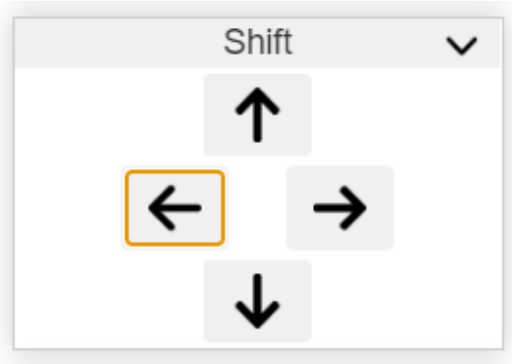


Рисунок 4.19 - Меню зміни положення зображення

Завдяки цій функціональності можна швидко, без копіювання, змістити зображення у горизонтальній або вертикальній проекції (рис. 4.20).

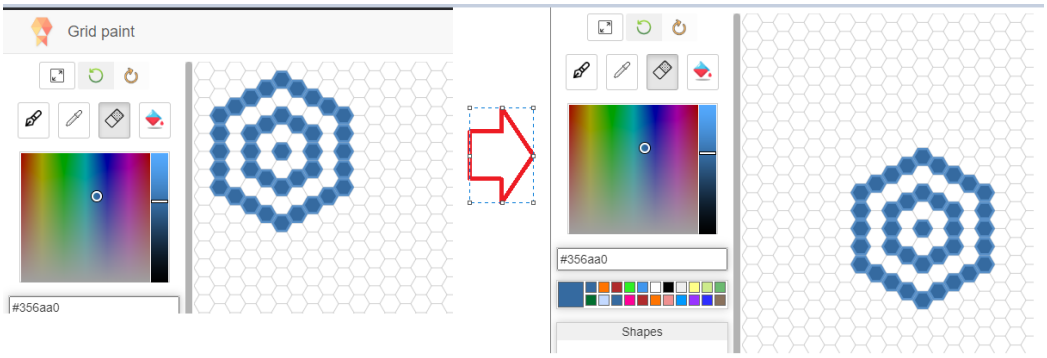


Рисунок 4.20 - Зміна положення зображення

Отже даний додаток був розроблений з базовою функціональністю та надає змогу створювати зображення, які можуть бути досить складними.

4.3 Реалізація алгоритмів додатку

Програма була розроблена з великою кількістю алгоритмів, більшість з яких призначені для графіки. Алгоритми - це певні послідовні операції (кроки), які в результаті свого виконання дають певний результат роботи. Зазвичай алгоритми використовуються для графіки, обробки даних. Сама програма реалізує великий алгоритм зі своїми розгалуженими варіантами виконання. Будь-який алгоритм складається з простих операцій. Даний додаток містить дуже багато алгоритмів, в основному для графіки. Алгоритми були застосовані до таких основних операцій:

- малювання гекса;
- створення гексагональної сітки з параметрами;
- заливання гекса певним кольором та схемою;
- зміна положення зображення;
- відміна або повернення до попередніх операцій;
- заливка гекса або групи гексів певним кольором, якщо навколо є межі з гексівю

Алгоритм обробки кліків по гексам, базований на вибраному модові, за умови, що мод дорівнює пензлику, то буде виконуватись певний алгоритм. Блок-схема алгоритму на рисунку 4.10.

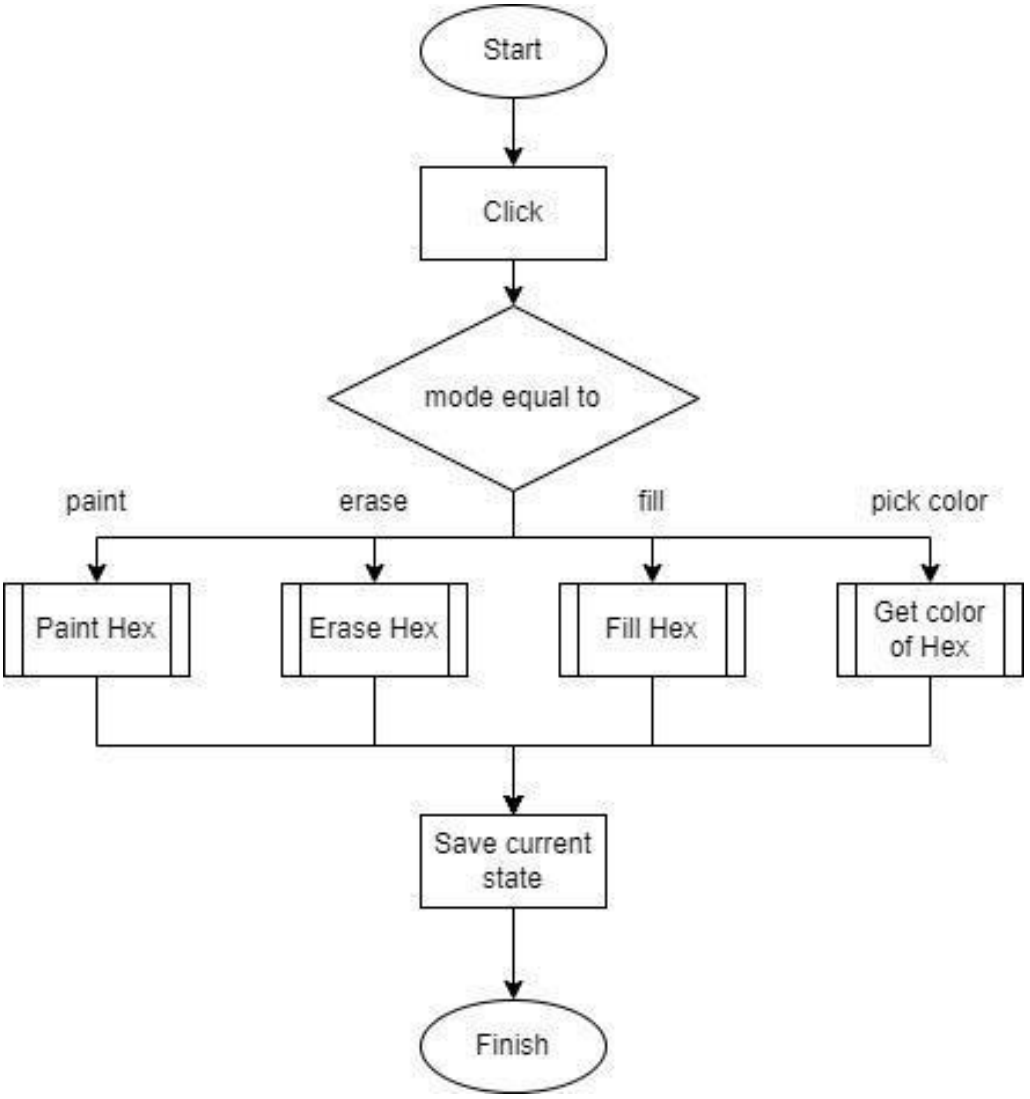


Рисунок 4.10 - Блок-схема обробки подій на гексагональному растрі

Даний алгоритм показує обробку певних подій на гексагональному растрі. Після виконання події, також завжди виконується функція збереження поточного стану, щоб при наступному запуску можна було продовжити роботу з останнього збереженого стану гексагонального растру. На рис. 4.11 наведено алгоритм отримання кольору певного гекса, звісно, якщо гекс має певний колір.

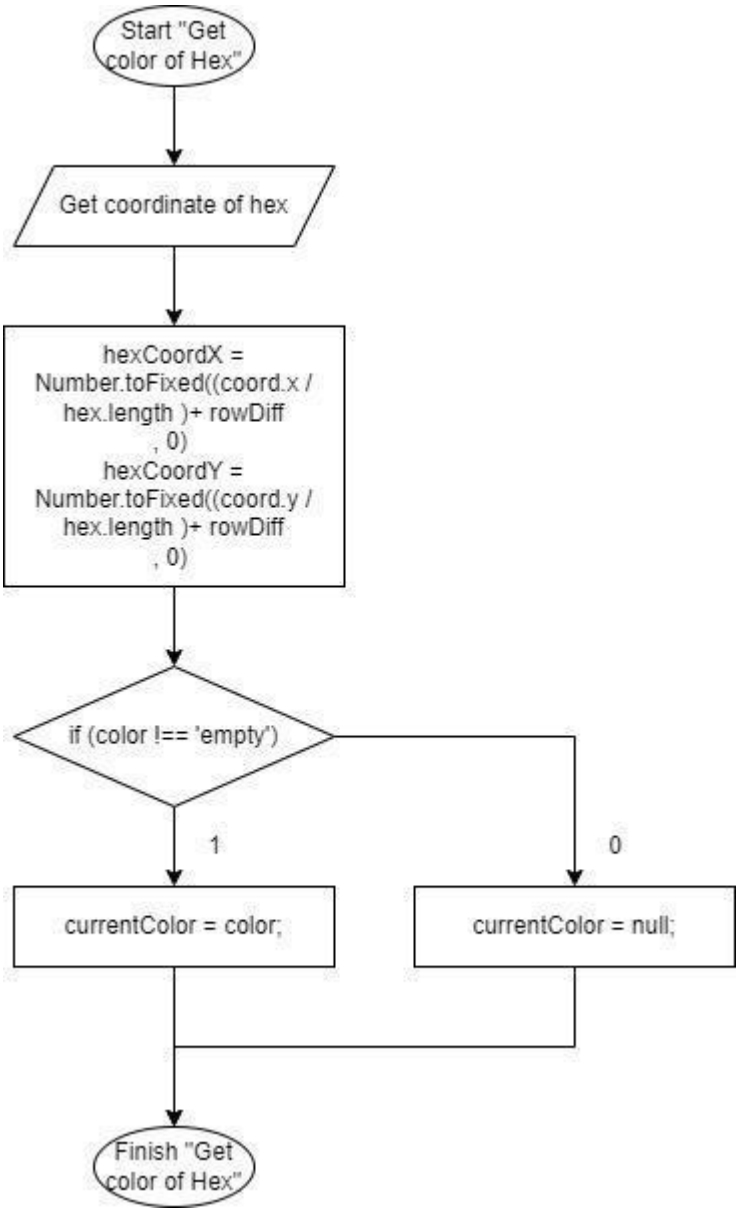


Рисунок 4.11 - блок-схема отримання кольору певного гекса

Даний алгоритм дозволяє отримати колір певного гекса за допомогою, алгоритму, який отримує дані позиції гекса з початковими даними натискання, визначає гекс, який містить у собі певні дані, в тому числі дані про колір гексу, які можуть містити колір, або просто бути у значення null.

Висновки

Як результат розробки, було отримано графічний редактор для формування

зображень на гексогональному растрі. У додатку було реалізовано базову функціональність редакторів, такі як: замальовування гекса; стирання кольору гекса; отримання кольору гекса; залиття гексів, які оточені гексами; зміна кольору; зміна форми замальовки гекса; можливість повертати останній дії у двосторонньому порядку; можливість копіювати гекса і вставляти; зміна гексагонального поля згідно своїх потреб.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1. Проведення комерційного аудиту науково-технічної розробки.

Результатом магістерської кваліфікаційної роботи “Методи та програмні засоби формування графічних зображень на гексогональному растрі” є програмний продукт у вигляді прогресивного додатку гексагонального редактора зображень у офлайн моді, який доступний на всіх платформах: Browser, Desktop, IOS та Android.

Для проведення комерційного аудиту залучено трьох незалежних експертів: доц. кафедри ПЗ Майданюк Володимир Павлович, доц. кафедри ПЗ Катєльніков Денис Іванович, доц. кафедри ПЗ Романюк Оксана Володимирівна.

Оцінювання комерційного потенціалу буде здійснене за критеріями, що наведені в таблиці 5.1 [7].

Таблиця 5.1 - Критерії оцінювання комерційного потенціалу розробки бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)

Кри-тер.	0	1	2	3	4
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу експертами розробки зведено в таблицю 5.2.

Таблиця 5.2 - Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1 – Майданюк	2 – Катєльніков	3 – Романюк
	Бали, виставлені експертами:		
1	4	3	4
Ринкові переваги (недоліки):			
2	3	2	3
3	4	4	4
4	4	3	4
5	3	4	3
Ринкові перспективи			
6	2	3	3
7	3	4	3
Практична здійсненність			
8	4	4	4
9	4	4	4
10	4	4	4
11	4	4	3
12	4	3	4
Сума балів	$СБ_1$=43	$СБ_2$=43	$СБ_3$=43
Середньоарифметична сума балів $\overline{СБ}$	43		

За даними таблиці 5.2 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 5.3.

Таблиця 5.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21–30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Рівень комерційного потенціалу розробки, становить 43 бали, що відповідає рівню «високий».

В якості аналога для розробки було обрано програмний додаток “Hexagonal Graph Paper”, який дозволяє малювати примітивні фігури на гексагональному растрі. Основними недоліками даного програмного продукту є:

– неможливість копіювати певні гекси, та вставляти у іншому місці,
– гекси мають тільки монолітну заливку,
– гекси не можуть змінювати свій розмір,
– немає заливки у межах певного гексу;

У розробленому додатку вирішуються дані недоліки, і користувач має більшу гнучкість у створенні зображень. Також програмний продукт випереджає аналог за такими параметрами як мультиплатформеність, швидкість обчислень, приємніший інтерфейс.

У таблиці 5.4 наведені основні технічні показники аналога і нового програмного продукту

Таблиця 5.4 - Основні технічні показники аналога і нового програмного продукту

Показники	Аналог	Нова розробка	Відношення параметрів нової розробки до параметрів аналога
Мультиплатформеність	2	5	2.5
Надійність	4	5	1.25
Функціональність	2	5	2.5
Супровід	5	4	0.8
Економія ресурсів і часу	2	4	2
Простота використання	5	4	0.8

Нову розробку можна використати у деяких сферах. Найбільш поширені сфери для використання даного продукту: освіта, UI/UX design, Frontend, Icon stocks, blockchain іконки, які можуть бути продані. Безпосередніми споживачами можуть бути: дизайнери, учні, розробники, stock traders.

Технічна готовність продукту становить 100%, додаток повністю функціональний, та готовий до використання. Існує промисловий зразок для Desktop версії, а також для Browser версії, який також можна запускати на смартфонах. Оскільки даний додаток є мультиплатформним, то кінцеві файли відрізняються для різних версій, але все це компілюється одним фреймоворком, що надає можливість використати додаток на різних обчислюваних машинах.

Документація не була розроблена з причин, тому, що додаток є інтуїтивно зрозумілим для користувача, та не вимагає специфічних навиків володінням комп’ютером.

Оскільки додаток уже має фінальну версію на різних девайсах, то було проведено показові роботи для деяких зацікавлених осіб, і було виявлено інтерес до даного додатку у контексті освіти. Даний додаток, може бути використаний учнями деяких шкіл, для кращого розуміння роботи гексагонального растру, а також початковими класами для кращого розвитку творчих навиків. На даний момент інвесторів не існує, оскільки даний продукт не був заявлений на широке коло осіб, а тільки для зацікавлених осіб у використанні даного додатку. Згодом

планується залучення інвестицій у випадку, якщо додаток буде мати більше 100 користувачів. В результаті відгуків цих користувачів буде можливо скласти кращу стратегію інвестицій, можливо навіть створити IT компанію, яка згодом буде продукувати нові додатки пов'язані з графікою, і можливо вийде на ринок IPO.

5.2 Розрахунок витрат на здійснення науково-технічної розробки

Проведемо розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи.

Обчислимо основну заробітну $З_о$, за формулою:

$$З_о = \frac{М}{Т_р} \cdot t[грн], \quad (5.1)$$

де М – місячний посадовий оклад конкретного розробника, грн;

$Т_р$ – число робочих днів в місяці –22;

t – число днів роботи розробника.

Обчислимо заробітну плату розробника з місячним окладом 13000 грн і кількістю робочих днів у місяці – 22. Число днів роботи розробника 66.

$$З_о = \frac{12600}{22} \cdot 66 = 37800,00(грн).$$

Результати розрахунків зведемо до таблиці 5.1.

Таблиця 5.5 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Розробник	12980,00	590,00	66	38940,00
Керівник роботи	14966,00	681,66	3	2045,00
Всього				40985,00

Обчислимо додаткову заробітну плату розробників $З_д$. Розрахуємо її як 10% від основної заробітної плати:

$$З_д = З_о \cdot 0,10 \text{ [грн]}. \quad (5.2)$$

$$З_д = 40985,00 \cdot 0,10 = 4098,50 \text{ (грн)}.$$

Згідно діючого законодавства нарахування на заробітну плату (Єдиний соціальний внесок) складають 22% від суми основної та додаткової заробітної плати.

$$Н_з = (З_{о\ р} + З_д) \cdot 0,22 \text{ [грн]}. \quad (5.3)$$

$$Н_з = (40985,00+4098,50) \cdot 0,22 = 9\ 918,37 \text{ (грн)}.$$

Обчислимо амортизацію обладнання, що використовувались для розробки. В спрощеному вигляді амортизаційні відрахування розраховується за формулою:

$$А = \frac{Ц \cdot Т}{12 \cdot Т_В} \text{ [грн]}, \quad (5.4)$$

де $Ц$ – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, грн;

Т – фактична тривалість використання, міс;

$Т_В$ – термін використання обладнання, приміщень тощо, роки.

Розрахуємо амортизаційні витрати на системний блок та монітор. Балансова вартість блоку становить 35000 грн, а термін його використання – 5 років, а фактична тривалість використання 3 місяці.

$$А = \frac{35000 \cdot 3}{12 \cdot 5} = 1750 \text{(грн)}.$$

Таблиця 5.6 - Величина амортизаційних відрахувань

№	Найменування	Балансова вартість, грн	Термін використання, р	Фактична тривалість використання, міс.	Величина амортизаційних відрахувань, грн
1	Системний блок	35000	5	3	1750
2	Монітор	20000	1	3	1000
Всього:					2750,00

Інформацію про витрати на матеріали, що використані при розробці програмного продукту винесемо до таблиці 5.6.

Таблиця 5.7 – Матеріали, що використовуються під час виконання роботи

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено, шт.	Вартість витраченого матеріалу, грн
Папір (пачка)	200,00	1	200,00
Ручка	15,00	2	30,00
Всього			230,00

Обчислимо витрати на силову електроенергію за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п \ [грн], \quad (5.5)$$

де В – вартість 1 кВт - години електроенергії, В = 2,89 грн/кВт – година;

П – встановлена потужність обладнання, кВт. П = 0,3 кВт;

Ф – фактична кількість годин роботи обладнання, годин. Ф = 528 годин;

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,3.

$$В_е = 2,89 \cdot 0,3 \cdot 528 \cdot 0,3 = 137,33 \text{ (грн)}.$$

Під час розробки програмного продукту використовувались лише безкоштовні програмні засоби.

Інші витрати ($В_{ін}$) охоплюють: витрати на Інтернет, управління організацією, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, тощо. Інші витрати приймаємо як 100% від суми основної заробітної плати розробників.

Величина інших витрат складе:

$$В_{ін} = З_о \cdot 100\% \ [грн]. \quad (5.6)$$

$$В_{ін} = 40985{,}00 \cdot 1 = 40985{,}00 \text{ (грн)}.$$

Обчислимо витрати на виконання даної роботи, що є суммою всіх попередніх витрат.

$$В = З_о + З_р + З_д + Н_з + А + В_е + В_{ін} \ [грн].$$

В=40985,00+4098,50+9 918,37+2750,00+230,00+137,33+40985,00=98104,20(грн).

Виконаємо прогнозування загальних витрат *(ЗВ)* на виконання та впровадження результатів виконаної науково-технічної роботи за формулою:

$$ЗВ = В_{заг} / \beta \ [грн], \quad (5.7)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи. Якщо розробка знаходиться на стадії впровадження, то $\beta \approx 0{,}9$.

$$ЗВ = 98104{,}20/0{,}9 = 109004{,}66 \text{ (грн)}.$$

Витрати на виконання наукової роботи та впровадження її результатів становитиме 109004,66 грн.

5.3 Розрахунок економічної ефективності та обґрунтування економічної доцільності комерціалізації науково-технічної розробки

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Виконання даної наукової роботи та впровадження її результатів складає приблизно до 1 року. Позитивні результати від впровадження розробки очікуються вже в перші місяці після впровадження.

Обчислимо збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

Збільшення чистого прибутку у потенційного інвестора протягом декількох років розраховується за формулою:

$$\Delta П_i = (\Delta Ц_o \bullet N + Ц_o \bullet \Delta N)_i \bullet \lambda \bullet \rho \bullet (1 - \frac{\vartheta}{100}) \qquad (5.8)$$

де $\Delta Ц_o$ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році;

N – основний кількісний показник, який визначає величину попиту на аналогічні розробки у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої науково-технічної розробки у році до впровадження результатів;

ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році;

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. λ =0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. ρ = 0,3;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у ϑ = 18%.

В результаті впровадження наукової розробки покращується якість певного продукту, що дозволяє підвищити ціну його реалізації на 200 грн. Кількість користувачів збільшиться протягом першого року на1100, другого – 1500, третього – 1600. Реалізація продукції до впровадження результатів наукової розробки складала 20 користувачів, а її ціна – 600 грн. Розрахуємо показник прибутку впродовж трьох років відносно базового.

Збільшення чистого прибутку $\Delta П_1$ протягом першого року складе:

$$\Delta П_1 = (200 \cdot 20 + (700 + 20) \cdot 1100) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 - 18/100) = 162\,398{,}60 \text{ (грн)}.$$

Обчислимо збільшення чистого прибутку $\Delta П_2$ протягом другого року:

$$\Delta П_1 = (200 \cdot 20 + (700 + 20) \cdot 1500) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 - 18/100) = 221\,436{,}24 \text{ (грн)}.$$

Збільшення чистого прибутку $\Delta П_3$ протягом третього року становитиме:

$$\Delta П_1 = (200 \cdot 20 + (700 + 20) \cdot 1600) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 - 18/100) = 236\,195{,}65 \text{ (грн)}.$$

Отже, розрахунки показують, що комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку.

Далі розраховуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{інв}} \bullet \text{ЗВ}$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. $k_{\text{інв}} = 2$.

$$\text{PV} = 2 \cdot 109004{,}66 = 218\,009{,}32 \text{ (грн)}.$$

Чистий приведений дохід для інвестора розраховуємо за формулою:

$$\text{ПП} = \sum_{1}^{\text{Т}} \frac{\Delta\text{П}_{\text{і}}}{(1+\tau)^{t}} \text{ [грн]}, \qquad (5.9)$$

де $\Delta П_i$ - збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої рроботи, грн;

т – період часу, протягом якого виявляються результати впровадженої роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки „0”.

$$\text{ПП} = \frac{162\,398{,}60}{(1+0{,}1)^{1}} + \frac{221\,436{,}24}{(1+0{,}1)^{2}} + \frac{236\,195{,}65}{(1+0{,}1)^{3}} = 508\,677{,}37 \text{(грн)}.$$

Розрахуємо абсолютну ефективність вкладених інвестицій $Е_{абс}$ за формулою:

$$\text{Е}_{\text{абс}} = (\text{ПП} - PV), \text{ [грн]}, \qquad (5.10)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.;

PV – теперішня вартість інвестицій PV = ЗВ, грн.

$E_{абс}$=508677,32– 218 009,32= 290 668,05 (грн).

Так як $E_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження принесе прибуток, але це також ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даного проекту.

На п’ятому етапі розрахуємо відносну (щорічну) ефективність вкладених у наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$\mathrm{E_в} = \sqrt[\text{Тж}]{1 + \frac{\mathrm{E_{абс}}}{PV}} - 1 \qquad (5.11)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

$$\mathrm{E_в} = \sqrt[3]{1 + \frac{290\ 668{,}05}{218\ 009{,}32}} - 1 = 0{,}32, \text{ або } 32\%.$$

Розраховану величину $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладати не вигідно. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування визначається за формулою:

$$\tau = d + f, \qquad (5.12)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = (0,09);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau = 0{,}09 + 0{,}05 = 0{,}14$$

Оскільки $E_в = 32\% > \tau$ мін $= 0{,}14 = 14\%$, то потенційний інвестор може бути зацікавлений у фінансуванні науково-технічної розробки.

На останньому етапі розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій.

Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ можна розрахувати за формулою:

$$\mathrm{T_{ок}} = \frac{1}{\mathrm{E_в}}, \qquad (5.13)$$

$$\mathrm{T_{ок}} = \frac{1}{0{,}32} = 3 \text{ (роки)},$$

Термін окупності становить 3 роки ($T_{ок}$ <3…5 років), що свідчить, що фінансування даної наукової розробки є доцільним.

Отже, розрахунок ефективності вкладених інвестицій та періоду їх окупності показав, що фінансування розробки є доцільним, адже інвестиції буде повернуто в термін до трьох років.

Висновки

В даному розділі було виконано оцінювання комерційного потенціалу розробки. Проведено технологічний аудит з залученням трьох незалежних експертів. Визначено, що рівень комерційного потенціалу розробки є високим.

Аналіз комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками випереджає аналогічні програмні продукти, що підтверджує її перспективність. Програмний продукт має кращі функціональні показники, а тому є конкурентоспроможним товаром на ринку. Існуючі переваги нової розробки дозволяють швидко поширити її на ринку.

Згідно із розрахунками всіх статей витрат на виконання науково-технічної роботи загальні витрати на розробку складають 109004,66 грн.

Розрахована абсолютна ефективність вкладених інвестицій в сумі 290 668,05 грн свідчить про отримання прибутку інвестором від комерціалізації програмного продукту.

Щорічна ефективність вкладених в наукову розробку інвестицій складає 32%, що вище за мінімальну бар'єрну ставку дисконтування, яка складає 14%. Це означає потенційну зацікавленість інвесторів у фінансуванні розробки.

Термін окупності вкладених у реалізацію проекту інвестицій становить 3 роки, що також свідчить про доцільність фінансування нової розробки.

ВИСНОВКИ

Метою роботи є підвищення реалістичності формування графічних зображень за рахунок використання гексагонального растру.

Проведено аналіз галузей використання гексагонального растру та його особливостей.

Модифіковано метод оцінювальної функції для колового інтерполювання, який відрізняється від відомого використанням нових функціональних залежностей, що дозволило підвищити реалістичність за рахунок використання гексагональної моделі пікселя.

Вперше запропоновано метод формування векторів на гексагональному растрі, особливість якого полягає у використання комбінованих переміщень, що дозволило до двох разів підвищити продуктивність лінійного інтерполювання.

Практичне значення отриманих результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботи теоретичних положень запропоновано алгоритми та розроблено програмні засоби для формування графічних зображень у комп’ютерних систем високо реалістичної візуалізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алгоритм Брезенхема для генерації окружності. [Електронний ресурс] – Режим доступу до ресурсу: http://www.um.co.ua/11/11-4/11-42703.html.
2. Гинзбург М. М., Путятин Є. П. Порівняльний аналіз прямокутної та гексагональної ґраток для дискретизації кривих. Бионика интеллекта: науч.-техн. жур- нал. 2012. –№ 2 (79). С. 13–18.
3. Квицинский Е. Алгоритм DDA-лінії [Електронний ресурс] / Евгений Квицинский. – 7. – Режим доступу до ресурсу: https://grafika.me/node/63.
4. Козубенко М. В. Архітектура frontend розробки [Електронний ресурс] / М. В. Козубенко, Н. П. Бабюк // Матеріали L науково-технічної конференції підрозділів ВНТУ, Вінниця, 10-12 березня 2021 р. – Електрон. текст. дані. – 2021. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2021/paper/view/12227.
5. Козубенко М. В., Мельник О.В., Романюк О. Н., Котлик С.В. Використання гексогонального растру в картографії. Інформаційні технології і автоматизація – 2022 / Матеріали XV міжнародної науково-практичної конференції. Одеса, 20-21 жовтня 2022 р. - Одеса, Видавництво ОНТУ, 2022 р. – 30-33 с.
6. Майстренко П. Растровий алгоритм Брезенхема побудови лінії [Електронний ресурс] / Поліна Майстренко. – 18. – Режим доступу до ресурсу: http://grafika.me/node/8.
7. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. 42 с.
8. Панфілова Ю. О., Романюк О Н., Мельник О.В. Використання гексагонального растру в комп’ютерних іграх.

Інформаційно-комп’ютерні технології: тези доп. XII Міжнародної науково-технічної конференції, м. Житомир, 01 - 03 квітня 2021 р. / Житомирська по-літехніка, 2021. С.205 https://doi.org/10.46505/IJBI.2021.3122

9. Пєтух А. М., Обідник Д. Т., Романюк О.Н. Інтерполяція в задачах контурного формоутворення: монографія. Вінниця: ВНТУ, 2007. 103 с.
10. Романюк О. Н. Моделювання гексагонального растра на квадратному растрі [Текст] / О. Н. Романюк, О. В. Мельник, О. В. Стукач // Матеріали сьомої міжнародної науково-технічної конференції "Моделювання і комп’ютерна графіка", м. Покровськ, м. Київ, 18-24 вересня 2017 р. – С. 286.
11. Романюк О. Н. Панфілова Ю. О. Деякі застосування гексагональної моделі піксела. Інформаційно-комп’ютерні технології – 2020 : тези доп. XI Міжнародної науково-технічної конференції, м. Житомир, 09 – 11 квітня 2020 р. / Житомирська політехніка, 2020. – С. 116–117. https://doi.org/10.1055/a-1078-2974
12. Романюк О. Н. Реалізація кругової інтерполяції при використанні гексагонального растру [Текст] / О. Н. Романюк, О. В. Мельник,О. В. Романюк // Наукові праці Донецького національного технічного університету. Серія «Інформатика, кібернетика та обчислювальна техніка». - Донецьк, 2017. - № 1. – С. 53 – 58.
13. Романюк О. Н. Формування відрізків прямих на гексагональному растрі [Текст] / О. Н. Романюк, О. В. Мельник, О. В. Романюк // Наукові праці Донецького національного технічного університету. Серія "Інформатика, кібернетика та обчислювальна техніка". – 2016. - №2(23). – С. 69-72.
14. Романюк О. Н., Мельник О. В., Коваль Л. Г. Використання гексагональних комірок у видавничій справі. Інформація, комунікація та управління знаннями в глобалізованому світі Матеріали П’ятої міжнародної наукової конференції «Інформація, комунікація та

управління знаннями в глобалізованому світі», Київ, 22 травня, 2022. С.45-47.

15. Романюк О. Н., Мельник О.В., Чехместрук Р. Ю., Романюк С. О. Основні співідношення гексагонального растру. Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі: матеріали VII Міжн. наук.-практ. конф. м. Київ, 21 квітня 2022. С. 59-61.
16. Романюк О. Н.,. Романюк О.В., Мельник О.В., Козубенко М.В., Вінтонюк В.В..Програмна реалізація графічного редактора на гесагональному растрі.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.
17. Романюк О.Н., Мельник О.В., Марущак А.В., Шмалюх В.А. Комп’ютерна програма для імітації гексагонального растру. Інформаційні технології в освіті, техніці та промисловості: тези Республ. наук.- практ. конф., м. Івано-Франківськ, 8 жовтня, 2020. С.70-71
18. GitHub. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/GitHub.
19. Hexagonal Grids [Електронний ресурс] // Red Blob Games. – 13. – Режим доступу до ресурсу: https://www.redblobgames.com/grids/hexagons/.
20. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/JavaScript.
21. Melnik O.,Romanyuk O.,Romanyuk O., Savratsky V. Applying of hexagonal raster in image formation scientific foundations of modern engineering. Monography/International Science Group. Boston: Primedia eLaunch, 2020. 166=175 p
22. Romanyuk Olexander, Pavlov Sergii, Melnyk Olexander, Romanyuk Sergii, Smolarz Andrzej, Bazarova Madina, Method of anti-aliasing with the use of the new pixel model, Proc. SPIE 9816, Optical Fibers and Their Applications 2015, 981617 (17 December 2015); doi: 10.1117/12.2229013. https://doi.org/10.1117/12.2229013

23. What is Electron [Електронний ресурс] – Режим доступу до ресурсу: https://www.electronjs.org/docs/latest/.

ДОДАТКИ

101

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., професор Романюк О.Н.

"16 " вересня 2022 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Методи та програмні засоби формування графічних зображень на гексогональному растрі»

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

д.т.н., проф. О.Н.Романюк

"15" вересня 2022 р.

Виконав:

студент гр. 1ПІ-21м М. В. Козубенко

"15" вересня 2022 р.

Вінниця – 2022 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Методи та програмні засоби формування графічних зображень на гексогональному растрі»

Галузь застосування - системи комп’ютерної графіки.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР і наказ ректора по ВНТУ № 205-А від «15» вересня 2022 р. про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення реалістичності формування графічних зображень за рахунок використання гексагонального растру.

Призначення роботи – розробка методів і засобів формування векторів і кіл на гексагональному растрі.

4.4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Цисарж В. В. Математические методы компьютерной графики / Цисарж., Р. И. Марусин. – К. : Факт, 2004. – 464 с.

2. Романюк О. Н. Комп’ютерна графіка. Навчальний посібник / О. Н. Романюк — Вінниця: ВДТУ, 2001. — 129 с.

3. Романюк О. Н., Мельник О.В., Чехместрук Р. Ю., Романюк С. О. Основні співідношення гексагонального растру. Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі: матеріали VII Міжнародної науково-практичної конференції. / М-во освіти і науки України; Київ. нац. ун-т культури і мистецтв.– Київ : Видавничий центр КНУКіМ, 2022. С. 59-61.

4. Романюк о. Н. Моделювання гексагонального растра на квадратному растрі [Текст] / О. Н. Романюк, О. В. Мельник, О. В. Стукач // Матеріали сьомої міжнародної науково-технічної конференції "Моделювання і комп’ютерна графіка", м. Покровськ, м. Київ, 18-24 вересня 2017 р. – С. 286.

5

5. Технічні вимоги

Тип растру - гексагональний; тип графічних примітивів – відрізок прямої, коло; метод інтерполяції - метод оцінювальної функції; кольоровий режим – TrueColor.

66. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

77. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до МКР;
- технічне завдання;
- лістинги програми.

88. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

99. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз використання та формування зображень на гексагональному растрі	18.09.2022-27.09.2022
2	Методи та програмні засоби формування відрізків прямих на гексогональному растрі	28.09.2022-15.10.2022
3	Метод та програмні засоби формування кола на гексогональному растрі	16.10.2022-7.11.2022
4	Програмна реалізація методів і програмних засобів формування зображень на гексогональному растрі	8.11.2022-27.11.2022
5	Економічна частина	28.11.2022-10.12.2022

1010. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

ДОДАТОК Б

Протокол перевірки

роботи

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Методи та програмні засоби формування графічних зображень на гексогональному растрі

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 1ПІ – 21м

Науковий керівник: д.т.н., професор каф. ПЗ Романюк О.Н.

Unicheck	
Оригінальність	**96.1%**
Схожість	3.9%

Аналіз звіту подібності

▪ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Козубенко М. В.

Керівник роботи Романюк О. Н.

ДОДАТОК В

Файл painter.js

```
// RaphaelJS documentation - http://raphaeljs.com/reference.html
// JQuery Minicolors color picker documentation - http://labs.abeautifulsite.net/jquery-miniColors/
// Tag manager - http://welldonethings.com/tags/manager

const artwork = {
  version: { major: 2, minor: 0 },
  effectiveRect: { left: 288, top: 166, width: 1087, height: 625 },
  canvasSize: { width: 2000, height: 2000 },
  backgroundColor: "#ffffff",
  layers: [
   {
     grid: "hex",
     cellSize: 24,
     gridThickness: 1,
     gridColor: "#d0d0d0",
     rows: [
      {
        row: 8,
        cells: [
          [40, "flat", "#4096EE"],
          [41, "flat", "#4096EE"],
          [42, "flat", "#4096EE"],
          [43, "flat", "#4096EE"],
          [44, "flat", "#4096EE"],
          [45, "flat", "#4096EE"],
          [46, "flat", "#4096EE"],
          [48, "flat", "#4096EE"],
          [50, "flat", "#4096EE"],
          [51, "flat", "#4096EE"],
          [52, "flat", "#4096EE"],
          [55, "flat", "#4096EE"],
          [57, "flat", "#4096EE"],
        ],
      },
      {
        row: 9,
        cells: [
          [40, "flat", "#4096EE"],
          [41, "flat", "#4096EE"],
          [42, "flat", "#4096EE"],
          [52, "flat", "#4096EE"],
          [53, "flat", "#4096EE"],
          [56, "flat", "#4096EE"],
          [57, "flat", "#4096EE"],
          [58, "flat", "#4096EE"],
```

```
  ],
},
{
  row: 10,
  cells: [
    [40, "flat", "#4096EE"],
    [41, "flat", "#4096EE"],
    [43, "flat", "#4096EE"],
    [47, "flat", "#4096EE"],
    [50, "flat", "#4096EE"],
    [51, "flat", "#4096EE"],
    [52, "flat", "#4096EE"],
    [53, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
    [56, "flat", "#4096EE"],
  ],
},
{
  row: 11,
  cells: [
    [40, "flat", "#4096EE"],
    [41, "flat", "#4096EE"],
    [47, "flat", "#4096EE"],
    [50, "flat", "#4096EE"],
    [51, "flat", "#4096EE"],
    [54, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
  ],
},
{
  row: 12,
  cells: [
    [40, "flat", "#4096EE"],
    [42, "flat", "#4096EE"],
    [44, "flat", "#4096EE"],
    [46, "flat", "#4096EE"],
    [49, "flat", "#4096EE"],
    [57, "flat", "#4096EE"],
  ],
},
{
  row: 13,
  cells: [
    [28, "flat", "#4096EE"],
    [30, "flat", "#4096EE"],
    [32, "flat", "#4096EE"],
    [34, "flat", "#4096EE"],
    [36, "flat", "#4096EE"],
    [37, "flat", "#4096EE"],
    [39, "flat", "#4096EE"],
```

```
    [40, "flat", "#4096EE"],
    [41, "flat", "#4096EE"],
    [42, "flat", "#4096EE"],
    [47, "flat", "#4096EE"],
    [58, "flat", "#4096EE"],
    [59, "flat", "#4096EE"],
  ],
},
{
  row: 14,
  cells: [
    [25, "flat", "#4096EE"],
    [39, "flat", "#4096EE"],
    [40, "flat", "#4096EE"],
    [43, "flat", "#4096EE"],
    [59, "flat", "#4096EE"],
  ],
},
{
  row: 15,
  cells: [
    [23, "flat", "#4096EE"],
    [36, "flat", "#4096EE"],
    [40, "flat", "#4096EE"],
    [41, "flat", "#4096EE"],
    [42, "flat", "#4096EE"],
    [44, "flat", "#4096EE"],
    [57, "flat", "#4096EE"],
    [58, "flat", "#4096EE"],
  ],
},
{
  row: 16,
  cells: [
    [21, "flat", "#4096EE"],
    [34, "flat", "#4096EE"],
    [41, "flat", "#4096EE"],
    [42, "flat", "#4096EE"],
    [43, "flat", "#4096EE"],
    [44, "flat", "#4096EE"],
    [46, "flat", "#4096EE"],
    [49, "flat", "#4096EE"],
    [56, "flat", "#4096EE"],
    [58, "flat", "#4096EE"],
  ],
},
{
  row: 17,
  cells: [
    [33, "flat", "#4096EE"],
```

```
    [41, "flat", "#4096EE"],
    [44, "flat", "#4096EE"],
    [45, "flat", "#4096EE"],
    [49, "flat", "#4096EE"],
    [51, "flat", "#4096EE"],
    [53, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
  ],
},
{
  row: 18,
  cells: [
    [19, "flat", "#4096EE"],
    [36, "flat", "#4096EE"],
    [41, "flat", "#4096EE"],
    [42, "flat", "#4096EE"],
    [43, "flat", "#4096EE"],
    [46, "flat", "#4096EE"],
    [47, "flat", "#4096EE"],
    [49, "flat", "#4096EE"],
    [56, "flat", "#4096EE"],
    [75, "flat", "#4096EE"],
  ],
},
{
  row: 19,
  cells: [
    [32, "flat", "#4096EE"],
    [33, "flat", "#4096EE"],
    [41, "flat", "#4096EE"],
    [42, "flat", "#4096EE"],
    [43, "flat", "#4096EE"],
    [44, "flat", "#4096EE"],
    [45, "flat", "#4096EE"],
    [46, "flat", "#4096EE"],
    [47, "flat", "#4096EE"],
    [48, "flat", "#4096EE"],
    [49, "flat", "#4096EE"],
    [53, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
    [56, "flat", "#4096EE"],
    [57, "flat", "#4096EE"],
  ],
},
{
  row: 20,
  cells: [
    [18, "flat", "#4096EE"],
    [40, "flat", "#4096EE"],
    [41, "flat", "#4096EE"],
```

```
    [42, "flat", "#4096EE"],
    [44, "flat", "#4096EE"],
    [45, "flat", "#4096EE"],
    [46, "flat", "#4096EE"],
    [48, "flat", "#4096EE"],
    [49, "flat", "#4096EE"],
    [50, "flat", "#4096EE"],
    [51, "flat", "#4096EE"],
    [57, "flat", "#4096EE"],
    [58, "flat", "#4096EE"],
  ],
},
{
  row: 21,
  cells: [
    [17, "flat", "#4096EE"],
    [31, "flat", "#4096EE"],
    [32, "flat", "#4096EE"],
    [39, "flat", "#4096EE"],
    [40, "flat", "#4096EE"],
    [43, "flat", "#4096EE"],
    [44, "flat", "#4096EE"],
    [46, "flat", "#4096EE"],
    [47, "flat", "#4096EE"],
    [49, "flat", "#4096EE"],
    [50, "flat", "#4096EE"],
    [51, "flat", "#4096EE"],
    [52, "flat", "#4096EE"],
    [53, "flat", "#4096EE"],
    [54, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
    [56, "flat", "#4096EE"],
    [57, "flat", "#4096EE"],
    [60, "flat", "#4096EE"],
    [62, "flat", "#4096EE"],
    [63, "flat", "#4096EE"],
    [65, "flat", "#4096EE"],
    [66, "flat", "#4096EE"],
    [68, "flat", "#4096EE"],
    [70, "flat", "#4096EE"],
    [71, "flat", "#4096EE"],
  ],
},
{
  row: 22,
  cells: [
    [30, "flat", "#4096EE"],
    [39, "flat", "#4096EE"],
    [40, "flat", "#4096EE"],
    [41, "flat", "#4096EE"],
```

```
    [42, "flat", "#4096EE"],
    [43, "flat", "#4096EE"],
    [48, "flat", "#4096EE"],
    [52, "flat", "#4096EE"],
    [53, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
    [56, "flat", "#4096EE"],
    [58, "flat", "#4096EE"],
    [59, "flat", "#4096EE"],
    [64, "flat", "#4096EE"],
    [65, "flat", "#4096EE"],
    [72, "flat", "#4096EE"],
  ],
},
{
  row: 23,
  cells: [
    [30, "flat", "#4096EE"],
    [37, "flat", "#4096EE"],
    [39, "flat", "#4096EE"],
    [40, "flat", "#4096EE"],
    [41, "flat", "#4096EE"],
    [42, "flat", "#4096EE"],
    [44, "flat", "#4096EE"],
    [47, "flat", "#4096EE"],
    [49, "flat", "#4096EE"],
    [50, "flat", "#4096EE"],
    [54, "flat", "#4096EE"],
    [59, "flat", "#4096EE"],
    [61, "flat", "#4096EE"],
    [63, "flat", "#4096EE"],
    [65, "flat", "#4096EE"],
  ],
},
{
  row: 24,
  cells: [
    [16, "flat", "#4096EE"],
    [29, "flat", "#4096EE"],
    [36, "flat", "#4096EE"],
    [38, "flat", "#4096EE"],
    [39, "flat", "#4096EE"],
    [40, "flat", "#4096EE"],
    [41, "flat", "#4096EE"],
    [43, "flat", "#4096EE"],
    [44, "flat", "#4096EE"],
    [45, "flat", "#4096EE"],
    [46, "flat", "#4096EE"],
    [47, "flat", "#4096EE"],
    [50, "flat", "#4096EE"],
```

```
    [51, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
    [56, "flat", "#4096EE"],
    [57, "flat", "#4096EE"],
    [59, "flat", "#4096EE"],
    [60, "flat", "#4096EE"],
    [63, "flat", "#4096EE"],
    [64, "flat", "#4096EE"],
    [67, "flat", "#4096EE"],
  ],
},
{
  row: 25,
  cells: [
    [30, "flat", "#4096EE"],
    [33, "flat", "#4096EE"],
    [34, "flat", "#4096EE"],
    [39, "flat", "#4096EE"],
    [40, "flat", "#4096EE"],
    [42, "flat", "#4096EE"],
    [48, "flat", "#4096EE"],
    [50, "flat", "#4096EE"],
    [52, "flat", "#4096EE"],
    [53, "flat", "#4096EE"],
    [58, "flat", "#4096EE"],
    [60, "flat", "#4096EE"],
    [62, "flat", "#4096EE"],
    [68, "flat", "#4096EE"],
    [69, "flat", "#4096EE"],
  ],
},
{
  row: 26,
  cells: [
    [16, "flat", "#4096EE"],
    [28, "flat", "#4096EE"],
    [37, "flat", "#4096EE"],
    [48, "flat", "#4096EE"],
    [50, "flat", "#4096EE"],
    [54, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
    [65, "flat", "#4096EE"],
    [68, "flat", "#4096EE"],
    [70, "flat", "#4096EE"],
    [71, "flat", "#4096EE"],
    [72, "flat", "#4096EE"],
    [73, "flat", "#4096EE"],
  ],
},
{
```

```
  row: 27,
  cells: [
    [30, "flat", "#4096EE"],
    [34, "flat", "#4096EE"],
    [39, "flat", "#4096EE"],
    [42, "flat", "#4096EE"],
    [43, "flat", "#4096EE"],
    [44, "flat", "#4096EE"],
    [46, "flat", "#4096EE"],
    [49, "flat", "#4096EE"],
    [50, "flat", "#4096EE"],
    [54, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
    [60, "flat", "#4096EE"],
    [70, "flat", "#4096EE"],
    [71, "flat", "#4096EE"],
  ],
},
{
  row: 28,
  cells: [
    [16, "flat", "#4096EE"],
    [28, "flat", "#4096EE"],
    [30, "flat", "#4096EE"],
    [34, "flat", "#4096EE"],
    [37, "flat", "#4096EE"],
    [49, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
  ],
},
{
  row: 29,
  cells: [
    [17, "flat", "#4096EE"],
    [23, "flat", "#4096EE"],
    [24, "flat", "#4096EE"],
    [25, "flat", "#4096EE"],
    [28, "flat", "#4096EE"],
    [29, "flat", "#4096EE"],
    [30, "flat", "#4096EE"],
    [31, "flat", "#4096EE"],
    [38, "flat", "#4096EE"],
    [40, "flat", "#4096EE"],
    [43, "flat", "#4096EE"],
    [45, "flat", "#4096EE"],
    [47, "flat", "#4096EE"],
    [49, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
    [72, "flat", "#4096EE"],
  ],
```

```
},
{
  row: 30,
  cells: [
    [17, "flat", "#4096EE"],
    [21, "flat", "#4096EE"],
    [22, "flat", "#4096EE"],
    [24, "flat", "#4096EE"],
    [31, "flat", "#4096EE"],
    [33, "flat", "#4096EE"],
    [39, "flat", "#4096EE"],
    [47, "flat", "#4096EE"],
    [48, "flat", "#4096EE"],
    [49, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
    [72, "flat", "#4096EE"],
  ],
},
{
  row: 31,
  cells: [
    [18, "flat", "#4096EE"],
    [19, "flat", "#4096EE"],
    [20, "flat", "#4096EE"],
    [22, "flat", "#4096EE"],
    [34, "flat", "#4096EE"],
    [35, "flat", "#4096EE"],
    [37, "flat", "#4096EE"],
    [46, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
    [73, "flat", "#4096EE"],
  ],
},
{
  row: 32,
  cells: [
    [39, "flat", "#4096EE"],
    [40, "flat", "#4096EE"],
    [42, "flat", "#4096EE"],
    [44, "flat", "#4096EE"],
    [55, "flat", "#4096EE"],
    [73, "flat", "#4096EE"],
  ],
},
{
  row: 33,
  cells: [
    [39, "flat", "#4096EE"],
    [44, "flat", "#4096EE"],
    [47, "flat", "#4096EE"],
```

```
          [49, "flat", "#4096EE"],
          [51, "flat", "#4096EE"],
          [53, "flat", "#4096EE"],
          [54, "flat", "#4096EE"],
          [73, "flat", "#4096EE"],
        ],
      },
      {
        row: 34,
        cells: [
          [40, "flat", "#4096EE"],
          [41, "flat", "#4096EE"],
          [73, "flat", "#4096EE"],
        ],
      },
      {
        row: 35,
        cells: [
          [43, "flat", "#4096EE"],
          [71, "flat", "#4096EE"],
          [72, "flat", "#4096EE"],
        ],
      },
      {
        row: 36,
        cells: [
          [46, "flat", "#4096EE"],
          [50, "flat", "#4096EE"],
          [54, "flat", "#4096EE"],
          [59, "flat", "#4096EE"],
          [63, "flat", "#4096EE"],
          [70, "flat", "#4096EE"],
        ],
      },
      { row: 37, cells: [[66, "flat", "#4096EE"]] },
    ],
  },
],
recentColors: [
  "#4096EE",
  "#FFFFFF",
  "#000000",
  "#EEEEEE",
  "#FFFF88",
  "#CDEB8B",
  "#6BBA70",
  "#006E2E",
  "#C3D9FF",
  "#356AA0",
  "#FF0096",
```

```
    "#B02B2C",
    "#FF7400",
    "#EF9090",
    "#0099FF",
    "#9933FF",
    "#2E2EFF",
    "#8A725D",
    "#FF3838",
    "#4BC8D1",
    "#CBD114",
    "#858585",
  ],
  transparentBackground: false,
  gridVisible: false,
  additionalPixelImage: false,
};
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-fillscreen-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;

  setBackgroundColor(backgroundColor);

  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 by 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="${tool.title}"><img src="${tool.iconUrl}" /></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) {
  updateCellCoordiantesPanel(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) {
      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>
                            </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",
  letterCase: "uppercase",
});

$("#workspace-background-color").minicolors({
  theme: "bootstrap",
```

```
  letterCase: "uppercase",
});

$("#btn-set-background-color").click(function () {
  undoStep = new UndoStep();
  undoStep.setBackgroundChange(backgroundColor, selectedColor);
  undoStack.push(undoStep);
  redoStack = [];
  updateUndoRedoButtons();

  setBackgroundColor(selectedColor);

  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();
  }
});
```

ДОДАТОК Д Лістинг коду 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%3
Dhttps://uc.appengine.google.com/_ah/logout%253Fcontinue%253Dhttps://google.com/url%25253Fs
a%25253DD%252526q%25253Dhttps://grid-paint.com/%252526ust%25253D1665603412179444%2
52526usg%25253DAOvVaw2GRh0_ZAR7S61BWL44gSy2%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="/">
                                <img src="/img/logo2-shadow.svg" width="40" height="40"
                                        style="float: left; margin-top:-10px; margin-bottom:-20px;
margin-left:10px; margin-right:10px;">
                                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"><img
                                        class="control-icons" src="./img/maximize.png"
alt="edit"></i></button>
                        <div class="btn-group">
                                <button id="btn-undo" class="btn btn-sm" type="button"
title="Undo (Ctrl-Z)"><img
                                        class="control-icons" src="./img/undo.png"
alt="undo"></button>
                                <button id="btn-redo" class="btn btn-sm" type="button"
title="Redo"><img class="control-icons"
                                        src="./img/redo.png" alt="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">
                                <img class="icon-manage" src="/img/buttons/pen.png" />
                        </button><button id="btn-pick-color" class="btn btn-sm btn-default
painter-btn-tool" type="button"
                                title="Pick color">
                                <img class="icon-manage" src="/img/buttons/pick-color.png" />
                        </button><button id="btn-erase" class="btn btn-sm btn-default
painter-btn-tool" type="button"
                                title="Erase">
                                <img class="icon-manage" src="/img/buttons/eraser.png" />
                        </button><button id="btn-flood-fill" class="btn btn-sm btn-default
painter-btn-tool" type="button"
                                title="Flood fill" style="display:none;">
                                <img class="icon-manage" src="/img/buttons/flood-fill.png" />
                        </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"><img class="arrow-down"
src="./img/down.png" alt="down">
                                        </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"><img
                                                                class="arrowsdirection"
src="./img/up-arrow.png" alt="arrow-up"></button>
                                        </div>
                                        <div class="painter-toolbar-container text-center">
                                                <button id="btn-shift-left" class="btn btn-sm"
type="button" title="Shift left"><img
                                                                class="arrowsdirection"
src="./img/arrow.png" alt="arrow-left"></button>
                                                <span>    </span>
                                                <button id="btn-shift-right" class="btn btn-sm"
type="button" title="Shift right"><img
                                                                class="arrowsdirection"
src="./img/right-arrow.png" alt="arrow-rigth"></button>
                                        </div>
                                        <div class="painter-toolbar-container text-center">
                                                <button id="btn-shift-down" class="btn btn-sm"
type="button" title="Shift down"><img
                                                                class="arrowsdirection"
src="./img/down-arrow.png" alt="arrow-down"></button>
                                        </div>
                                </div>
                        </div>
                        <div class="painter-toolbar">
                                <!-- Copy/Paste toolbar -->
                                <div class="painter-toolbar-header" id="copy-paste-toolbar-header"
style="cursor:pointer;">
                                        Copy / Paste
```

```
                                        <div class="pull-right arrows-position"><img class="arrow-down"
src="./img/down.png" alt="down">
                                        </div>
                                </div>
                                <div id="copy-paste-toolbar-content" style="display:none">
                                        <div class="painter-toolbar-container text-center">
                                                <button type="button" id="btn-copy-mode" class="btn
btn-sm btn-success"> Mark area </button>
                                                <button type="button" id="btn-paste-mode" class="btn
btn-sm btn-primary"> Paste </button>
                                        </div>
                                </div>
                        </div>
                        <div id="coordinates"></div>
                        <div id="test"></div>
                </div>

                <div style="margin-left:0 !important;">
                        <div id="canvas-wrapper" class="painter-canvas-wrapper">
                                <div id="canvas"></div>
                        </div>

                </div>
        </div>

        <div class="group-image-online" style="display: none;">
                <div id="socketio-online" style="display: none;"><i class="icon icon-ok"></i>
Online</div>
                <div id="socketio-offline"><i class="icon icon-remove"></i> Offline</div>
                <div id="call-collaborators" style="display: none;">Call your friends to draw
together</div>
        </div>
        <div class="group-image-users-online" style="display: none;">
        </div>
        <div class="group-image-chat-wrapper" id="group-image-chat-wrapper" style="display:
none;">
                <div class="group-image-chat-show-button" style="display: none;"><i
                                class="glyphicon glyphicon-chevron-down"></i></div>
                <div class="group-image-chat-messages" style="display: none;">
                        <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 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"
                                        name="workspace-width">
                                </div>
                            </div>
                            <div class="col-xs-3">
                                <div>Height (px)</div>
                                <div>
                                    <input type="number"
class="form-control" id="workspace-height"
                                        name="workspace-height">
                                </div>
                            </div>
                            <div class="col-xs-6">
                                <div>Background color</div>
                                <div>
                                    <input type="text"
class="form-control minicolors-input" value="#ffffff"

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"

name="workspace-cell-size">
                                </div>
                            </div>
                            <div class=col-xs-3>
                                <div>Grid thickness (px)</div>
                                <div>
                                    <input type="number"
class="form-control" id="workspace-grid-thickness"

name="workspace-grid-thickness">
                                </div>
                            </div>
                            <div class=col-xs-6>
                                <div>Grid color</div>
                                <div>
```

```
                                                    <input type="text"
class="form-control minicolors-input" value="#d0d0d0"
                                                        id="workspace-grid-color"
name="workspace-grid-color">
                                                    </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>

                        <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>
                                                    <i class="icon icon-caret-right"></i>
Artwork description <span

class="save-dialog-expandable-content-preview"></span>
                                                </label>
                                                <textarea name="modal_artwork_description"
id="modal_artwork_description"
                                                    style="height:80px; display:none;"
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>
                                                        <i class="icon icon-caret-right"></i> Tags
<span
class="save-dialog-expandable-content-preview"></span>
                                                </label>
                                                <div id="modal_artwork_tags_field"
style="display:none;">
                                                        <input type="text" value=""
name="modal_artwork_tags" id="modal_artwork_tags"
                                                                autocomplete="off" />
                                                </div>
                                                <div id="modal-tags-hint" class="form-field-hint"
style="display:none;">Tag length should be
                                                        longer 1 and shorter 64 characters.</div>
                                        </div>
                                        <div id="modal_square_grid_special_properties"
class="form-group">
                                                <label class="save-dialog-expandable-label"

id="modal_square_grid_special_properties_label"><i class="icon icon-caret-right"></i>
                                                        Image rendering properties</label>
                                                <div class="save-dialog-special-properties-frame"
style="display:none;">
                                                        <div class="checkbox">
                                                                <label>
                                                                        <input
name="modal_artwork_grid_visible" id="modal_artwork_grid_visible"
                                                                                type="checkbox" />
Visible grid
                                                                </label>
                                                        </div>

                                                        <div class="checkbox">
                                                                <label>
                                                                        <input
name="modal_artwork_pixel_art" id="modal_artwork_pixel_art"
                                                                                type="checkbox" />
Pixel-art image
                                                                </label>
                                                        </div>

                                                        <div class="checkbox">
                                                                <label>
```

```
                                                                    <input
name="modal_transparent_background" id="modal_transparent_background"
                                                                        type="checkbox" />
Transparent background
                                                                </label>
                                                            </div>
                                                        </div>
                                                    </div>
                                                    <input type="hidden" name="modal_success_action"
id="modal_success_action" />
                                                </form>
                                            </div>
                                            <div class="modal-footer">
                                                <button type="button" class="btn"
data-dismiss="modal">Close</button>
                                                <button id="btn-properties-save" type="button" class="btn
btn-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>
                                <div class="modal-body">
                                        <div id="message-text"></div>
                                </div>
                                <div class="modal-footer">
                                        <button type="button" class="btn"
data-dismiss="modal">Close</button>
                                </div>
                        </div>
                </div>
        </div>

        <form name="f" style="display:none" action="/save-image" method="post">
                <input type="hidden" name="artwork_id" value="5197956457168896" />
                <input type="hidden" name="artwork_name" id="artwork_name" value="qwerty" />
                <textarea name="artwork_description" id="artwork_description"></textarea>
                <input type="hidden" name="artwork_tags" id="artwork_tags" value="" />
                <input type="hidden" name="artwork_json" value="" />
                <input type="hidden" name="artwork_grid_visible" id="artwork_grid_visible" value="" />
                <input type="hidden" name="artwork_pixel_art" id="artwork_pixel_art" value="" />
        </form>

        <script src="/js/grids.js"></script>
        <script src="/js/grid-square.js"></script>
        <script src="/js/grid-triangle.js"></script>
        <script src="/js/grid-iso-triangle.js"></script>
        <script src="/js/grid-hex.js"></script>
        <script src="/js/grid-iso-hex.js"></script>
        <script src="/js/grid-triangles4.js"></script>
        <script src="/js/grid-diamond.js"></script>
```

<script>
var artwork = { "version": { "major": 2, "minor": 0 }, "effectiveRect": { "left": 288, "top": 166, "width": 1087, "height": 625 }, "canvasSize": { "width": 2000, "height": 2000 }, "backgroundColor": "#ffffff", "layers": [{ "grid": "hex", "cellSize": 24, "gridThickness": 1, "gridColor": "#d0d0d0", "rows": [{ "row": 8, "cells": [[40, "flat", "#4096EE"], [41, "flat", "#4096EE"], [42, "flat", "#4096EE"], [43, "flat", "#4096EE"], [44, "flat", "#4096EE"], [45, "flat", "#4096EE"], [46, "flat", "#4096EE"], [48, "flat", "#4096EE"], [50, "flat", "#4096EE"], [51, "flat", "#4096EE"], [52, "flat", "#4096EE"], [55, "flat", "#4096EE"], [57, "flat", "#4096EE"]] }, { "row": 9, "cells": [[40, "flat", "#4096EE"], [41, "flat", "#4096EE"], [42, "flat", "#4096EE"], [52, "flat", "#4096EE"], [53, "flat", "#4096EE"], [56, "flat", "#4096EE"], [57, "flat", "#4096EE"], [58, "flat", "#4096EE"]] }, { "row": 10, "cells": [[40, "flat", "#4096EE"], [41, "flat", "#4096EE"], [43, "flat", "#4096EE"], [47, "flat", "#4096EE"], [50, "flat", "#4096EE"], [51, "flat", "#4096EE"], [52, "flat", "#4096EE"], [53, "flat", "#4096EE"], [55, "flat", "#4096EE"], [56, "flat", "#4096EE"]] }, { "row": 11, "cells": [[40, "flat", "#4096EE"], [41, "flat", "#4096EE"], [47, "flat", "#4096EE"], [50, "flat", "#4096EE"], [51, "flat", "#4096EE"], [54, "flat", "#4096EE"], [55, "flat", "#4096EE"]] }, { "row": 12, "cells": [[40, "flat", "#4096EE"], [42, "flat", "#4096EE"], [44, "flat", "#4096EE"], [46, "flat", "#4096EE"], [49, "flat", "#4096EE"], [57, "flat", "#4096EE"]] }, { "row": 13, "cells": [[28, "flat", "#4096EE"], [30, "flat", "#4096EE"], [32, "flat", "#4096EE"], [34, "flat", "#4096EE"], [36, "flat", "#4096EE"], [37, "flat", "#4096EE"], [39, "flat", "#4096EE"], [40, "flat", "#4096EE"], [41, "flat", "#4096EE"], [42, "flat", "#4096EE"], [47, "flat", "#4096EE"], [58, "flat", "#4096EE"], [59, "flat", "#4096EE"]] }, { "row": 14, "cells": [[25, "flat", "#4096EE"], [39, "flat", "#4096EE"], [40, "flat", "#4096EE"], [43, "flat", "#4096EE"], [59, "flat", "#4096EE"]] }, { "row": 15, "cells": [[23, "flat", "#4096EE"], [36, "flat", "#4096EE"], [40, "flat", "#4096EE"], [41, "flat", "#4096EE"], [42, "flat", "#4096EE"], [44, "flat", "#4096EE"], [57, "flat", "#4096EE"], [58, "flat", "#4096EE"]] }, { "row": 16, "cells": [[21, "flat", "#4096EE"], [34, "flat", "#4096EE"], [41, "flat", "#4096EE"], [42, "flat", "#4096EE"], [43, "flat", "#4096EE"], [44, "flat", "#4096EE"], [46, "flat", "#4096EE"], [49, "flat", "#4096EE"], [56, "flat", "#4096EE"], [58, "flat", "#4096EE"]] }, { "row": 17, "cells": [[33, "flat", "#4096EE"], [41, "flat", "#4096EE"], [44, "flat", "#4096EE"], [45, "flat", "#4096EE"], [49, "flat", "#4096EE"], [51, "flat", "#4096EE"], [53, "flat", "#4096EE"], [55, "flat", "#4096EE"]] }, { "row": 18, "cells": [[19, "flat", "#4096EE"], [36, "flat", "#4096EE"], [41, "flat", "#4096EE"], [42, "flat", "#4096EE"], [43, "flat", "#4096EE"], [46, "flat", "#4096EE"], [47, "flat", "#4096EE"], [49, "flat", "#4096EE"], [56, "flat", "#4096EE"], [75, "flat", "#4096EE"]] }, { "row": 19, "cells": [[32, "flat", "#4096EE"], [33, "flat", "#4096EE"], [41, "flat", "#4096EE"], [42, "flat", "#4096EE"], [43, "flat", "#4096EE"], [44, "flat", "#4096EE"], [45, "flat", "#4096EE"], [46, "flat", "#4096EE"], [47, "flat", "#4096EE"], [48, "flat", "#4096EE"], [49, "flat", "#4096EE"], [53, "flat", "#4096EE"], [55, "flat", "#4096EE"], [56, "flat", "#4096EE"], [57, "flat", "#4096EE"]] }, { "row": 20, "cells": [[18, "flat", "#4096EE"], [40, "flat", "#4096EE"], [41, "flat", "#4096EE"], [42, "flat", "#4096EE"], [44, "flat", "#4096EE"], [45, "flat", "#4096EE"], [46, "flat", "#4096EE"], [48, "flat", "#4096EE"], [49, "flat", "#4096EE"], [50, "flat", "#4096EE"], [51, "flat", "#4096EE"], [57, "flat", "#4096EE"], [58, "flat", "#4096EE"]] }, { "row": 21, "cells": [[17, "flat", "#4096EE"], [31, "flat", "#4096EE"], [32, "flat", "#4096EE"], [39, "flat", "#4096EE"], [40, "flat", "#4096EE"], [43, "flat", "#4096EE"], [44, "flat", "#4096EE"], [46, "flat", "#4096EE"], [47, "flat", "#4096EE"], [49, "flat", "#4096EE"], [50, "flat", "#4096EE"], [51, "flat", "#4096EE"], [52, "flat", "#4096EE"], [53, "flat", "#4096EE"], [54, "flat", "#4096EE"], [55, "flat", "#4096EE"], [56, "flat", "#4096EE"], [57, "flat", "#4096EE"], [60, "flat", "#4096EE"], [62, "flat", "#4096EE"], [63, "flat", "#4096EE"], [65, "flat", "#4096EE"], [66, "flat", "#4096EE"], [68, "flat", "#4096EE"], [70, "flat", "#4096EE"], [71, "flat", "#4096EE"]] }, { "row": 22, "cells": [[30, "flat", "#4096EE"], [39, "flat", "#4096EE"], [40, "flat", "#4096EE"], [41, "flat", "#4096EE"], [42, "flat", "#4096EE"], [43, "flat", "#4096EE"], [48, "flat", "#4096EE"], [52, "flat", "#4096EE"], [53, "flat", "#4096EE"], [55, "flat", "#4096EE"], [56, "flat", "#4096EE"], [58, "flat", "#4096EE"], [59, "flat", "#4096EE"], [64, "flat", "#4096EE"], [65, "flat", "#4096EE"], [72, "flat", "#4096EE"]] }, { "row": 23, "cells": [[30, "flat",

"#4096EE"], [37, "flat", "#4096EE"], [39, "flat", "#4096EE"], [40, "flat", "#4096EE"], [41, "flat", "#4096EE"], [42, "flat", "#4096EE"], [44, "flat", "#4096EE"], [47, "flat", "#4096EE"], [49, "flat", "#4096EE"], [50, "flat", "#4096EE"], [54, "flat", "#4096EE"], [59, "flat", "#4096EE"], [61, "flat", "#4096EE"], [63, "flat", "#4096EE"], [65, "flat", "#4096EE"]] }, { "row": 24, "cells": [[16, "flat", "#4096EE"], [29, "flat", "#4096EE"], [36, "flat", "#4096EE"], [38, "flat", "#4096EE"], [39, "flat", "#4096EE"], [40, "flat", "#4096EE"], [41, "flat", "#4096EE"], [43, "flat", "#4096EE"], [44, "flat", "#4096EE"], [45, "flat", "#4096EE"], [46, "flat", "#4096EE"], [47, "flat", "#4096EE"], [50, "flat", "#4096EE"], [51, "flat", "#4096EE"], [55, "flat", "#4096EE"], [56, "flat", "#4096EE"], [57, "flat", "#4096EE"], [59, "flat", "#4096EE"], [60, "flat", "#4096EE"], [63, "flat", "#4096EE"], [64, "flat", "#4096EE"], [67, "flat", "#4096EE"]] }, { "row": 25, "cells": [[30, "flat", "#4096EE"], [33, "flat", "#4096EE"], [34, "flat", "#4096EE"], [39, "flat", "#4096EE"], [40, "flat", "#4096EE"], [42, "flat", "#4096EE"], [48, "flat", "#4096EE"], [50, "flat", "#4096EE"], [52, "flat", "#4096EE"], [53, "flat", "#4096EE"], [58, "flat", "#4096EE"], [60, "flat", "#4096EE"], [62, "flat", "#4096EE"], [68, "flat", "#4096EE"], [69, "flat", "#4096EE"]] }, { "row": 26, "cells": [[16, "flat", "#4096EE"], [28, "flat", "#4096EE"], [37, "flat", "#4096EE"], [48, "flat", "#4096EE"], [50, "flat", "#4096EE"], [54, "flat", "#4096EE"], [55, "flat", "#4096EE"], [65, "flat", "#4096EE"], [68, "flat", "#4096EE"], [70, "flat", "#4096EE"], [71, "flat", "#4096EE"], [72, "flat", "#4096EE"], [73, "flat", "#4096EE"]] }, { "row": 27, "cells": [[30, "flat", "#4096EE"], [34, "flat", "#4096EE"], [39, "flat", "#4096EE"], [42, "flat", "#4096EE"], [43, "flat", "#4096EE"], [44, "flat", "#4096EE"], [46, "flat", "#4096EE"], [49, "flat", "#4096EE"], [50, "flat", "#4096EE"], [54, "flat", "#4096EE"], [55, "flat", "#4096EE"], [60, "flat", "#4096EE"], [70, "flat", "#4096EE"], [71, "flat", "#4096EE"]] }, { "row": 28, "cells": [[16, "flat", "#4096EE"], [28, "flat", "#4096EE"], [30, "flat", "#4096EE"], [34, "flat", "#4096EE"], [37, "flat", "#4096EE"], [49, "flat", "#4096EE"], [55, "flat", "#4096EE"]] }, { "row": 29, "cells": [[17, "flat", "#4096EE"], [23, "flat", "#4096EE"], [24, "flat", "#4096EE"], [25, "flat", "#4096EE"], [28, "flat", "#4096EE"], [29, "flat", "#4096EE"], [30, "flat", "#4096EE"], [31, "flat", "#4096EE"], [38, "flat", "#4096EE"], [40, "flat", "#4096EE"], [43, "flat", "#4096EE"], [45, "flat", "#4096EE"], [47, "flat", "#4096EE"], [49, "flat", "#4096EE"], [55, "flat", "#4096EE"], [72, "flat", "#4096EE"]] }, { "row": 30, "cells": [[17, "flat", "#4096EE"], [21, "flat", "#4096EE"], [22, "flat", "#4096EE"], [24, "flat", "#4096EE"], [31, "flat", "#4096EE"], [33, "flat", "#4096EE"], [39, "flat", "#4096EE"], [47, "flat", "#4096EE"], [48, "flat", "#4096EE"], [49, "flat", "#4096EE"], [55, "flat", "#4096EE"], [72, "flat", "#4096EE"]] }, { "row": 31, "cells": [[18, "flat", "#4096EE"], [19, "flat", "#4096EE"], [20, "flat", "#4096EE"], [22, "flat", "#4096EE"], [34, "flat", "#4096EE"], [35, "flat", "#4096EE"], [37, "flat", "#4096EE"], [46, "flat", "#4096EE"], [55, "flat", "#4096EE"], [73, "flat", "#4096EE"]] }, { "row": 32, "cells": [[39, "flat", "#4096EE"], [40, "flat", "#4096EE"], [42, "flat", "#4096EE"], [44, "flat", "#4096EE"], [55, "flat", "#4096EE"], [73, "flat", "#4096EE"]] }, { "row": 33, "cells": [[39, "flat", "#4096EE"], [44, "flat", "#4096EE"], [47, "flat", "#4096EE"], [49, "flat", "#4096EE"], [51, "flat", "#4096EE"], [53, "flat", "#4096EE"], [54, "flat", "#4096EE"], [73, "flat", "#4096EE"]] }, { "row": 34, "cells": [[40, "flat", "#4096EE"], [41, "flat", "#4096EE"], [73, "flat", "#4096EE"]] }, { "row": 35, "cells": [[43, "flat", "#4096EE"], [71, "flat", "#4096EE"], [72, "flat", "#4096EE"]] }, { "row": 36, "cells": [[46, "flat", "#4096EE"], [50, "flat", "#4096EE"], [54, "flat", "#4096EE"], [59, "flat", "#4096EE"], [63, "flat", "#4096EE"], [70, "flat", "#4096EE"]] }, { "row": 37, "cells": [[66, "flat", "#4096EE"]] }] }], "recentColors": ["#4096EE", "#FFFFFF", "#000000", "#EEEEEE", "#FFFF88", "#CDEB8B", "#6BBA70", "#006E2E", "#C3D9FF", "#356AA0", "#FF0096", "#B02B2C", "#FF7400", "#EF9090", "#0099FF", "#9933FF", "#2E2EFF", "#8A725D", "#FF3838", "#4BC8D1", "#CBD114", "#858585"], "transparentBackground": false, "gridVisible": false, "additionalPixelImage": false };
var exchangeToken = "";
var exchangeUrl = "https://gp-exchange.avhost.info/socket.io"
</script>

<script src="/js/painter.js?t=2021-01-08"></script>

```
        <!-- Global site tag (gtag.js) - Google Analytics -->
        <!-- <script async
src="https://www.googletagmanager.com/gtag/js?id=UA-43623653-1"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'UA-43623653-1');
</script> -->


</body>

</html>
```

ДОДАТОК Г Java лістинг

```
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.antialiazing;
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 isAntialiazing = false;
    public static boolean isAntialiazingPro = 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(isAntialiazing){
            HexConrtoller.antialiazing();
         }
         else if(isAntialiazingPro){
            HexConrtoller.antialiazingPro();
         }
         //


         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.isAntialiazing = false;
   App.isInterpolation =false;
   App.isAntialiazingPro =true;
}
if(chooser.getValue().equals("Антиаліайзінг 7 точок")){
   checkBoxAddLine.setVisible(true);
   vBoxRadius.setDisable(true);
   App.isline = true;
   App.isAntialiazing = true;
   App.isInterpolation =false;
   App.isAntialiazingPro =false;
}
else if(chooser.getValue().equals("Лінія")){
   checkBoxAddLine.setVisible(true);
```

```
        vBoxRadius.setDisable(true);
        App.isline = true;
        App.isAntialiazing = false;
        App.isInterpolation =true;
        App.isAntialiazingPro =false;
        //interpolationHexagonLine();

    }
    else if(chooser.getValue().equals("Коло")){
        checkBoxAddLine.setVisible(false);
        vBoxRadius.setDisable(false);
        App.isline = false;
        interpolationHexagonCircle();
        App.isAntialiazing = false;
        App.isInterpolation =false;
        App.isAntialiazingPro =false;
    }
    else if(chooser.getValue().equals("Коло Pro")){
        checkBoxAddLine.setVisible(false);
        vBoxRadius.setDisable(false);
        App.isline = false;
        interpolationHexagonCirclePro();
        App.isAntialiazing = false;
        App.isInterpolation =false;
        App.isAntialiazingPro =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;
 printInTextField("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;

               printInTextField("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();

            printInTextField("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;

            printInTextField("діагонально вліво P/6_P/2 | 30° до 90°");
            System.out.print(" діагонально вліво P/6_P/2 | 30° до 90°");
        }


    }
}

public void interpolationHexagonCirclePro(){
    double tabX = (R+15)  *  Hex.getWidth();
    double tabY = (R + 40)  *  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;

int intensivtistColory = 0;
double OF1 = 0;
double OF2 = 0;
double OF3 = 0;
double OF4 = 0;
double OF5 = 0;
double OF6 = 0;
double OF7 = 0;



//double x , y;

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);
   intensivtistColory = 0;
   OF1 = (x)*(x)
         + (y+ 0.25 * Hex.getHeight())*(y+ 0.25 * Hex.getHeight())
         - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
   //OF2 = (y + 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
   OF2 = (x+0.5*Hex.getWidth())*(x+0.5*Hex.getWidth())
         + (y+ 0.25 * Hex.getHeight())*(y+ 0.25 * Hex.getHeight() )
         - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
```

```
//OF3 = (y - 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
OF3 = (x+0.5*Hex.getWidth())*(x+0.5*Hex.getWidth())
      + (y- 0.25 * Hex.getHeight())*(y- 0.25 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF4 = (y - 0.5 * Hex.getWidth()) * dX - (x) * dY;
OF4 = (x)*(x)
      + (y- 0.5 * Hex.getHeight())*(y -0.5 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF5 = (y - 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF5 = (x-0.5*Hex.getWidth())*(x-0.5*Hex.getWidth())
      + (y- 0.25 * Hex.getHeight())*(y -0.25 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF6 = (y + 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF6 = (x-0.5*Hex.getWidth())*(x-0.5*Hex.getWidth())
      + (y+ 0.25* Hex.getHeight())*(y + 0.25* Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF7 = (y) * dX - (x) * dY;
OF7 = (x)*(x)
      + (y)*(y)
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
      //(y + 0.5 * Hex.getWidth()) * dX - (x) * dY;
if (OF1 > 0) {
   intensivtistColory++;
}
if (OF2 > 0) {
   intensivtistColory++;
}
if (OF3 > 0) {
   intensivtistColory++;
}
```

```
        if (OF4 > 0) {
            intensivtistColory++;
        }
        if (OF5 > 0) {
            intensivtistColory++;
        }
        if (OF6 > 0) {
            intensivtistColory++;
        }
    if (OF7 > 0) {
            intensivtistColory++;
        }
        double coefColor = 0.0;
        switch (intensivtistColory) {
            case 0:
                coefColor = 0.0;
                break;
            case 1:
                coefColor = 0.14;
                break;
            case 2:
                coefColor = 0.28;
                break;
            case 3:
                coefColor = 0.42;
                break;
            case 4:
                coefColor = 0.57;
                break;
            case 5:
```

```
            coefColor = 0.71;
            break;
        case 6:
            coefColor = 0.85;
            break;
        case 7:
            coefColor = 1.0;
            break;
        default:
            coefColor = 0.0;
            break;
    }
    Hex.setFilledColor(Color.hsb(filledColor.getHue(), coefColor, 1.0));
    h.filled = true;
    h.print();
    printInTextField("Позиція на гексагону у вікні: (x=" + (x+tabX) + " y=" +
(y+tabY) + "), коефіцієнт зафарбування: " + coefColor);




    h = new Hex(x+tabX,-y+tabY);
    intensivtistColory = 0;
    OF1 = (x)*(x)
        + (-y+ 0.25 * Hex.getHeight())*(-y+ 0.25 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
    //OF2 = (y + 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
    OF2 = (x+0.5*Hex.getWidth())*(x+0.5*Hex.getWidth())
        + (-y+ 0.25 * Hex.getHeight())*(-y+ 0.25 * Hex.getHeight() )
```

```
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF3 = (y - 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
OF3 = (x+0.5*Hex.getWidth())*(x+0.5*Hex.getWidth())
        + (-y- 0.25 * Hex.getHeight())*(-y- 0.25 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF4 = (y - 0.5 * Hex.getWidth()) * dX - (x) * dY;
OF4 = (x)*(x)
        + (-y- 0.5 * Hex.getHeight())*(-y -0.5 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF5 = (y - 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF5 = (x-0.5*Hex.getWidth())*(x-0.5*Hex.getWidth())
        + (-y- 0.25 * Hex.getHeight())*(-y -0.25 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF6 = (y + 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF6 = (x-0.5*Hex.getWidth())*(x-0.5*Hex.getWidth())
        + (-y+ 0.25* Hex.getHeight())*(-y + 0.25* Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF7 = (y) * dX - (x) * dY;
OF7 = (x)*(x)
        + (-y)*(-y)
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//(y + 0.5 * Hex.getWidth()) * dX - (x) * dY;
if (OF1 > 0) {
   intensivtistColory++;
}
if (OF2 > 0) {
   intensivtistColory++;
}
if (OF3 > 0) {
   intensivtistColory++;
```

```
}
if (OF4 > 0) {
    intensivtistColory++;
}
if (OF5 > 0) {
    intensivtistColory++;
}
if (OF6 > 0) {
    intensivtistColory++;
}
if (OF7 > 0) {
    intensivtistColory++;
}
coefColor = 0.0;
switch (intensivtistColory) {
    case 0:
        coefColor = 0.0;
        break;
    case 1:
        coefColor = 0.14;
        break;
    case 2:
        coefColor = 0.28;
        break;
    case 3:
        coefColor = 0.42;
        break;
    case 4:
        coefColor = 0.57;
        break;
```

```
    case 5:
        coefColor = 0.71;
        break;
    case 6:
        coefColor = 0.85;
        break;
    case 7:
        coefColor = 1.0;
        break;
    default:
        coefColor = 0.0;
        break;
}
Hex.setFilledColor(Color.hsb(filledColor.getHue(), coefColor, 1.0));
h.filled = true;
h.print();




h = new Hex(-x+tabX,-y+tabY);
intensivtistColory = 0;
OF1 = (-x)*(-x)
        + (-y+ 0.25 * Hex.getHeight())*(-y+ 0.25 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF2 = (y + 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
OF2 = (-x+0.5*Hex.getWidth())*(-x+0.5*Hex.getWidth())
        + (-y+ 0.25 * Hex.getHeight())*(-y+ 0.25 * Hex.getHeight() )
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF3 = (y - 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
```

```
        OF3 = (-x+0.5*Hex.getWidth())*(-x+0.5*Hex.getWidth())
              + (-y- 0.25 * Hex.getHeight())*(-y- 0.25 * Hex.getHeight())
              - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
        //OF4 = (y - 0.5 * Hex.getWidth()) * dX - (x) * dY;
        OF4 = (-x)*(-x)
              + (-y- 0.5 * Hex.getHeight())*(-y -0.5 * Hex.getHeight())
              - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
        //OF5 = (y - 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
        OF5 = (-x-0.5*Hex.getWidth())*(-x-0.5*Hex.getWidth())
+ (-y- 0.25 * Hex.getHeight())*(-y -0.25 * Hex.getHeight())
              - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
        //OF6 = (y + 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
        OF6 = (-x-0.5*Hex.getWidth())*(-x-0.5*Hex.getWidth())
              + (-y+ 0.25* Hex.getHeight())*(-y + 0.25* Hex.getHeight())
              - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
        //OF7 = (y) * dX - (x) * dY;
        OF7 = (-x)*(-x)
              + (-y)*(-y)
              - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
        //(y + 0.5 * Hex.getWidth()) * dX - (x) * dY;
        if (OF1 > 0) {
           intensivtistColory++;
        }
        if (OF2 > 0) {
           intensivtistColory++;
        }
        if (OF3 > 0) {
           intensivtistColory++;
        }
        if (OF4 > 0) {
```

```
    intensivtistColory++;
}
if (OF5 > 0) {
    intensivtistColory++;
}
if (OF6 > 0) {
    intensivtistColory++;
}
if (OF7 > 0) {
    intensivtistColory++;
}
coefColor = 0.0;
switch (intensivtistColory) {
    case 0:
        coefColor = 0.0;
        break;
    case 1:
        coefColor = 0.14;
        break;
    case 2:
        coefColor = 0.28;
        break;
    case 3:
        coefColor = 0.42;
        break;
    case 4:
        coefColor = 0.57;
        break;
    case 5:
        coefColor = 0.71;
```

```
        break;
    case 6:
        coefColor = 0.85;
        break;
    case 7:
        coefColor = 1.0;
        break;
    default:
        coefColor = 0.0;
        break;
}
Hex.setFilledColor(Color.hsb(filledColor.getHue(), coefColor, 1.0));
h.filled = true;
h.print();



h = new Hex(-x+tabX,y+tabY);
intensivtistColory = 0;
OF1 = (x)*(x)
        + (-y+ 0.25 * Hex.getHeight())*(-y+ 0.25 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF2 = (y + 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
OF2 = (x+0.5*Hex.getWidth())*(x+0.5*Hex.getWidth())
        + (-y+ 0.25 * Hex.getHeight())*(-y+ 0.25 * Hex.getHeight() )
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF3 = (y - 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
OF3 = (x+0.5*Hex.getWidth())*(x+0.5*Hex.getWidth())
        + (-y- 0.25 * Hex.getHeight())*(-y- 0.25 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
```

```
//OF4 = (y - 0.5 * Hex.getWidth()) * dX - (x) * dY;
OF4 = (x)*(x)
      + (-y- 0.5 * Hex.getHeight())*(-y -0.5 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF5 = (y - 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF5 = (x-0.5*Hex.getWidth())*(x-0.5*Hex.getWidth())
      + (-y- 0.25 * Hex.getHeight())*(-y -0.25 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF6 = (y + 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF6 = (x-0.5*Hex.getWidth())*(x-0.5*Hex.getWidth())
      + (-y+ 0.25* Hex.getHeight())*(-y + 0.25* Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF7 = (y) * dX - (x) * dY;
OF7 = (x)*(x)
      + (-y)*(-y)
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//(y + 0.5 * Hex.getWidth()) * dX - (x) * dY;
if (OF1 > 0) {
   intensivtistColory++;
}
if (OF2 > 0) {
   intensivtistColory++;
}
if (OF3 > 0) {
   intensivtistColory++;
}
if (OF4 > 0) {
   intensivtistColory++;
}
if (OF5 > 0) {
```

```
    intensivtistColory++;
}
if (OF6 > 0) {
    intensivtistColory++;
}
if (OF7 > 0) {
    intensivtistColory++;
}
coefColor = 0.0;
switch (intensivtistColory) {
    case 0:
        coefColor = 0.0;
        break;
    case 1:
        coefColor = 0.14;
        break;
    case 2:
        coefColor = 0.28;
        break;
    case 3:
        coefColor = 0.42;
        break;
    case 4:
        coefColor = 0.57;
        break;
    case 5:
        coefColor = 0.71;
        break;
    case 6:
        coefColor = 0.85;
```

```
            break;
        case 7:
            coefColor = 1.0;
            break;
        default:
            coefColor = 0.0;
            break;
    }
    Hex.setFilledColor(Color.hsb(filledColor.getHue(), coefColor, 1.0));
h.filled = true;
    h.print();




    h = new Hex(-x+tabX,-y+tabY);
    intensivtistColory = 0;
    OF1 = (-x)*(-x)
        + (y+ 0.25 * Hex.getHeight())*(y+ 0.25 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
    //OF2 = (y + 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
    OF2 = (-x+0.5*Hex.getWidth())*(-x+0.5*Hex.getWidth())
        + (y+ 0.25 * Hex.getHeight())*(y+ 0.25 * Hex.getHeight() )
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
    //OF3 = (y - 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
    OF3 = (-x+0.5*Hex.getWidth())*(-x+0.5*Hex.getWidth())
        + (y- 0.25 * Hex.getHeight())*(y- 0.25 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
    //OF4 = (y - 0.5 * Hex.getWidth()) * dX - (x) * dY;
    OF4 = (-x)*(-x)
```

```
        + (y- 0.5 * Hex.getHeight())*(y -0.5 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF5 = (y - 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF5 = (-x-0.5*Hex.getWidth())*(-x-0.5*Hex.getWidth())
        + (y- 0.25 * Hex.getHeight())*(y -0.25 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF6 = (y + 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF6 = (-x-0.5*Hex.getWidth())*(-x-0.5*Hex.getWidth())
        + (y+ 0.25* Hex.getHeight())*(y + 0.25* Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF7 = (y) * dX - (x) * dY;
OF7 = (-x)*(-x)
        + (y)*(y)
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//(y + 0.5 * Hex.getWidth()) * dX - (x) * dY;
if (OF1 > 0) {
   intensivtistColory++;
}
if (OF2 > 0) {
   intensivtistColory++;
}
if (OF3 > 0) {
   intensivtistColory++;
}
if (OF4 > 0) {
   intensivtistColory++;
}
if (OF5 > 0) {
   intensivtistColory++;
}
```

```
if (OF6 > 0) {
   intensivtistColory++;
}
if (OF7 > 0) {
   intensivtistColory++;
}
coefColor = 0.0;
switch (intensivtistColory) {
   case 0:
      coefColor = 0.0;
      break;
   case 1:
      coefColor = 0.14;
      break;
   case 2:
      coefColor = 0.28;
      break;
   case 3:
      coefColor = 0.42;
      break;
   case 4:
      coefColor = 0.57;
      break;
   case 5:
      coefColor = 0.71;
      break;
   case 6:
      coefColor = 0.85;
      break;
   case 7:
```

```
                coefColor = 1.0;
                break;
            default:
                coefColor = 0.0;
                break;
        }
        Hex.setFilledColor(Color.hsb(filledColor.getHue(), coefColor, 1.0));
        h.filled = true;
        h.print();

        Hex.setFilledColor(filledColor);



        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;

           printInTextField("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;

            printInTextField("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);
        intensivtistColory = 0;
        OF1 = (x)*(x)
                + (y+ 0.25 * Hex.getHeight())*(y+ 0.25 * Hex.getHeight())
                - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
        //OF2 = (y + 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
        OF2 = (x+0.5*Hex.getWidth())*(x+0.5*Hex.getWidth())
                + (y+ 0.25 * Hex.getHeight())*(y+ 0.25 * Hex.getHeight() )
                - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
        //OF3 = (y - 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
        OF3 = (x+0.5*Hex.getWidth())*(x+0.5*Hex.getWidth())
                + (y- 0.25 * Hex.getHeight())*(y- 0.25 * Hex.getHeight())
                - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
        //OF4 = (y - 0.5 * Hex.getWidth()) * dX - (x) * dY;
        OF4 = (x)*(x)
                + (y- 0.5 * Hex.getHeight())*(y -0.5 * Hex.getHeight())
                - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
```

```
//OF5 = (y - 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF5 = (x-0.5*Hex.getWidth())*(x-0.5*Hex.getWidth())
      + (y- 0.25 * Hex.getHeight())*(y -0.25 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF6 = (y + 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF6 = (x-0.5*Hex.getWidth())*(x-0.5*Hex.getWidth())
      + (y+ 0.25* Hex.getHeight())*(y + 0.25* Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF7 = (y) * dX - (x) * dY;
OF7 = (x)*(x)
      + (y)*(y)
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//(y + 0.5 * Hex.getWidth()) * dX - (x) * dY;
if (OF1 > 0) {
   intensivtistColory++;
}
if (OF2 > 0) {
   intensivtistColory++;
}
if (OF3 > 0) {
   intensivtistColory++;
}
if (OF4 > 0) {
   intensivtistColory++;
}
if (OF5 > 0) {
   intensivtistColory++;
}
if (OF6 > 0) {
   intensivtistColory++;
```

```
}
if (OF7 > 0) {
    intensivtistColory++;
}
double coefColor = 0.0;
switch (intensivtistColory) {
    case 0:
        coefColor = 0.0;
        break;
    case 1:
        coefColor = 0.14;
        break;
    case 2:
        coefColor = 0.28;
        break;
    case 3:
        coefColor = 0.42;
        break;
    case 4:
        coefColor = 0.57;
        break;
    case 5:
        coefColor = 0.71;
        break;
    case 6:
        coefColor = 0.85;
        break;
    case 7:
        coefColor = 1.0;
        break;
```

```
    default:
        coefColor = 0.0;
        break;
}
Hex.setFilledColor(Color.hsb(filledColor.getHue(), coefColor, 1.0));
h.filled = true;
h.print();




h = new Hex(x+tabX,-y+tabY);
intensivtistColory = 0;
OF1 = (x)*(x)
        + (-y+ 0.25 * Hex.getHeight())*(-y+ 0.25 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF2 = (y + 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
OF2 = (x+0.5*Hex.getWidth())*(x+0.5*Hex.getWidth())
        + (-y+ 0.25 * Hex.getHeight())*(-y+ 0.25 * Hex.getHeight() )
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF3 = (y - 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
OF3 = (x+0.5*Hex.getWidth())*(x+0.5*Hex.getWidth())
        + (-y- 0.25 * Hex.getHeight())*(-y- 0.25 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF4 = (y - 0.5 * Hex.getWidth()) * dX - (x) * dY;
OF4 = (x)*(x)
        + (-y- 0.5 * Hex.getHeight())*(-y -0.5 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF5 = (y - 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
```

```
OF5 = (x-0.5*Hex.getWidth())*(x-0.5*Hex.getWidth())
      + (-y- 0.25 * Hex.getHeight())*(-y -0.25 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF6 = (y + 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF6 = (x-0.5*Hex.getWidth())*(x-0.5*Hex.getWidth())
      + (-y+ 0.25* Hex.getHeight())*(-y + 0.25* Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF7 = (y) * dX - (x) * dY;
OF7 = (x)*(x)
      + (-y)*(-y)
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//(y + 0.5 * Hex.getWidth()) * dX - (x) * dY;
if (OF1 > 0) {
   intensivtistColory++;
}
if (OF2 > 0) {
   intensivtistColory++;
}
if (OF3 > 0) {
   intensivtistColory++;
}
if (OF4 > 0) {
   intensivtistColory++;
}
if (OF5 > 0) {
   intensivtistColory++;
}
if (OF6 > 0) {
   intensivtistColory++;
}
```

```
        if (OF7 > 0) {
            intensivtistColory++;
        }
        coefColor = 0.0;
        switch (intensivtistColory) {
            case 0:
                coefColor = 0.0;
break;
            case 1:
                coefColor = 0.14;
                break;
            case 2:
                coefColor = 0.28;
                break;
            case 3:
                coefColor = 0.42;
                break;
            case 4:
                coefColor = 0.57;
                break;
            case 5:
                coefColor = 0.71;
                break;
            case 6:
                coefColor = 0.85;
                break;
            case 7:
                coefColor = 1.0;
                break;
            default:
```

```
        coefColor = 0.0;
        break;
}
Hex.setFilledColor(Color.hsb(filledColor.getHue(), coefColor, 1.0));
h.filled = true;
h.print();



h = new Hex(-x+tabX,-y+tabY);
intensivtistColory = 0;
OF1 = (-x)*(-x)
        + (-y+ 0.25 * Hex.getHeight())*(-y+ 0.25 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF2 = (y + 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
OF2 = (-x+0.5*Hex.getWidth())*(-x+0.5*Hex.getWidth())
        + (-y+ 0.25 * Hex.getHeight())*(-y+ 0.25 * Hex.getHeight() )
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF3 = (y - 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
OF3 = (-x+0.5*Hex.getWidth())*(-x+0.5*Hex.getWidth())
        + (-y- 0.25 * Hex.getHeight())*(-y- 0.25 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF4 = (y - 0.5 * Hex.getWidth()) * dX - (x) * dY;
OF4 = (-x)*(-x)
        + (-y- 0.5 * Hex.getHeight())*(-y -0.5 * Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF5 = (y - 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF5 = (-x-0.5*Hex.getWidth())*(-x-0.5*Hex.getWidth())
        + (-y- 0.25 * Hex.getHeight())*(-y -0.25 * Hex.getHeight())
```

```
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF6 = (y + 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF6 = (-x-0.5*Hex.getWidth())*(-x-0.5*Hex.getWidth())
        + (-y+ 0.25* Hex.getHeight())*(-y + 0.25* Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF7 = (y) * dX - (x) * dY;
OF7 = (-x)*(-x)
        + (-y)*(-y)
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//(y + 0.5 * Hex.getWidth()) * dX - (x) * dY;
if (OF1 > 0) {
   intensivtistColory++;
}
if (OF2 > 0) {
   intensivtistColory++;
}
if (OF3 > 0) {
   intensivtistColory++;
}
if (OF4 > 0) {
   intensivtistColory++;
}
if (OF5 > 0) {
   intensivtistColory++;
}
if (OF6 > 0) {
   intensivtistColory++;
}
if (OF7 > 0) {
   intensivtistColory++;
```

```
}
coefColor = 0.0;
switch (intensivtistColory) {
    case 0:
        coefColor = 0.0;
        break;
    case 1:
        coefColor = 0.14;
        break;
    case 2:
        coefColor = 0.28;
        break;
    case 3:
        coefColor = 0.42;
        break;
    case 4:
        coefColor = 0.57;
        break;
    case 5:
        coefColor = 0.71;
        break;
    case 6:
        coefColor = 0.85;
        break;
    case 7:
        coefColor = 1.0;
        break;
    default:
        coefColor = 0.0;
        break;
```

```
}
Hex.setFilledColor(Color.hsb(filledColor.getHue(), coefColor, 1.0));
h.filled = true;
h.print();



h = new Hex(-x+tabX,y+tabY);
intensivtistColory = 0;
OF1 = (x)*(x)
      + (-y+ 0.25 * Hex.getHeight())*(-y+ 0.25 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF2 = (y + 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
OF2 = (x+0.5*Hex.getWidth())*(x+0.5*Hex.getWidth())
      + (-y+ 0.25 * Hex.getHeight())*(-y+ 0.25 * Hex.getHeight() )
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF3 = (y - 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
OF3 = (x+0.5*Hex.getWidth())*(x+0.5*Hex.getWidth())
      + (-y- 0.25 * Hex.getHeight())*(-y- 0.25 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF4 = (y - 0.5 * Hex.getWidth()) * dX - (x) * dY;
OF4 = (x)*(x)
      + (-y- 0.5 * Hex.getHeight())*(-y -0.5 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF5 = (y - 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF5 = (x-0.5*Hex.getWidth())*(x-0.5*Hex.getWidth())
      + (-y- 0.25 * Hex.getHeight())*(-y -0.25 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF6 = (y + 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF6 = (x-0.5*Hex.getWidth())*(x-0.5*Hex.getWidth())
```

```
        + (-y+ 0.25* Hex.getHeight())*(-y + 0.25* Hex.getHeight())
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF7 = (y) * dX - (x) * dY;
OF7 = (x)*(x)
        + (-y)*(-y)
        - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//(y + 0.5 * Hex.getWidth()) * dX - (x) * dY;
if (OF1 > 0) {
   intensivtistColory++;
}
if (OF2 > 0) {
   intensivtistColory++;
}
if (OF3 > 0) {
   intensivtistColory++;
}
if (OF4 > 0) {
   intensivtistColory++;
}
if (OF5 > 0) {
   intensivtistColory++;
}
if (OF6 > 0) {
   intensivtistColory++;
}
if (OF7 > 0) {
   intensivtistColory++;
}
coefColor = 0.0;
switch (intensivtistColory) {
```

```
    case 0:
        coefColor = 0.0;
        break;
    case 1:
        coefColor = 0.14;
        break;
    case 2:
        coefColor = 0.28;
        break;
    case 3:
        coefColor = 0.42;
        break;
    case 4:
        coefColor = 0.57;
        break;
    case 5:
        coefColor = 0.71;
        break;
    case 6:
        coefColor = 0.85;
        break;
    case 7:
        coefColor = 1.0;
        break;
    default:
        coefColor = 0.0;
        break;
}
Hex.setFilledColor(Color.hsb(filledColor.getHue(), coefColor, 1.0));
h.filled = true;
```

```
h.print();



h = new Hex(-x+tabX,-y+tabY);
intensivtistColory = 0;
OF1 = (-x)*(-x)
      + (y+ 0.25 * Hex.getHeight())*(y+ 0.25 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF2 = (y + 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
OF2 = (-x+0.5*Hex.getWidth())*(-x+0.5*Hex.getWidth())
      + (y+ 0.25 * Hex.getHeight())*(y+ 0.25 * Hex.getHeight() )
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF3 = (y - 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
OF3 = (-x+0.5*Hex.getWidth())*(-x+0.5*Hex.getWidth())
      + (y- 0.25 * Hex.getHeight())*(y- 0.25 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF4 = (y - 0.5 * Hex.getWidth()) * dX - (x) * dY;
OF4 = (-x)*(-x)
      + (y- 0.5 * Hex.getHeight())*(y -0.5 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF5 = (y - 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF5 = (-x-0.5*Hex.getWidth())*(-x-0.5*Hex.getWidth())
      + (y- 0.25 * Hex.getHeight())*(y -0.25 * Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
//OF6 = (y + 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
OF6 = (-x-0.5*Hex.getWidth())*(-x-0.5*Hex.getWidth())
      + (y+ 0.25* Hex.getHeight())*(y + 0.25* Hex.getHeight())
      - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
```

```
        //OF7 = (y) * dX - (x) * dY;
        OF7 = (-x)*(-x)
              + (y)*(y)
              - ((double)(R*R) * Hex.getWidth()*Hex.getWidth());
        //(y + 0.5 * Hex.getWidth()) * dX - (x) * dY;
        if (OF1 > 0) {
           intensivtistColory++;
}
        if (OF2 > 0) {
           intensivtistColory++;
        }
        if (OF3 > 0) {
           intensivtistColory++;
        }
        if (OF4 > 0) {
           intensivtistColory++;
        }
        if (OF5 > 0) {
           intensivtistColory++;
        }
        if (OF6 > 0) {
           intensivtistColory++;
        }
        if (OF7 > 0) {
           intensivtistColory++;
        }
        coefColor = 0.0;
        switch (intensivtistColory) {
           case 0:
              coefColor = 0.0;
```

```
            break;
        case 1:
            coefColor = 0.14;
            break;
        case 2:
            coefColor = 0.28;
            break;
        case 3:
            coefColor = 0.42;
            break;
        case 4:
            coefColor = 0.57;
            break;
        case 5:
            coefColor = 0.71;
            break;
        case 6:
            coefColor = 0.85;
            break;
        case 7:
            coefColor = 1.0;
            break;
        default:
            coefColor = 0.0;
            break;
    }
    Hex.setFilledColor(Color.hsb(filledColor.getHue(), coefColor, 1.0));
    h.filled = true;
    h.print();
```

```
        Hex.setFilledColor(filledColor);

        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();

           printInTextField("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;

            printInTextField("діагонально вліво P/6_P/2 | 30° до 90°");
            System.out.print(" діагонально вліво P/6_P/2 | 30° до 90°");
        }


    }
}
public static void colo_(){
    double tabX = 20*Hex.getWidth();
    double tabY = 20*Hex.getHeight()*0.75;


    //createNewScreen();
    int R = 15;
    double x = R * Hex.getWidth();
    double y = 0;
    double pi = 3.14159265;

    Hex h = new Hex(tabX,tabY);
    h.filled = true;
    h.print();




    double c = Math.sqrt(x*x+y*y);
    if (0.5 >=  (double) (dX) / c) {
        //cosBilshe60 = true;
```

```
        }
        //boolean alphaMenshe30 = true;

        double OF = 0;



        while (//y <= pi/6 * Hex.getHeight()*0.75){
         pi/6 < (double) x / Math.sqrt(x*x+y*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);
            OF = Math.pow( Math.pow(x ,2) + Math.pow(y ,2) ,2)- R * Hex.getWidth() * R
* (Hex.getHeight() - Hex.getHeight()/4) ;
            double OFdL = Math.pow( Math.pow(x - 0.5* Hex.getWidth() ,2) + Math.pow(y
+ (3.0/(2.0* Math.sqrt(3)))* (Hex.getHeight() - Hex.getHeight()/4),2) ,2)- R *
Hex.getWidth() * R * (Hex.getHeight() - Hex.getHeight()/4) ;
                    //OF - x + Math.sqrt(3) * y + 1;
            double OFdR = Math.pow( Math.pow(x + 0.5* Hex.getWidth() ,2) + Math.pow(y
```

```
+ (3.0/(2.0* Math.sqrt(3)))* (Hex.getHeight() - Hex.getHeight()/4),2) ,2)- R *
Hex.getWidth() * R * (Hex.getHeight() - Hex.getHeight()/4) ;
                //OF + x + Math.sqrt(3) * y + 1;
            if(OFdR < OFdL){
               OF = OFdR;
               x+= Hex.getWidth()/2;
               y+= Hex.getHeight() - Hex.getHeight()/4;
               System.out.print(" діагонально вправо 0_P/6 | 0° до 30°");
            }
            else {
               OF = OFdL;
               x-= Hex.getWidth()/2;
               y+= Hex.getHeight() - Hex.getHeight()/4;

               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(  Math.pow( Math.pow(x - 0.5 ,2) + Math.pow(y +
3.0/(2.0* Math.sqrt(3)),2) ,2)- R*R );
        // OF - x + Math.sqrt(3) * y + 1;
        double OFhL = Math.abs( Math.pow( Math.pow(x - 1.0 ,2) + Math.pow(y,2) ,2)-
R*R);
              //OF -2 * x + 1;
        if(OFhL < OFdL){
           OF = OFhL;
           x-= Hex.getWidth();

           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;

           System.out.print(" діагонально вліво P/6_P/2 | 30° до 90°");
        }
```

```
    }
}
public static void coloCopy(){
    double tabX = 20*Hex.getWidth();
    double tabY = 20*Hex.getHeight()*0.75;



    //createNewScreen();
    int R = 15;
    double x = R * Hex.getWidth();
    double y = 0;
    double pi = 3.14159265;

    Hex h = new Hex(tabX,tabY);
    h.filled = true;
    h.print();




    double c = Math.sqrt(x*x+y*y);
    if (0.5 >=  (double) (dX) / c) {
        //cosBilshe60 = true;
    }
    //boolean alphaMenshe30 = true;

    double OF = 0;
```

```
        while (//y <= pi/6 * Hex.getHeight()*0.75){
            pi/6 < (double) x / Math.sqrt(x*x+y*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( Math.pow( Math.pow(x - 0.5 ,2) + Math.pow(y -
3.0/(2.0* Math.sqrt(3)),2) ,2)- R*R );
            //OF - x + Math.sqrt(3) * y + 1;
            double OFdR = Math.abs( Math.pow( Math.pow(x + 0.5 ,2) + Math.pow(y -
3.0/(2.0* Math.sqrt(3)),2) ,2)- R*R );
            //OF + x + Math.sqrt(3) * y + 1;
            if(OFdR < OFdL){
               OF = OFdR;
               x+= Hex.getWidth()/2;
               y+= Hex.getHeight() - Hex.getHeight()/4;
               System.out.print(" діагонально вправо 0_P/6 | 0° до 30°");
            }
```

```
    else {
        OF = OFdL;
        x-= Hex.getWidth()/2;
        y+= Hex.getHeight() - Hex.getHeight()/4;

        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(  Math.pow( Math.pow(x - 0.5 ,2) + Math.pow(y +
3.0/(2.0* Math.sqrt(3)),2) ,2)- R*R );
            // OF - x + Math.sqrt(3) * y + 1;
            double OFhL = Math.abs( Math.pow( Math.pow(x - 1.0 ,2) + Math.pow(y,2) ,2)-
R*R);
            //OF -2 * x + 1;
            if(OFhL < OFdL){
               OF = OFhL;
               x-= Hex.getWidth();

               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;

               System.out.print(" діагонально вліво P/6_P/2 | 30° до 90°");
            }
         }
      }

      static void createNewScreen() {
         App.workSpace.getChildren().clear();
         fieldHexagon = new Hex[columns][lines];
         for (int i = 0; i < columns; i++) {
            for (int j = 0; j < lines; j++) {
               fieldHexagon[i][j] = new Hex(i,j);
               System.out.println(i+" "+j);
```

```
        }
    }
}
```

```
public static void clearFilledHexagon() {
    for (int i = 0; i < columns; i++) {
        for (int j = 0; j < lines; j++) {
            fieldHexagon[i][j].filled =false;
        }
    }
}
```

```
public static void antialiazing() {
    App.workSpace.getChildren().clear();

    //dX = 10 * Hex.getWidth();
    //dY = 10 * Hex.getHeight();
    int dXint = (int) (dX/Hex.getWidth());
    int dYint = (int) (dY/Hex.getHeight());

    double posX = 0;
```

```
double posY = 0;
double OFij = 0;
double OF0 = 0;

int intensivtistColory = 0;
double OF1 = 0;
double OF2 = 0;
double OF3 = 0;
double OF4 = 0;
double OF5 = 0;
double OF6 = 0;
double OF7 = 0;

double x , y;

fieldHexagon = new Hex[columns][lines];
for (int i = 0; i < columns; i++) {
   for (int j = 0; j < lines; j++) {
      intensivtistColory = 0;
      if(j%2 == 0){
         fieldHexagon[i][j] =
               new Hex(i * Hex.getWidth(),
                     j * Hex.getHeight()*3/4    );
         x = i * Hex.getWidth();
         y = j * Hex.getHeight()*3/4;
      }
      else {
         fieldHexagon[i][j] =
               new Hex(i * Hex.getWidth() -Hex.getWidth()/2,
                     j * Hex.getHeight()*3/4   );
```

```
            x = i * Hex.getWidth()-Hex.getWidth()/2;
            y = j * Hex.getHeight()*3/4;
         }
//
         if(x <= dX && y <= dY){
            OFij = y * dX  - x * dY;
//
            OF1 = (y + 0.5 * Hex.getWidth()) * dX  - (x) * dY;
            OF2 = (y + 0.25 * Hex.getWidth()) * dX  - (x + 0.5* Hex.getHeight()) * dY;
            OF3 = (y - 0.25 * Hex.getWidth()) * dX  - (x + 0.5* Hex.getHeight()) * dY;
            OF4 = (y - 0.5 * Hex.getWidth()) * dX  - (x) * dY;
            OF5 = (y - 0.25 * Hex.getWidth()) * dX  - (x - 0.5* Hex.getHeight()) * dY;
            OF6 = (y + 0.25 * Hex.getWidth()) * dX  - (x - 0.5* Hex.getHeight()) * dY;
            OF7 = (y) * dX  - (x) * dY;
//
            //OF1 = (y + 0.5) * dX  - (x - 0.25) * dY;
            //OF1 = OFij +( (2) * dX + dY)/4;
            //OF2 = OFij +( (2) * dX - dY)/4;
            //OF3 = OFij -dY/2;
            //OF4 = OFij -( (2) * dX - dY)/4;
            //OF5 = OFij -( (2) * dX + dY)/4;
            //OF6 = OFij +dY/2;
            //OF7 = OFij;
            //boolean boundsPixel = false;
//
            if(OF1 > 0){
               intensivtistColory++;
            }
            if(OF2 > 0){
               intensivtistColory++;
```

```
}
if(OF3 > 0){
   intensivtistColory++;
}
if(OF4 > 0){
   intensivtistColory++;
}
if(OF5 > 0){
   intensivtistColory++;
}
if(OF6 > 0){
   intensivtistColory++;
}
if(OF7 > 0){
   intensivtistColory++;
}
double coefColor = 0.0;
switch (intensivtistColory){
   case 0:
      coefColor = 0.0;
      break;
   case 1:
      coefColor = 0.14;
      break;
   case 2:
      coefColor = 0.28;
      break;
   case 3:
      coefColor = 0.42;
      break;
```

```
                    case 4:
                        coefColor = 0.57;
                        break;
                    case 5:
                        coefColor = 0.71;
                        break;
                    case 6:
                        coefColor = 0.85;
                        break;
                    case 7:
                        coefColor = 1.0;
                        break;
                    default:
                        coefColor = 0.0;
                        break;
                }
//
                    Hex.setFilledColor(Color.hsb(filledColor.getHue(),coefColor,1.0));
                    fieldHexagon[i][j].filled = true;


                }
                fieldHexagon[i][j].print();
                Hex.setFilledColor(filledColor);
                //System.out.println(i+" "+j );
            }
        }
    }

    public static void antialiazingPro() {
```

```
App.workSpace.getChildren().clear();


//dX = 10 * Hex.getWidth();
//dY = 10 * Hex.getHeight();
int dXint = (int) (dX / Hex.getWidth());
int dYint = (int) (dY / Hex.getHeight());


double posX = 0;
double posY = 0;
double OFij = 0;
double OF0 = 0;

int intensivtistColory = 0;
double OF1 = 0;
double OF2 = 0;
double OF3 = 0;
double OF4 = 0;
double OF5 = 0;
double OF6 = 0;
double OF7 = 0;
double OF8 = 0;
double OF9 = 0;
double OF10 = 0;
double OF11 = 0;
double OF12 = 0;


double x, y;
```

```
        fieldHexagon = new Hex[columns][lines];
        for (int i = 0; i < columns; i++) {
            for (int j = 0; j < lines; j++) {
                intensivtistColory = 0;
                if (j % 2 == 0) {
                    fieldHexagon[i][j] =
                            new Hex(i * Hex.getWidth(),
                                    j * Hex.getHeight() * 3 / 4);
                    x = i * Hex.getWidth();
                    y = j * Hex.getHeight() * 3 / 4;
                } else {
                    fieldHexagon[i][j] =
                            new Hex(i * Hex.getWidth() - Hex.getWidth() / 2,
                                    j * Hex.getHeight() * 3 / 4);
                    x = i * Hex.getWidth() - Hex.getWidth() / 2;
                    y = j * Hex.getHeight() * 3 / 4;
                }
//
                if (x <= dX && y <= dY) {
                    OFij = y * dX - x * dY;

//
                    OF1 = (y + 0.5 * Hex.getWidth()) * dX - (x) * dY;
                    OF2 = (y + 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
                    OF3 = (y - 0.25 * Hex.getWidth()) * dX - (x + 0.5 * Hex.getHeight()) * dY;
                    OF4 = (y - 0.5 * Hex.getWidth()) * dX - (x) * dY;
                    OF5 = (y - 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
                    OF6 = (y + 0.25 * Hex.getWidth()) * dX - (x - 0.5 * Hex.getHeight()) * dY;
```

```
            OF7 = (y + 0.375 * Hex.getWidth()) * dX - (x + 0.25 * Hex.getHeight()) *
dY;
            OF8 = (y) * dX - (x + 0.5 * Hex.getHeight()) * dY;
            OF9 = (y - 0.375 * Hex.getWidth()) * dX - (x + 0.25 * Hex.getHeight()) *
dY;
            OF10 = (y - 0.375 * Hex.getWidth()) * dX - (x - 0.25 * Hex.getHeight()) *
dY;
            OF11 = (y) * dX - (x - 0.5 * Hex.getHeight()) * dY;
            OF12 = (y + 0.375 * Hex.getWidth()) * dX - (x - 0.25 * Hex.getHeight()) *
dY;

            //OF1 = (y + 0.5) * dX  - (x - 0.25) * dY;
            //OF1 = OFij +( (2) * dX + dY)/4;
            //OF2 = OFij +( (2) * dX - dY)/4;
            //OF3 = OFij -dY/2;
            //OF4 = OFij -( (2) * dX - dY)/4;
            //OF5 = OFij -( (2) * dX + dY)/4;
            //OF6 = OFij +dY/2;
            //OF7 = OFij;
            //boolean boundsPixel = false;
            intensivtistColory = 0;
            if (OF1 < 0) {
               intensivtistColory++;
            }
            if (OF2 < 0) {
               intensivtistColory++;
            }
            if (OF3 < 0) {
               intensivtistColory++;
            }
```

```
if (OF4 < 0) {
   intensivtistColory++;
}
if (OF5 < 0) {
   intensivtistColory++;
}
if (OF6 < 0) {
   intensivtistColory++;
}
if (OF7 < 0) {
   intensivtistColory++;
}
if (OF8 < 0) {
   intensivtistColory++;
}
if (OF9 < 0) {
   intensivtistColory++;
}
if (OF10 < 0) {
   intensivtistColory++;
}
if (OF11 < 0) {
   intensivtistColory++;
}
if (OF12 < 0) {
   intensivtistColory++;
}
```

```
double coefColor = 0.0;
double p = 3.14159265359;
switch (intensivtistColory) {
    case 0:
        coefColor = 0.0;
        break;
    case 1:
        coefColor = 0.0905934/p;
        break;
    case 2:
        coefColor = 0.3802725/p;
        break;
    case 3:
        coefColor = 0.56586498/p;
        break;
    case 4:
        coefColor = 0.75145745/p;
        break;
    case 5:
        coefColor = 1.39672605/p;
        break;
    case 6:
        coefColor = 1.57/p;
        break;
    case 7:
        coefColor = 1.9147276/p;
        break;
    case 8:
        coefColor = 2.2396469/p;
        break;
```

```
                    case 9:
                        coefColor = 2.5258007/p;
                        break;
                    case 10:
                        coefColor = 2.9342628/p;
                        break;
 case 11:
                        coefColor = 2.99897295/p;
                        break;
                    case 12:
                        coefColor = 3.0494066/p;
                        break;
                    default:
                        coefColor = 0.0;
                        break;
                }
//
                Hex.setFilledColor(Color.hsb(filledColor.getHue(), coefColor, 1.0));
                //if (coefColor != 1.0) {
                    fieldHexagon[i][j].filled = true;
                //}
            }
            fieldHexagon[i][j].print();
            Hex.setFilledColor(filledColor);
            //System.out.println(i+" "+j );
        }
    }
  }

  @Override
```

```
    public void initialize(URL url, ResourceBundle resourceBundle) {
        ObservableList<String> langs = FXCollections.observableArrayList("Лінія",
"Коло", "Антиаліайзінг 7 точок", "Антиаліайзінг 12 точок", "Коло Pro");
        chooser.setItems(langs);
        chooser.setValue("Лінія");

        chooser.setOnAction(e->{
            System.out.println("Обрана розрядність: " + chooser.getValue());

        });
        vBoxRadius.setDisable(true);
        //refreshButton.fire();
    }
}
```

ДОДАТОК Е.
Лістинг коду
програмної
компоненти

```
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.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()

haxaScrollPane.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();
            haxaScrollPane.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/REA
DME.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 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();
    }
}
```

ДОДАТОК Є

ІЛЮСТРАТИВНА ЧАСТИНА

МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ ФОРМУВАННЯ ГРАФІЧНИХ ЗОБРАЖЕНЬ НА ГЕКСОГОНАЛЬНОМУ РАСТР

(Назва магістерської кваліфікаційної роботи)

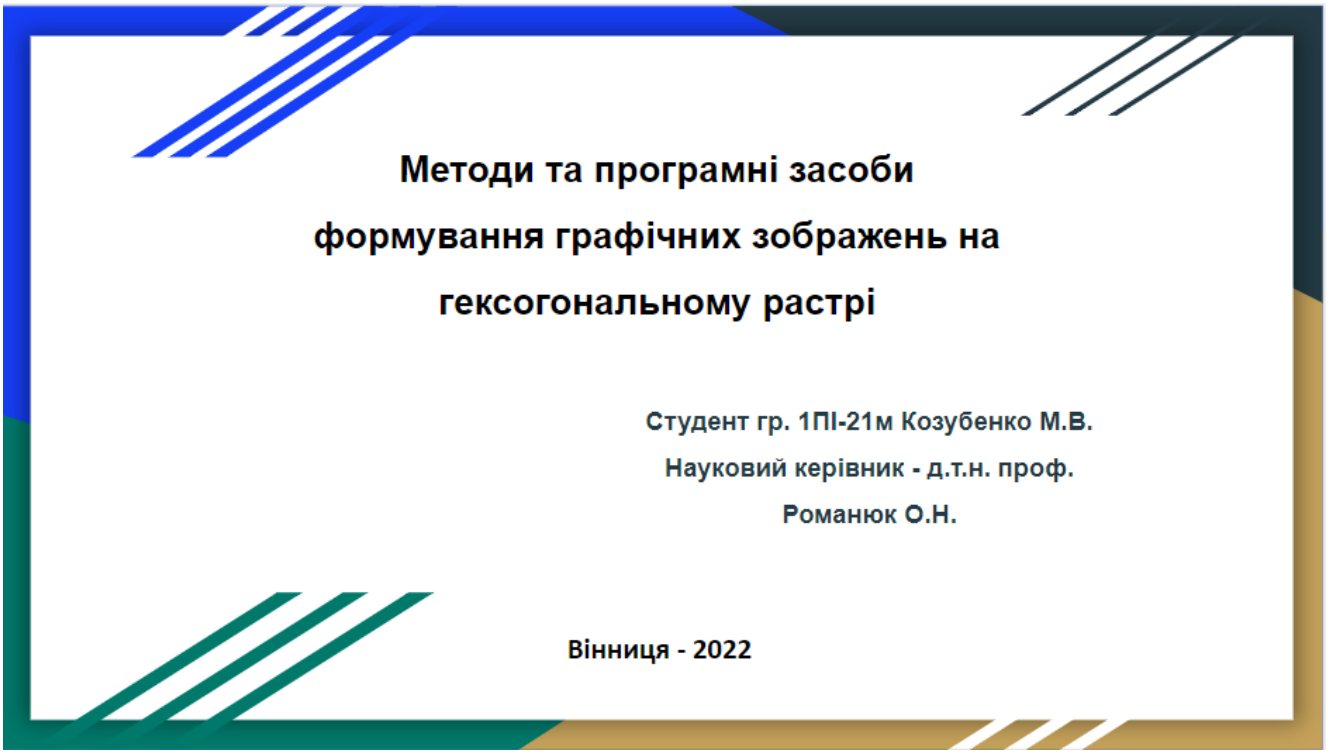


Рисунок Є.1 - Слайд презентації №1

Рисунок Є.2 - Слайд презентації №2

Мета і завдання дослідження

Мета та завдання дослідження. Метою роботи є підвищення реалістичності формування графічних зображень за рахунок використання гексагонального растру.

Предмет дослідження – методи та засоби формування графічних примітивів на гексагональному растрі

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

Основними задачами дослідження є:

- провести аналіз галузей використання гексагонального растру;
- запропонувати нові: методи формування примітивів на гексагональному растрі;
- метод підвищення продуктивності формування вектора;
- розробити програмні компоненти для систем візуалізації на основі запропонованих методів;
- провести експериментальні дослідження розроблених засобів текстурування.

Рисунок Є.3 - Слайд презентації №3

Екран на гексагональному растрі

Даний екран має підвищену роздільну здатність завдяки гексагональному растру.

Рисунок Є.4 - Слайд презентації №4

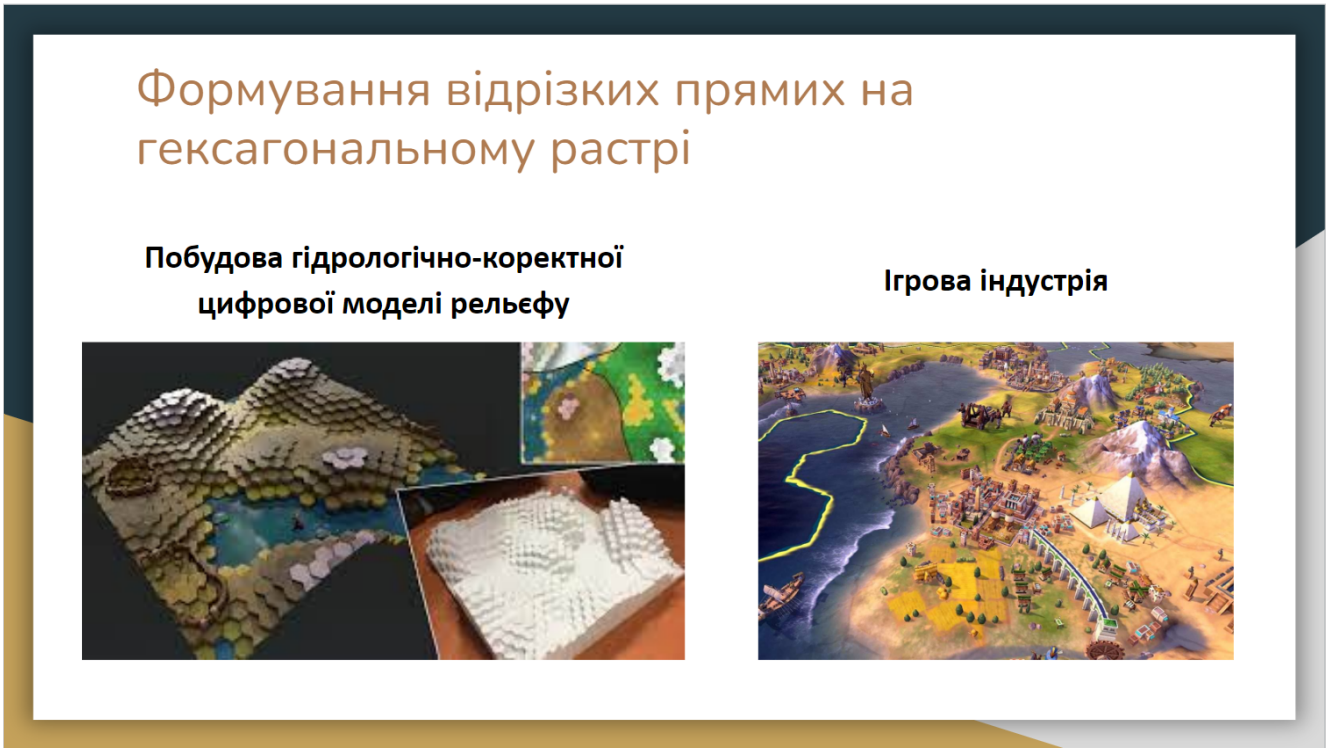


Рисунок Є.5 - Слайд презентації №5

Рисунок Є.6 - Слайд презентації №6

Формування відрізких прямих на гексагональному растрі

Метод оцінювально функції для побудови вектора на гексагональному растрі.

$$OF_i = \frac{y_i}{x_i} - \frac{\Delta y}{\Delta x} = \frac{y_i \Delta x - x_i \Delta y}{x_i \Delta x}$$

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

Завдяки методу оцінювальної функції, можливо визначити наступний крок, наступний гекс, який буде замальований.

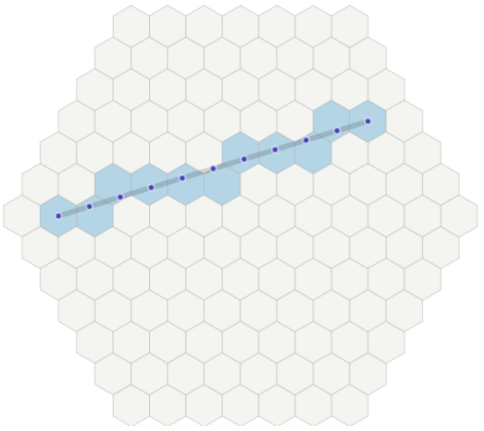


Рисунок Є.7 - Слайд презентації №7

Блок-схема формування відрізких прямих на гексагональному растрі

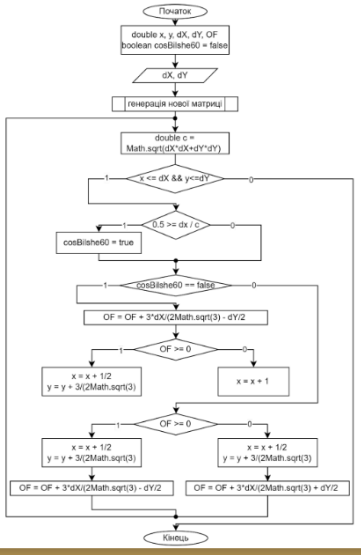
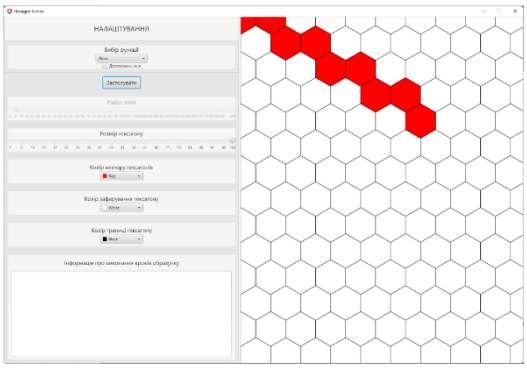
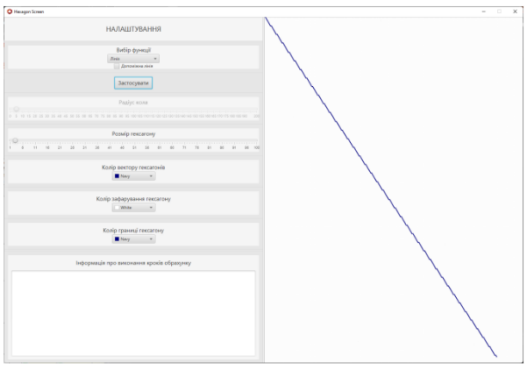


Рисунок Є.8 - Слайд презентації №8

Приклад формування відрізка прямої на гексагональному растрі



Формування відрізка



Формування відрізка зі розміром гексагону в 3 піксела

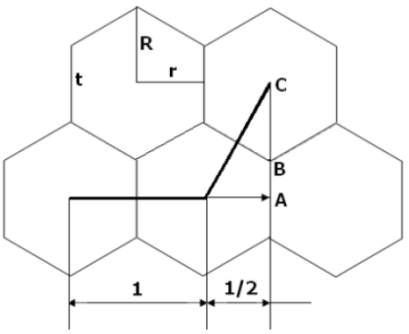
Рисунок Є.9 - Слайд презентації №9

Формування відрізких прямих на гексагональному растрі комбінованими приростами

Формування відрізка, комбінованого приросту, який включає горизонтальний та діагональний крок.

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

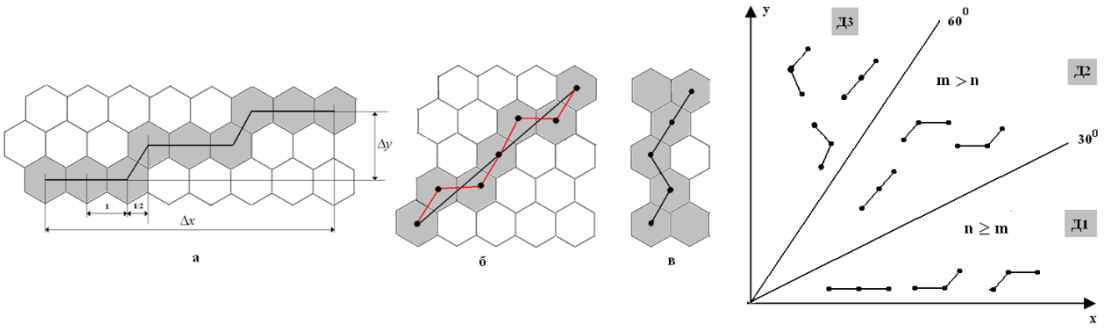
$$OF_{i+1} = y_i * \Delta x - (x_i + 1) * \Delta y = y_i * \Delta x - x_i * \Delta y - \Delta y = OF_i - \Delta y$$



Відстань між гексами

Рисунок Є.10 - Слайд презентації №10

Формування відрізких прямих на гексагональному растрі комбінованими приростами. Типи допустимих сполучень.



Формування векторів у різних секторах

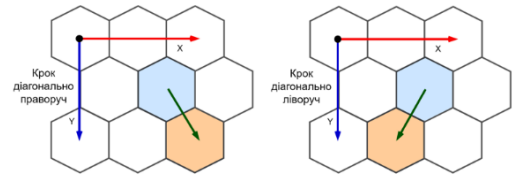
Рисунок Є.11 - Слайд презентації №11

Формування кола на гексагональному растрі. Зсув по гексах

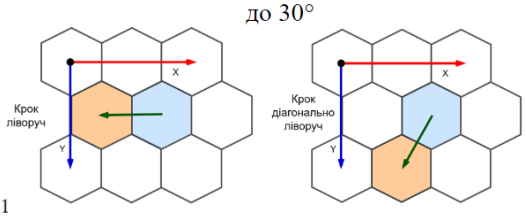
На кожному кроці інтерполяції, доцільно вибирати той піксел, для якого є мінімальним значення квадрату відстані між пікселем і точкою на колі. Залежно від результату порівняння оцінювальних функцій обирається такий крок інтерполяції, який відповідає найменшому значенню оцінювальних функцій.

$$OF_i = \left(x_i^2 + y_i^2\right) - R^2$$

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



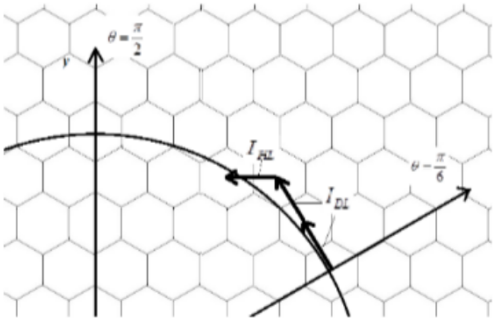
Можливі кроки при формуванні сектора від 0° до 30°



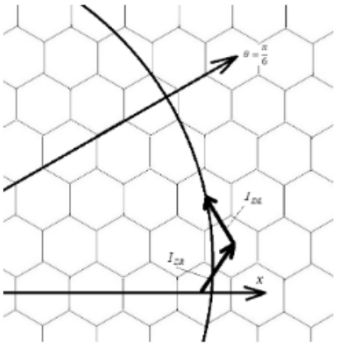
Можливі кроки при формуванні сектора від 30° до 90°

Рисунок Є.12 - Слайд презентації №12

Кругова інтерполяція на гексагональному растрі



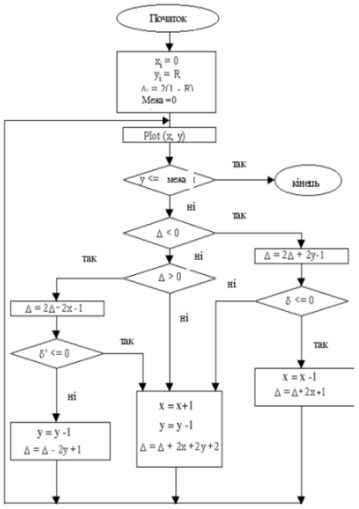
Інтерполяція дуги кола в секторі від 30° до 90°



Інтерполяція дуги кола в секторі від 0° до 30°

Рисунок Є.13 - Слайд презентації №13

Блок-схема формування кола



Блок-схема алгоритму формування кола

Рисунок Є.14 - Слайд презентації №14

Рисунок Є.15 - Слайд презентації №15

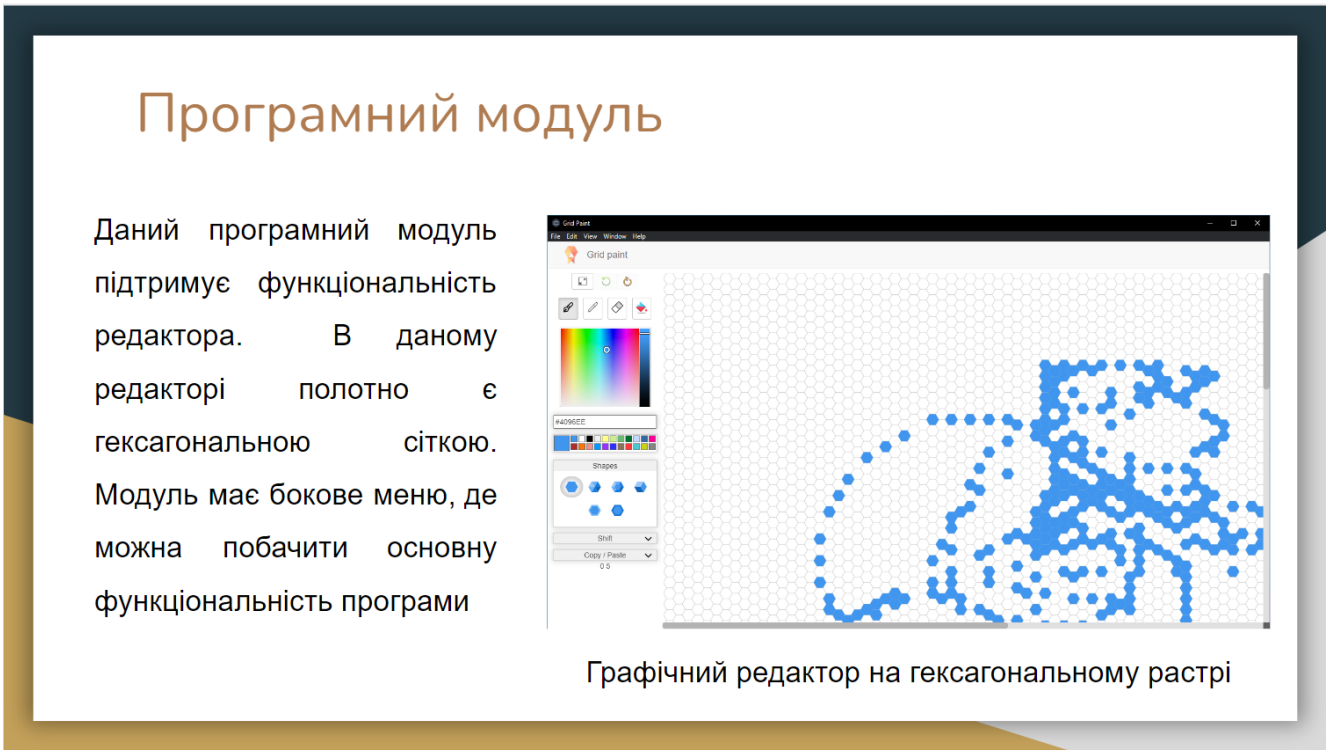


Рисунок Є.16 - Слайд презентації №16

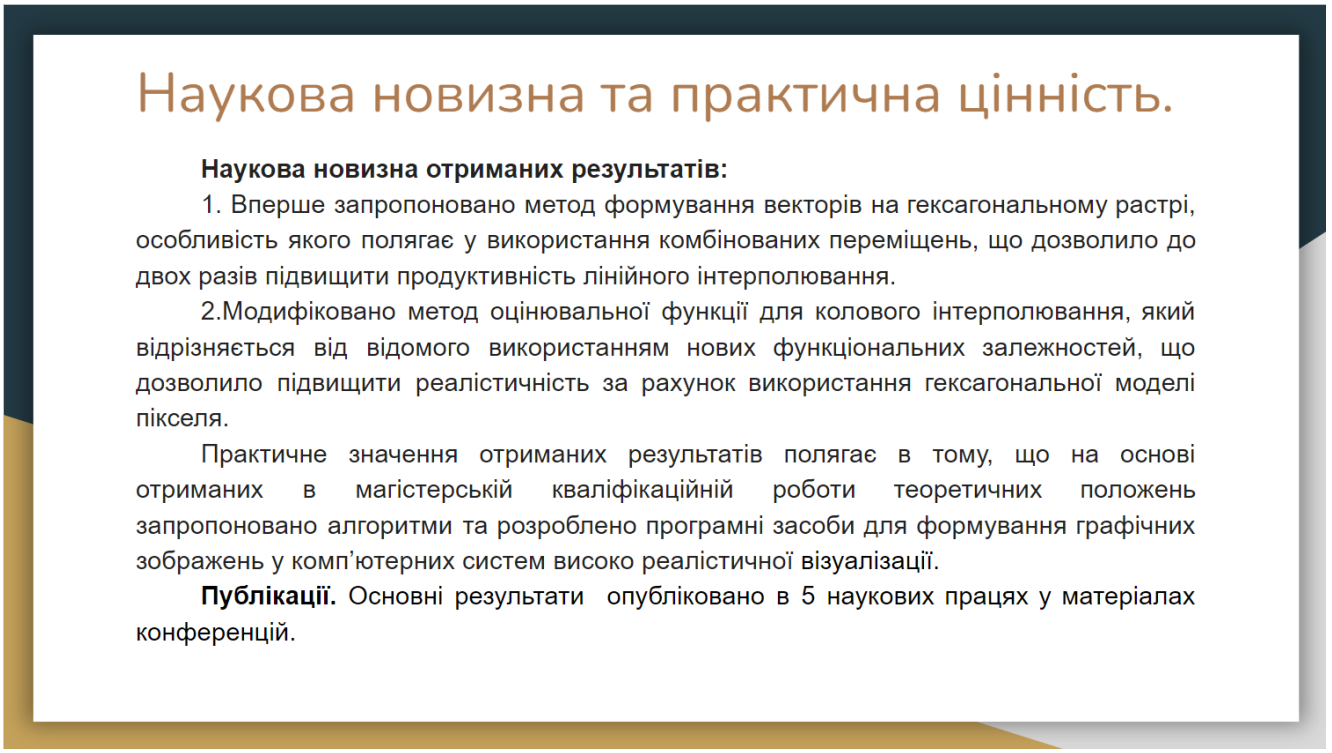


Рисунок Є.17 - Слайд презентації №17

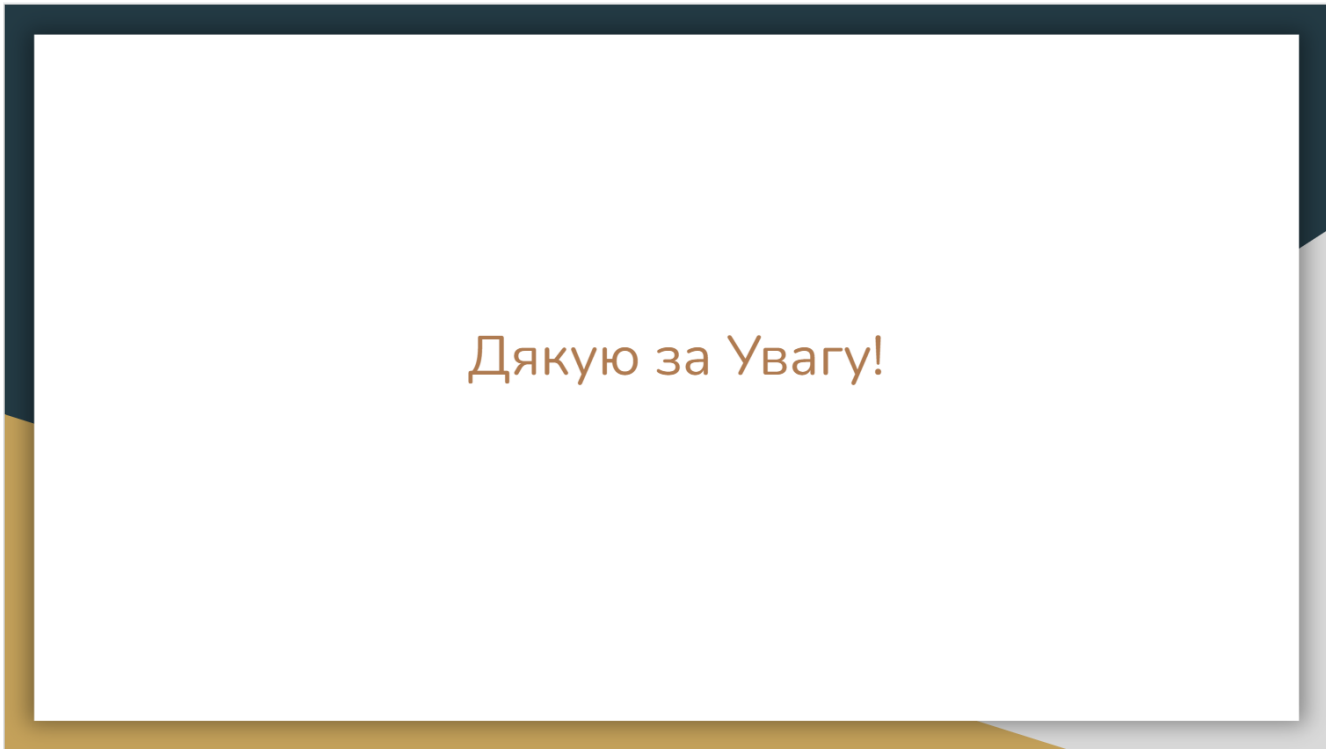


Рисунок Є.18 - Слайд презентації №18

ВІДГУК НАУКОВОГО КЕРІВНИКА
на магістерську кваліфікаційну роботу студента гр. 1ПІ-21м
Козубенка Максима Володимировича
«Методи та програмні засоби формування графічних зображень на гексогональному растрі»

Дослідники все частіше звертають увагу на переваги застосування гексагонального растру при формуванні та відтворенні зображень. Такі переваги в багатьох випадках дозволяють підвищувати реалістичність формування графічних зображень. Переваги обумовлені здатністю гексагона замощувати площину екрану без розривів і накладань, а, також, геометричними особливостями гексагона, такими як рефлекційна симетрія, шестизв'язність гексагонального растру У зв'язку з цим тема магістерської кваліфікаційної роботи роботи Козубенка Максима Володимировича, в якій розглядаються питання підвищення реалістичності відтворення графічних примітивів на гексагональному растрі, є актуальною.

У роботі проведено аналіз галузей використання гексагонального растру, його особливостей.

Модифіковано метод оцінювальної функції для формування відрізків прямих і кіл на гексагональному растрі. Запропоновано метод підвищення продуктивності формування векторів подвійними комбінованими приростами. Це дозволило підвищити продуктивність лінійної інтерполяції до двох разів.
Розроблено графічний редактор з широкими функціональними можливостями.

Поставлені в роботі завдання виконані якісно та в повному обсязі. Пояснювальну записку написано технічно грамотно. Всі прийняті рішення обґрунтовано та експериментально перевірено. Графічна частина виконана відповідно до встановлених вимог.

За тематикою МКР опубліковано 5 наукових праць.. Робота має достатню апробацію.

Вважаю, що магістерську кваліфікаційну роботу виконано на високому науково-технічному рівні, з дотриманням встановлених вимог. Робота заслуговує оцінки «А», кількість балів – 90 а її автор Козубенко М.В. – присвоєння ступеня вищої освіти магістр, спеціальність 121 – «Інженерія програмного забезпечення», освітня програма «Інженерія програмного забезпечення».

Науковий керівник
д.т.н., проф. кафедри ПЗ Романюк О.Н..

ВІДГУК ОПОНЕНТА
на магістерську кваліфікаційну роботу студента гр. 1ПІ-21м
Козубенка Максима Володимировича
«Методи та програмні засоби формування графічних зображень на гексогональному растрі»

Роль комп'ютерної графіки, як однієї з основних забезпечуючих підсистем обчислювальної техніки постійно зростає, оскільки вона дозволяє в умовах сучасного рівня розвитку комп'ютерної техніки реалізувати найбільш прийнятну й звичну для користувача технологію подання інформації.

Основною задачею комп'ютерної графіки є синтез зображень з високим ступенем реалістичності. Для формування таких зображень почали використовувати гексагональний растр, який забезпечує кращу згладженість контурних зображень та підвищення розподільну здатність пристроїв відображення. Тому тема магістерської кваліфікаційної роботи Козубенко М.В., яку присвячено питанням формування графічних примітивів на гексагональному растрі, є актуальною.

В МКР проведено детальний аналіз предметної галузі. Розглянуто особливості гексагонального растру та галузі його ефективного використання.

Модифіковано метод оцінювальної функції для колового інтерполювання, який відрізняється від відомого використанням нових функціональних залежностей, що дозволило підвищити реалістичність формування графічних зображень.. Запропоновано метод формування векторів на гексагональному растрі, особливість якого полягає у використання комбінованих переміщень, що дозволило до двох разів підвищити продуктивність лінійного інтерполювання.

Запропоновано алгоритми та розроблено програмні засоби для формування графічних зображень у комп'ютерних систем високо реалістичної візуалізації

За тематикою дослідження опубліковано 5 наукових праць. Результати роботи доповідалися на 5 Міжнародних науково - технічних конференціях.

Зауваження до магістерської кваліфікаційної роботи: не зрозуміло, чи можливо використовувати запропоновані методи для формування інших примітивів, зокрема, кривих.

Зазначене зауваження не зменшує цінність роботи та не стосуються основних її положень.

Вважаю, що магістерська кваліфікаційна робота виконана відповідно до завдання із дотриманням встановлених вимог. Робота заслуговує оцінки «А», кількість балів - 90 а її автор Козубенко М.В. – присвоєння ступеня вищої освіти магістр, спеціальність 121 – «Інженерія програмного забезпечення», освітня програма «Інженерія програмного забезпечення».

Опонент
к.т.н., доц. кафедри ЗІ Дудатьєв А. В.