Довідка
про відсутність плагіату у випускній кваліфікаційній роботі

Міністерство освіти і науки України
Український державний університет науки та технологій

Кафедра «Комп'ютерні інформаційні технології»

ДОВІДКА

За результатами перевірки випускної кваліфікаційної роботи здобувача вищої освіти Мосієнка Владислава Сергійовича
(прізвище, ім'я, по батькові)
на тему: Конструктивні просторові перетворення двовимірних фракталів
в роботі не виявлено порушень академічної доброчесності.

Керівник ВКР _____ Шинкаренко В.І.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Український державний університет науки і технологій

Кафедра Комп'ютерні інформаційні технології

«ДО ЗАХИСТУ»

Завідувач кафедри

_____ /Вадим ГОРЯЧКІН/

«17» 12 2021 р.

ДИПЛОМНА РОБОТА

на здобуття освітнього ступеня «магістр»

Галузь знань **12 Інформаційні технології**

Спеціальність **121 Інженерія програмного забезпечення**

Тема **Конструктивні просторові перетворення двовимірних фракталів**

Theme **Constructive spatial transformations of two-dimensional fractals**

Керівник дипломної роботи проф. _____ Віктор ШИНКАРЕНКО

Нормоконтролер доц. _____ Олена КУРОП'ЯТНИК

Студент групи ПЗ2021 _____ Владислав МОСІЄНКО

Student _____ Vladislav MOSIENKO

Дніпро – 2021

Український державний університет науки і технологій

Факультет Комп'ютерних технологій та систем кафедра Комп'ютерні інформаційні технології

Спеціальність Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____ В.І. ШИНКАРЕНКО
(підпис)
«___»_____2021р.

ЗАВДАННЯ

до дипломної роботи на здобуття ОС ___магістр_____
(освітній ступінь)

студента групи ПЗ2021 (951м) Мосієнка Владислава Сергійовича
(номер групи) (ПІБ)

1 Тема дипломної роботи: Конструктивні просторові перетворення двовимірних фракталів
затверджена наказом по університету від «18» листопада 2020 р. № 690 ст.

2 Термін подання студентом закінченої роботи «10» грудня 2021р.

3 Вихідні дані до дипломної роботи _____

_____

_____

4 Зміст пояснювальної записки (перелік питань до розробки) проведення дослідження можливостей та ефективності застосування конструктивно-продукційного підходу для побудови двовимірних фракталів у просторі.

5 Перелік демонстраційного матеріалу презентація на тему використання методу конструктивно-продукційних перетворень двовимірних фракталів, результати експериментів; відео демонстрації роботи розробленої системи для досліджень.

6. Консультанти (з назвами розділів):

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Техніко-економічні розрахунки	доц. Гненний М.В.	18.10.21	24.10.21
Охорона праці	проф. Саблін О.І.	18.10.21	24.10.21

КАЛЕНДАРНИЙ ПЛАН

№ пор.	Назва розділів дипломної роботи	Термін виконання розділів роботи	Примітка
1	Вступ		
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами		
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження		
4	Постановка задачі, технічне завдання	11.10.21 – 17.10.21	30%
5	Техніко-економічні показники		
6	Розробка інструментальних засобів дослідження		
7	Виконання досліджень	08.11.21 – 14.11.21	60%
8	Оформлення тез доповідей		
9	Оформлення статті у фаховий журнал		
10	Оформлення пояснювальної записки		
11	Розробка демонстраційних матеріалів	29.11.21 – 05.12.21	100%

Дата видачі завдання «18» 11 2021 р.

Керівник дипломної роботи _____ Віктор Гненний
(підпис) (ПІБ)

Завдання прийняв до виконання _____ Владислав Моісієнко
(підпис) (ПІБ)

РЕФЕРАТ

Об’єктом дослідження є процеси конструктивних просторових перетворень двовимірних фракталів. Конструктори, як і формальні граматики, є інструментом програмної інженерії.

Предметом дослідження є дослідження конструктивно-продукційного підходу для побудови двовимірних фракталів у просторі.

Метою дослідження є розробка методу та засобів конструктивного формування двовимірних фракталів.

Результати та їх новизна: побудова графічного відображення сконструйованих фігур, в тому числі фракталів.

Пояснювальна записка складається зі вступу, п’яти розділів, висновків, бібліографічного списку та чотирьох додатків.

Вступ описує суть, актуальність дослідження, об’єкт та предмет дослідження, мету та завдання досліджень.

Перший розділ описує проблематику, огляд аналогічних програмних додатків та визначення інструментів для розробки.

Другий розділ описує обґрунтування напрямку досліджень.

В третьому розділі викладено процес проектування і розробки ПЗ.

Четвертий розділ дослідження процесу просторових перетворень з використанням конструктивно-просторового підходу.

П’ятий розділ розкриті питання охорони та безпеки праці в надзвичайних ситуаціях.

Додатки технічне завдання, робочий проект, тези, стаття.

ЗМІСТ

ВСТУП

Самоподібність давно привернула увагу науковців і у зв’язку с цим з’явилася потреба в відображенні цього математичного означення графічно, для кращого розуміння того, що описує це поняття.

Дослідження фракталів розпочалося у 1975 році. Фактично ми тільки розпочали вивчення цієї величезної та незвіданої території. Фрактали виходять за рамки чистої математики, мистецтва, схожого з музикою та поезією, або практичного інструменту вирішення прикладних завдань. Вони можуть дати набагато більше: наприклад, пояснити явища, що знаходяться поза нашим розумінням при поточному розвитку науки. Вся фрактальна космологія будується на теорії нескінченності простору Всесвіту та розподілі у ньому астрономічних об'єктів за принципом фрактальної розмірності (в діапазоні від 2 до 3).

Нове геометричне поняття «фрактал» було введено Бенда Б. Мандельбротом для опису об'єктів та явищ, які не мають певного лінійного розміру. Характерною особливістю фрактальної освіти є те, що його структура виявляється лише при спільному поділі кількох рівнів, різниця масштабів яких ускладнює наочне уявлення цієї структури та послідовне опис багатьох масштабних структур, безпосередні спостереження яких утруднено, може бути досягнуто в рамках фрактальної геометрії.

Актуальність роботи. В даний час знайшло широке застосування фракталів у різних розділах математики, фізики та астрофізики до опису таких об'єктів і явищ, які у своєму втіленні спираються на математично визначений дробовий показник геометричного ступеня фрактальних ефектів, що описуються. Крім того, ці фрактальні об'єкти характеризуються ієрархічною супідрядністю, самоподібністю та масштабною інваріативністю. Наприклад, для поверхні Місяця характерна самоподібність поверхні та контурів кратерів тож застосування методів фрактальної геометрії для опису поверхні Місяця дозволило б надати нові можливості для її аналізу.

У цій роботі наведено вступ до фракталів, фрактальної розмірності та фрактальної геометрії. Для ілюстрації самоподібних геометричних фігур описано багато фракталів.

Далі слідує відкриття динамічних систем і ітераційних відображень з чудовими прикладами фракталів Джулії та фракталів Мандельброта. Фрактали, хаос, біфуркації та розмірність Хаусдорфа здаються істотними елементами фрактальної геометрії.

Об'єкт дослідження. Процеси конструктивних просторових перетворень двовимірних фракталів.

Предмет дослідження. Предметом є можливості використання конструктивно-продукційного підходу задля побудови двовимірних фракталів у просторі, методи формування двовимірних фракталів, їх перетворення.

Мета дослідження. Метою дослідження є розробка методів та засобів конструктивного формування двовимірних фракталів.

Наукова новизна. Побудова двовимірних фракталів у тривимірному просторі з використанням конструктивно-продукційного підходу. За допомогою конструктивно-продукційного моделювання можливе досягнення побудови та відображення двовимірних фракталів у просторі, що розширює можливості для більш детального дослідження фракталів та експериментів з ними для наукових працівників.

Апробація результатів дослідження та публікації. Результати магістерської роботи доповідались на семінарі кафедри КІТ 22.02.2021 р., представлено всеукраїнських науково-практичних конференціях: Науково-технічний прогрес на транспорті (29 березня 2021 р.) та Наука і сталий розвиток транспорту (28 жовтня 2021 року).

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ПОБУДОВИ ФРАКТАЛІВ У ПРОСТОРІ

Латинське слово fractus означає "складений із фрагментів". Фрактал можна визначити як об'єкт досить складної форми, що утворюється в результаті виконання простого ітераційного циклу. Ітераційність, рекурсивність зумовлюють такі властивості фракталів, як самоподібність – окремі фрагменти фракталу схожі формою весь фрактал загалом.

У 1975 році французький математик Бенуа Мандельброт видав книгу "The fractal Geometry of Nature".

"Чому геометрію часто називають холодною та сухою? Одна з причин полягає в її нездатності описати форму хмари, гори, дерева або берега моря. Хмари – це не сфери, гори – не конуси, лінії берега – не кола, і кора не є гладкою, і блискавка не поширюється прямою… Природа демонструє нам не просто більш високий ступінь, а зовсім інший рівень складності. Математики… віддали перевагу все більше і більше віддалятися від природи, винаходячи теорії, які не відповідають нічому з того, що можна побачити чи відчути…" (Б. Мандельброт).

Мандельброт запропонував спочатку визначення фракталу в такому вигляді: "Фракталом називається безліч, розмірність Хаусдорфа-Безиковича якого строго більша за його топологічну розмірність". Потім він сформулював простіший вислів: "Фракталом називається структура, що складається з частин, які в якомусь сенсі подібні до цілого". Так що суворого та повного визначення фракталів досі не існує.

Виділяють кілька різновидів фракталів: алгоритмічні, геометричні та фрактали на основі методу IFS.

Розглянемо алгоритмічні фрактали. Суть алгоритму зводиться до ітераційного висновку точок, які у коло збіжності кожної початкової точки, не більше заданого графічного вікна. Для кожного циклу ітерації (кожної початкової точки) вибирається свій колір.

Якщо прийняти значення k для кожної початкової точки (x,y) як висот певної поверхні в даній точці можна побудувати об'ємне зображення або його частини, яке

при спеціально підібраному освітленні може виглядати і як скеля з плоскою вершиною, і як водоспад, і як гірська печера.

Типовий фрактал, є ієрархічний об'єкт, що складається з батьківського тіла у формі кардіоїди і численних нащадків, що повторюють форму предків, від яких вони відгалужуються. Сама подібність елементів фракталу добре видно на його зображенні.

Зв'язок безлічі означає, що його елементи, навіть найменшого розміру, не відокремлені, а з'єднані найтоншими нитками в одне ціле. Продовжуючи процес збільшення прикордонних областей, ми всюди бачимо нескінченну різноманітність форм, що вражають гармонією, пишнотою і дивовижною схожістю із зображеннями регулярно-хаотичних явищ природи: блискавок, сніжинок, крижаного візерунка, інею на гілках дерев, коралів, павутини, сонячних протубера. т.п.

Розглянемо геометричні фрактали. Геометричними названі фрактал, форма яких може бути описана як послідовність простих геометричних операцій. Геометричні фрактали будуються на основі вихідної фігури – лінії, полігону, поліедра – шляхом її дроблення та виконання афінних та логічних перетворень (об'єднання чи виключення) отриманих фрагментів.

Метод побудови фрактального об'єкта може бути як ітераційним, так і рекурсивним, причому рекурсивний частіше буває багатшим можливостями і простіше в програмуванні. Наприклад, крива Кох стає фракталом в результаті нескінченної кількості ітерацій, в ході яких виконується розподіл кожного відрізка прямої на три частини. На середній частині відрізка будується правильний трикутник, потім процес повторюється кожному з отриманих відрізків, крім середнього.

Алгоритм заснований на рекурсивному виклик процедури для кожного з отриманих відрізків. Аналогічний алгоритм використовується під час побудови фрактального трикутника.

Взагалі алгоритм побудови кривої Кох можна застосувати до будь-якого багатокутника або ламаної лінії, запустивши алгоритм побудови кривої Кох для кожного зі складових фігур.

Розглянемо плоскі фрактали. Будівельними елементами плоскі геометричних фракталів є плоскі полігони, найчастіше три- і чотирикутники.

Найвідомішим фракталом цього класу є трикутник Серпінського, що будується шляхом розбиття трикутника – не обов'язково рівностороннього – середніми лініями на чотири подібні трикутники, виключення центрального та рекурсивного розбиття кутових трикутників до отримання майданних елементів бажаного або мінімально видимого дозволу.

Фрактали на основі методу IFS

Цю групу складають фрактали, які генеруються згідно з методом "систем ітеративних функцій" – IFS (Iterated Functions Systems).

Даний метод може бути описаний як послідовний ітеративний розрахунок координат нових точок у просторі:

$$\begin{cases} \mathbf{x_{k+1}} = \mathbf{F_x}(\mathbf{x_k}, \mathbf{y_k}) \\ \mathbf{y_{k+1}} = \mathbf{F_y}(\mathbf{x_k}, \mathbf{y_k}) \end{cases},$$

де $\mathbf{F_x}$ та $\mathbf{F_y}$ – функції перетворення координат, наприклад афінного перетворення. Ці функції визначають форму фрактала. У разі афінного перетворення необхідно знайти відповідні числові значення коефіцієнтів. Прикладом такого фракталу є побудова зображення папоротевого аркуша.

Метод IFS застосовується не тільки для створення зображень. Барнслі та Слоан використовували його для ефективного стиснення графічних зображень під час запису у файли. Основна ідея така: оскільки фрактали можуть представляти дуже складні зображення за допомогою простих ітерацій, опис цих ітерацій вимагає значно меншого обсягу інформації, ніж відповідні растрові зображення. При кодуванні зображення необхідно вирішувати обернену задачу: для зображення (або його фрагмента) підібрати відповідні коефіцієнти афінного перетворення. Цей метод

використовується для запису кольорових фотографій у файли зі стисненням у десятки та сотні разів без помітного погіршення зображення. Формат таких графічних файлів названий FIF (Fractal Image Format) та запатентований фірмою Iterated Sistems.

Фрактальний підхід знайшов стала вельми поширеною у багатьох галузях комп'ютерної графіки, мистецтва та науки. Так, з'явилася теорія фрактальних тріщин, фрактальна механіка деревополімерних композитів та інші галузі.

У задачах КГ фрактальна технологія набула найбільшого поширення при формуванні об'єктів природного ландшафту: лінії горизонту, нерівних поверхонь, пагорбів, гір, каньйонів та інших нерегулярних утворень. Побудова заснована на рекурсивному розбиття вихідного об'єкта середніми точками та зміщення цих точок за методом керованої випадковості. Початкові об'єкти вибираються із простих геометричних фігур: відрізків, трикутників, прямокутників, тетраедрів.

Фрактали широко застосовують у растрових редакторах (Adobe Photoshop, Corel Painter), у векторній (Corel Draw, Adobe Illustrator) та тривимірній (Corel Bryce, 3Dmax) графіку.

1.1. Призначення та область застосування

Області в яких застосування фракталів найбільш поширене:

- комп'ютерна графіка;
- фізика та інші природничі науки;
- стиснення зображень;
- аналіз ринків;
- інші застосування.

Розглянемо фрактали у містах. Деякі міста мають тенденцію рости у вигляді фрактальних моделей з часом і називаються фрактальними містами. Оскільки велике фрактальне місто поглинає сусідні міста та села, розроблений шаблон нагадує самоподібну структуру, яка спочатку здається випадковою, але є динамічною мережею, яка може виявитися більш ефективною, ніж сучасні «заплановані» міста.

Фрактали у медицині. Знання фракталів особливо корисні в медичній діагностиці, в тому числі раку. Оскільки здорові клітини кровоносних судин людини зазвичай ростуть упорядковано фрактально, ракові клітини, які ростуть аномальним чином, стає легше виявити. Ця форма фрактального аналізу значно полегшує розрізнення здорових клітин і ознак занепокоєння.

Стиснення і роздільна здатність зображення. Оскільки фрактали дозволяють нам передавати, здавалося б, випадкові шаблони з невеликою кількістю даних, робота з роздільною здатністю зображення і навіть створення 3D-моделей стає надзвичайно ефективною для даних за допомогою кодування фрактального зображення (FIC) та інших додатків.

Розглянемо використання фракталів в антенах. Самоподібна природа фракталів також корисна при створенні та експлуатації антен. Такі криві, як крива Гільберта, можна використовувати для проектування високопродуктивних і низькопрофільних антен. Якщо вони поєднані з концепціями електромагнітного випромінювання, багатодіапазонні антени також можуть працювати.

Розглянемо використання фракталів в мистецтві. Від спрощеного до складного діапазону правил, які керують створенням фракталів, митців приваблює. Наприклад, набір Мандельброта добре відомий тим, що надає різні «сцени» на основі колірної схеми, яка використовується для його відображення.

Розглянемо використання фракталів при аналізі ринків. Останнім часом фрактали стали найпопулярнішим інструментом у трейдерів для аналізу стану біржових ринків. Фрактали ринку є одним із індикаторів у торговельній системі Біла Вільямса. Вважається, що він же вперше і ввів цю назву у трейдинг. При торгівлі фракталами, у поєднанні зі своїм індикатором Алігатор, автор виявляв локальні максимуми або мінімуми ринку.

Теорія фракталів на ринку свого часу викликала бурхливі суперечки, перш за все, тому що автор, як багато хто вважає, вставив у свою теорію багато наукової термінології (фрактал, атрактор і т.д.), і зробив це не зовсім коректно. Оскільки фрактали Вільямса з'являються на ринку досить часто і практично на всіх часових

таймфреймах і є, по суті, простими локальними екстремумами на відрізку з 5 барів і практично не відповідають математичній теорії фракталів. Такою самою освітою на графіку є і ТД-точки другого порядку Томаса Демарка. Однак, незважаючи на всі ці збіги ця теорія дуже популярна і досі.

1.2. Формулювання задачі та складових

Необхідно спроектувати та розробити програмний продукт, за допомогою якого можна наочно подивитися зображення фрактальної графіки. Програма повинна дозволяти розкрити сутність фракталу – багаторазове самоповторення (всього зображення чи певної його частини). Інтерфейс має бути максимально зрозумілим. Швидкість роботи мусить бути такою, щоб збалансувати продуктивність і якість, тобто. при цій швидкості промальовується досить наочне зображення. Необхідна також можливість збереження фрактального зображення. Програма має бути інтуїтивно зрозумілою та "не відштовхувати при першому погляді". Можливостями програми мають бути доступні промальовування щонайменше десяти алгебраїчних і щонайменше двох геометричних фракталів.

Дослідити структуру та особливості фракталів на основі побудови різних видів фракталів (наприклад крива Госпера, сніжинка Коха та інші) використовуючи JavaScript.

1.3. Аналіз існуючих програмних аналогів та формулювання сильних та слабких сторін

Розглянемо аналог Ultra Fractal. Ultra Fractal багатофункціональна фрактальна програма (рис. 1.1). Особливістю, яка виділила цю програму на початку, була можливість створювати шари в зображенні, що раніше було лише функцією графічної програми. Ця програма тепер має можливість створювати анімацію, розширену систему фарбування, чудові файли довідки та широко використовувану базу даних підтримки. Завдяки можливості розшарування практично всі фрактали, які ви бачите в цій програмі, не є «чистим» фрактальним зображенням, а мозаїкою

або композицією багатьох зображень. Однак з художньої точки зору це чудова особливість, яка відкриває нові шляхи для творчості [13].

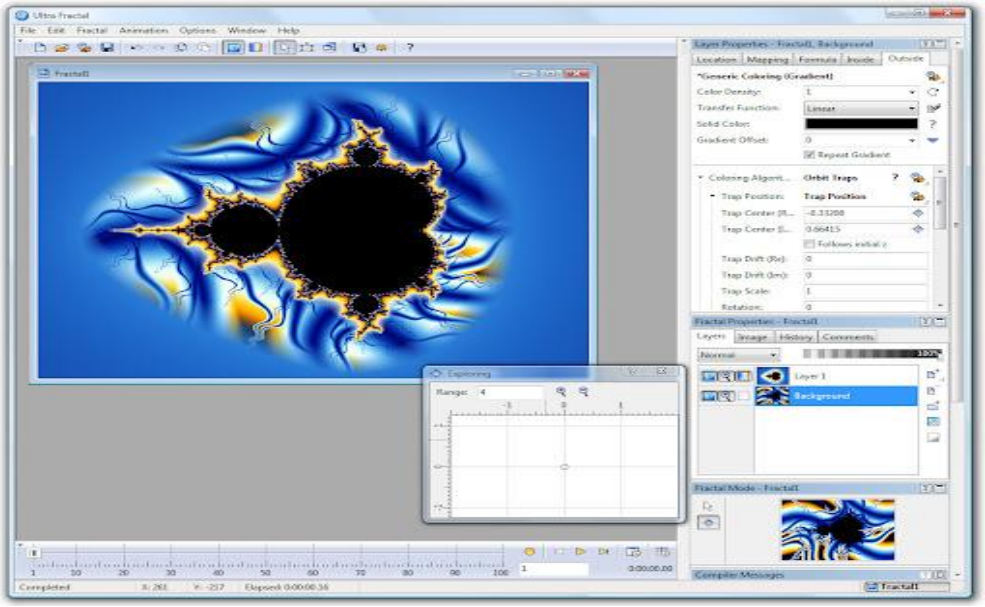

Рисунок 1.1 – Інтерфейс програмного додатку Ultra Fractal

Розглянемо аналог XenoDream. Програма дозволяє створювати 3D-фрактали IFS. Ви не використовуєте формули, як у більшості фрактальних програм, натомість ви працюєте з візуальними будівельними блоками, які називаються холонами, і звичайними операціями, такими як масштаб, поворот і положення, створюючи «фрактальні скульптури». Ця програма має безліч додаткових функцій, включаючи світлові ефекти, шари, метаморфи, фільтри, анімацію та багато інших розширених функцій програми графічного типу (рис. 1.2). Є пробна версія цього програмного забезпечення [14].

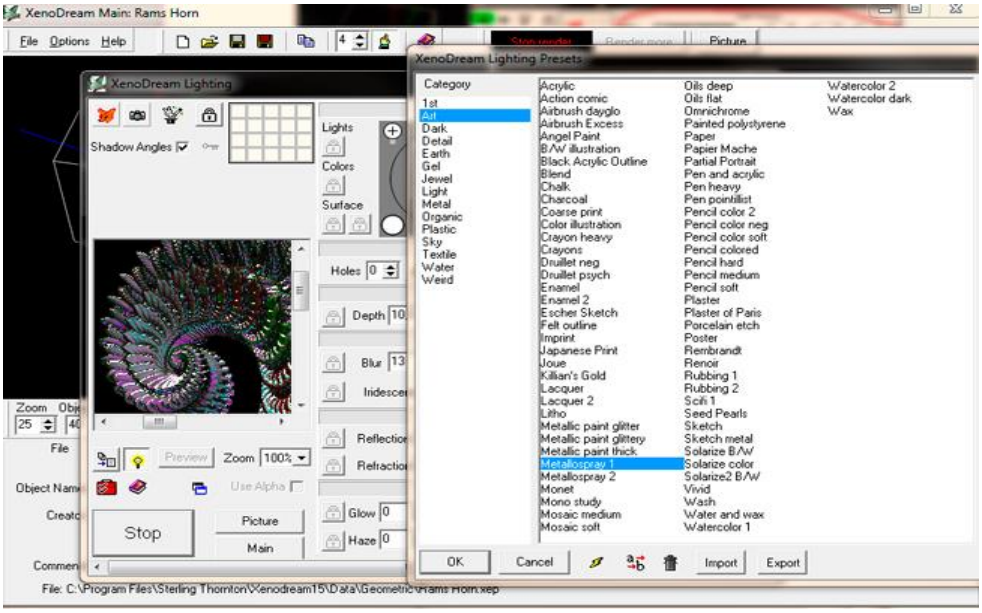

Рисунок 1.2 – Інтерфейс програмного додатку XenoDream

Розглянемо аналог Jux. Від творців Xenodream, Jux є фрактальним дослідником для 2D наборів Джулія та Мандельброта (рис. 1.3). Він містить різноманітні формули. Має красиві кольорові та світлові ефекти. Він простий у використанні, але немає редактора формул чи сценаріїв [15].

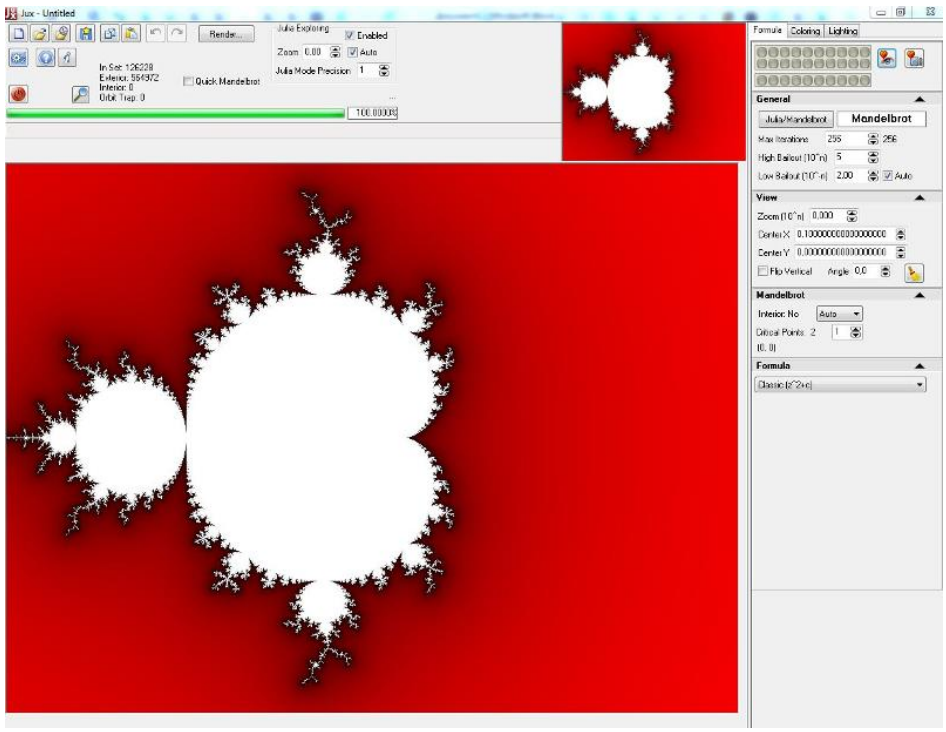

Рисунок 1.3 – Інтерфейс програмного додатку Jux

Розглянемо аналог Fractal Explorer. Fractal Explorer — чудова і популярна безкоштовна програма, яка створює фрактальні зображення та анімацію (рис. 1.4). За допомогою FE ви можете генерувати поліноміальні/ітераційні набори (подібні Мандельброту/Джулії/Ньютону), орбітальні фрактали, ітераційну функціональну систему (IFS), фрактальні ландшафти та тривимірні «дивні» атрактори. Ця програма є однією з найбільш багатофункціональних безкоштовних програм, доступних на даний момент [16].

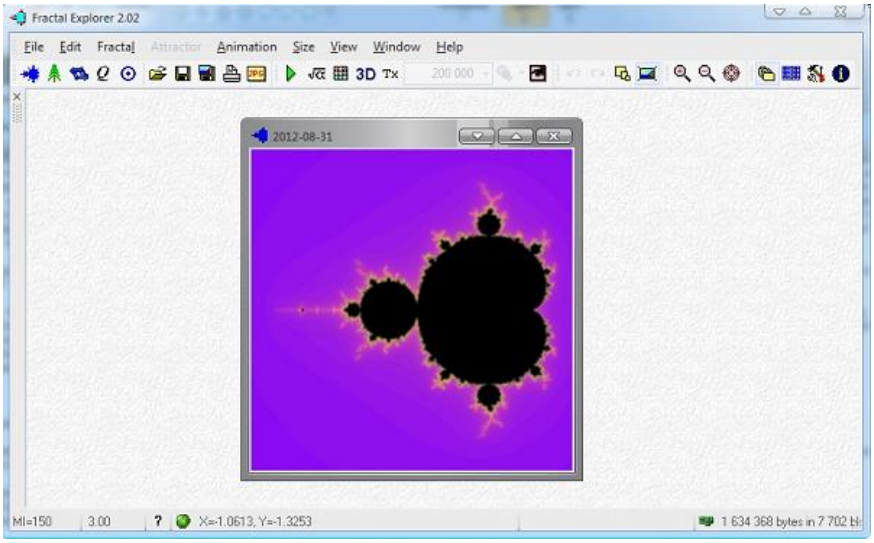

Рисунок 1.4 – Інтерфейс програмного додатку Fractal Explorer

Розглянемо аналог XaoS. XaoS — це інтерактивний фрактальний зумер у реальному часі, який дозволяє легко та плавно наближати фрактал. Він відображає багато різних типів фракталів, включаючи Мандельброта, Барнслі, Ньютона, Фенікса та багато інших. Ця програма створює чудові «класичні» фрактали [17].

Розглянемо аналог Apophysis. Apophysis — безкоштовна програма для Windows для створення незвичайних фракталів Fractal Flames і IFS (рис. 1.5). Його можна використовувати в поєднанні з програмою Ultra Fractal. Під час останньої перевірки фактично не було файлів довідки, доступних у програмі, але деякі підручники доступні в інших місцях [18].

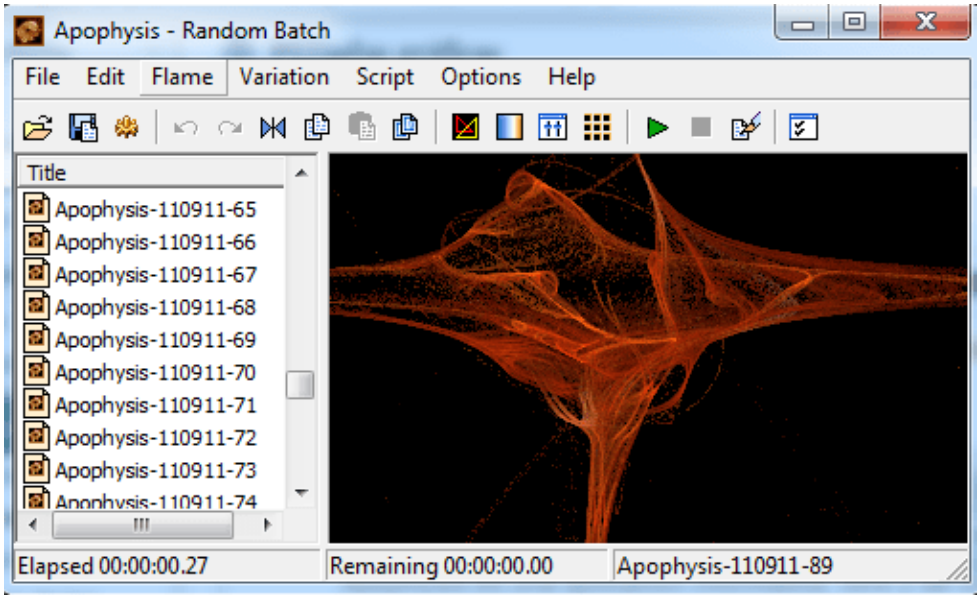

Рисунок 1.5 – Інтерфейс програмного додатку Apophysis

Розглянемо аналог Fractal Science Kit. Fractal Science Kit — це фрактальний генератор, який генерує фрактальне зображення з набору властивостей, які можна встановити для керування процесом генерації фракталу (рис. 1.6). Властивості прикладу включають тип фракталу, розмір зображення та параметри для керування генерацією орбіти, нормалізації даних, передвибірки, тиснення, згладжування, корекції гамми тощо [19].

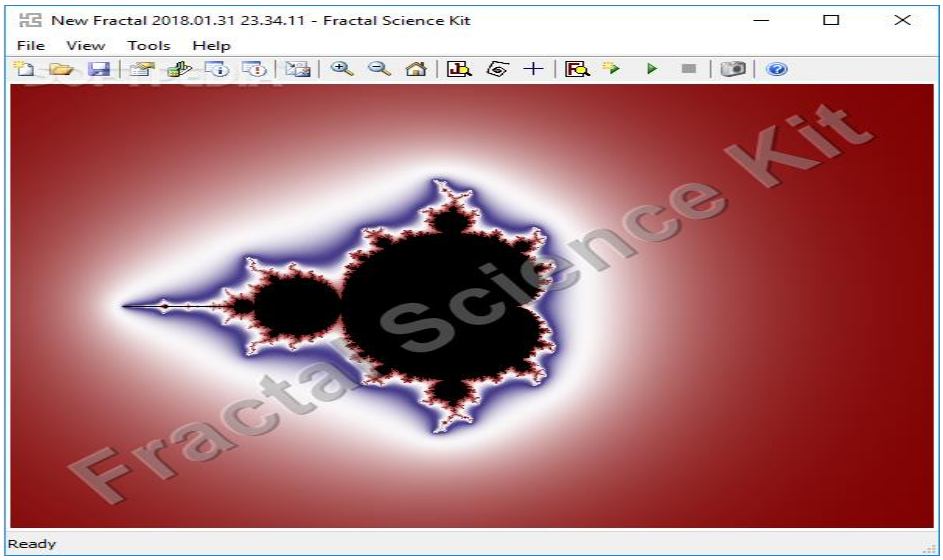

Рисунок 1.6 – Інтерфейс програмного додатку Fractal Science Kit

Розглянемо аналог ChaosPro. ChaosPro — це безкоштовний генератор фракталів у реальному часі з підтримкою багатьох різних типів фракталів (2D та

3D), підтримкою реального кольору та анімації. Завдяки інтегрованому компілятору це досить швидко, навіть якщо ви пишете власні формули в ChaosPro [20].

1.4. Аналіз використання конструктивно-продукційного підходу в проектуванні програмних систем

Проектування – це процес визначення архітектури, компонентів, інтерфейсів та інших характеристик системи або її частини. Результатом проектування є проект – цілісна сукупність моделей, властивостей чи характеристик, що описані у формі, придатній для реалізації системи.

Проектування, поряд з аналізом вимог, є частиною великої стадії життєвого циклу системи, яка називається визначенням системи. Результати цієї стадії є вхідний інформацією стадії реалізації (втілення) системи.

Проектування системи спрямоване уявлення системи, відповідне передбаченої мети, принципам і планам; воно включає оцінку та прийняття рішень щодо вибору таких компонентів системи, які відповідають її архітектурі та укладаються у вказані обмеження.

В даний час існує сильна тенденція розглядати архітектурне та детальне проектування як різні види діяльності; робляться спроби визначити їх як окремі практики, проте ці види проектування значною мірою переплетені. Архітектурні рішення в порівнянні зі "звичайними" проектними рішеннями розглядаються як абстрактніші, концептуальні та глобальні; вони націлені успіх всієї місії і найбільш високорівневі структури системи. Детальне проектування, своєю чергою, визначається як процес деталізації та розширення попереднього проекту (архітектури) настільки, коли проект повністю готовий до реалізації.

1.5. Огляд літератури

Більше двадцяти років минуло з того часу, як Бенуа Мандельброт опублікував своє знамените зображення так званої безлічі Мандельброта [21]. Ця картинка кардинально змінила наш погляд на математичний та фізичний Всесвіт. Ця книга розглядає не той чи інший клас проблем, а підхід до опису математичного та

фізичного Всесвіту в цілому. Фрактали (термін, придуманий автором) настільки міцно вкоренилися в нашій свідомості, що зараз дуже складно згадати той психологічний шок, який ми зазнали у момент їхньої появи. Ця багато ілюстрована книга поєднує ранні статті автора, які сьогодні стали бібліографічною рідкістю, з розділами, що описують історію розвитку фрактальної геометрії. Ключові теми книги – квадратична динаміка, множини Жюліа і Мандельброта, неквадратична динаміка, клейнові граничні множини та міра Мінковського.

Класична робота засновника теорії фракталів, відомого американського математика Б. Мандельброта, яка витримала за кордоном кілька видань і була перекладена багатьма мовами. Переклад російською мовою виходить із великим запізненням (перше англійське видання вийшло 1977 р.). За минулий період книга зовсім не застаріла і залишається найкращим та основним введенням у теорію фракталів та фрактальну геометрію [22]. Написана в живій та яскравій манері, вона містить безліч ілюстрацій (у тому числі кольорових), а також прикладів з різних галузей науки.

Перший повноцінний навчальний посібник з нової математичної дисципліни, що швидко розвивається – досі російською мовою виходили лише монографії [23].

Добре підібрані вправи та алгоритми роблять книгу відмінним посібником для студентів старших курсів та аспірантів, фахівців із додатків цієї теорії у різних галузях від біології до лінгвістики.

Основна мета книги – допомогти читачеві глибше зрозуміти, що таке самоподібність – можливо, найбільш важливу з симетрії, що зустрічаються в природі, а також продемонструвати найширший діапазон застосувань масштабної інваріантності у фізиці, хімії, біології, музиці і, особливо, в образотворчому мистецтві. Матеріал викладено на доступному рівні та забезпечений безліччю ілюстрацій [24].

1.6. Конструктори та композитні конструктори

Назвемо конструктором трійку [8]:

$$C = \langle M, \Sigma, \Lambda \rangle,$$

де M – неоднорідний поповнюваний носій структури, Σ – сигнатура операцій (і відповідних відносин) зв'язування, підстановки та виведення та операцій над атрибутами, Λ – множина (формальних і неформальних) тверджень інформаційного забезпечення конструювання (ІОК). ІОК (конструктивна аксіоматика) включає: онтологію, мету, правила, обмеження, початкові умови та умови завершення конструювання.

Передбачається низка уточнюючих перетворень узагальненого конструктора.

Спеціалізація визначає онтологію предметної області: семантичну природу носія, загальне для сімейства завдань мети, кінцева безліч операцій та їх семантику, атрибутику операцій, порядок їх виконання та обмеження. Операція спеціалізації виконується зовнішнім виконавцем.

Інтерпретація полягає у зв'язуванні операцій сигнатури з алгоритмами виконання деякої алгоритмічної структури. Під час інтерпретації виконується зв'язування моделей конструктора та внутрішнього виконавця процесу конструювання. Результатом інтерпретації є конструктивна система. Операція інтерпретації $_S\mapsto$ виконується зовнішнім виконавцем.

Конкретизація конструктора передбачає завдання конкретних правил, обмежень, початкових умов та умови завершення конструювання, конкретизації елементної бази носія: множини нетермінальних та термінальних символів з їх атрибутами і, при необхідності, значень атрибутів. Після інтерпретації та конкретизації (, що виконується зовнішнім виконавцем) конструктивна система має все необхідне для самостійного формування конструкцій.

Інтерпретація полягає у зв'язуванні операцій сигнатури з алгоритмами виконання деякої алгоритмічної структури. Під час інтерпретації виконується зв'язування моделей конструктора та внутрішнього виконавця процесу конструювання. Результатом інтерпретації $_I\mapsto$ є конструктивна система. Операція інтерпретації виконується зовнішнім виконавцем.

Конкретизація конструктора передбачає завдання конкретних правил, обмежень, початкових умов та умови завершення конструювання, конкретизації елементної бази носія: множини нетермінальних та термінальних символів з їх атрибутами і, при необхідності, значень атрибутів. Після інтерпретації та конкретизації ($_K\mapsto$, що виконується зовнішнім виконавцем) конструктивна система має все необхідне для самостійного формування конструкцій.

Реалізація ($_R\mapsto$), що виконується внутрішнім виконавцем системи, полягає у формуванні конструкції з елементів носія шляхом виконання алгоритмів, пов'язаних з операціями сигнатури. Реалізація можлива тільки для попередньо спеціалізованого, інтерпретованого та конкретизованого конструктора.

1.6.1. Основні особливості середовища розробки програмних додатків Visual Studio Code

Visual Studio Code (VS Code) – редактор вихідного коду, розроблений Microsoft для Windows, Linux та MacOS. Позиціонується як «легкий» редактор коду для кросплатформної розробки веб- та хмарних програм. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense та засоби для рефакторингу. Має широкі можливості для кастомізації: теми користувача, поєднання клавіш і файли конфігурації. Поширюється безкоштовно, розробляється як програмне забезпечення з відкритим вихідним кодом, але готові зборки поширюються під ліцензією проприетарной.

1.6.2. Особливості мови JavaScript та обгрунтування її обрання для розробки програмної системи

JavaScript – мова програмування, що є прототипно-орієнтованою. Він відображає мову ECMAScript, прототипом якого спочатку і був. Перша варіація з'явилася ще 1995 року і з того часу постійно вдосконалювалася, поки не дійшла нинішнього вигляду.

Найчастіше ця мова використовується в розробці додатків та браузерах з метою надання їм інтерактивності та «живості».

Базовою особливістю цієї мови наголошується на тому, що на неї вплинули інші (Python, Java та ін.) мови програмування з метою надання максимального комфорту JavaScript та легкості у розумінні його тими користувачами, які не мають відповідної освіти та глибинних знань – не програмістами. JavaScript – офіційно зареєстрована торгова марка компанії Oracle.

За допомогою нього доступні для виконання такі функції:

- можливість змінювати сторінки браузерів;
- додавання чи видалення тегів;
- зміна стилів сторінки;
- інформація про дії користувача на сторінці;
- запит доступу до випадкової частини вихідного коду сторінки;
- внесення змін до цього коду;
- виконання дії з cookie-файлами.

Область застосування цієї мови напрочуд велика і нічим не обмежена: серед програм, які використовують JS, присутні і тестові редактори, і програми (як для комп'ютерів, так і мобільні і навіть серверні), і прикладне програмне забезпечення.

Переваги JavaScript:

- жоден сучасний браузер не обходиться без підтримки JavaScript;
- з використанням написаних на JavaScript плагінів та скриптів упорається навіть не фахівець;
- корисні функціональні налаштування;
- мова, що постійно вдосконалюється – зараз розробляється бета-варіація проекту, JavaScript2;
- взаємодія з програмою може здійснюватися навіть через текстові редактори – Microsoft Office та Open Office;
- перспектива використання мови у процесі навчання програмування та інформатики.

Недоліки JavaScript:

- низький рівень безпеки через повсюдний та вільний доступ до вихідних кодів популярних скриптів;
- безліч дрібних дратівливих помилок кожному етапі роботи. Більшість їх легко виправляється, та їх наявність дозволяє вважати цю мову менш професійним, порівняно коїться з іншими;
- повсюдне поширення. Своєрідним недоліком можна вважати той факт, що частина програм, що активно використовуються (особливо додатків) перестануть існувати за відсутності мови, оскільки цілком базуються на ньому.

1.7. Формування вимог до системи

- розроблювані програмні продукти повинні мати можливість бути встановленими (запущеними) на існуючих апаратно-програмних засобах (для виключення надмірної кількості серверів та персональних комп'ютерів);
- при необхідності застосування додаткових технічних засобів, обов'язкове письмове узгодження складу та характеристик необхідних апаратних та програмних засобів з технічними фахівцями. Обладнання, що знову поставляється, повинно мати опис (Посібник користувача) російською мовою та комплект необхідних системних програм (драйверів) на CD або DVD диску (для складових пристроїв – на кожен модуль);
- рівень систем, що розробляються, повинен забезпечувати можливість запуску системи (інсталяції програмного забезпечення), а також необхідного поточного обслуговування (заміна блоків, індексація баз даних, архівування, оновлення модулів і т.д.) силами технічних фахівців.

1.7.1. Вимоги від зацікавлених суб’єктів

Основні вимоги:

- візуалізація побудови різних рівнів фракталів;
- використання мови програмування JavaScript.

1.7.2. Результати та висновки огляду літератури

Під час ознайомлення з літературою присвяченій викладу основних ідей фрактальної та мультифрактальної геометрії. Приклади різних фрактальних структур можна зустріти у багатьох явищах природи. Фрактальні образи з успіхом використовуються при описі хаотичної поведінки нелінійних динамічних та дисипатинних систем, турбулентного перебігу рідини, неоднорідного розподілу матерії у Всесвіті, при дослідженні тріщин та дислокаційних скупчень у твердих тілах, при вивченні електричного пробою, дифузії та агрегації частинок. буд. Багато цікавих ідей фрактальної геометрії знайшли своє застосування в економіці при аналізі коливань курсу валют, у біології для пояснення морфологічної будови різних біологічних об'єктів, у фізиці твердого тіла для опису переходу Андерсона на метал-діелектрик та інших властивостей невпорядкованих систем.

Основна мета книг – допомогти читачеві глибше зрозуміти, що таке самоподібність – можливо, найбільш важливу з симетрії, що зустрічаються в природі, а також продемонструвати найширший діапазон застосувань масштабної інваріантності у фізиці, хімії, біології, музиці і, особливо, в образотворчому мистецтві. Книги будуть корисні і цікаві найширшому колу читачів.

Висновки до першого розділу

У роботі закладено основи конструктивно-продукційного моделювання (КПМ), у рамках якого є можливим моделювання будь-яких конструкцій та конструктивних процесів у галузі інформаційних технологій, будівництва, машинобудування, робототехніки, біології тощо. Запропонований апарат дозволяє формалізувати процеси і результати формування конструкцій різної природи, пов'язуючи елементи конструкцій і враховуючи властивості елементів, їх агрегатів (форм) і зв'язків.

Також розглядається застосування КПМ на формування фракталов. Цей підхід найближчий до L-систем. Однак він є більш пристосованим до роботи з атрибутикою та більш гнучким при завданні процесу виведення.

2 ОБГРУНТУВАННЯ НАПРЯМКУ ДОСЛІДЖЕННЯ КОНСТРУКТИВНО-ПРОДУКЦІЙНОГО ПІДХОДУ ДЛЯ ВИКОРИСТАННЯ ПРИ ПОБУДОВІ ГРАФІЧНИХ ОБЬЕКТІВ

2.1. Формування основної мети та задач дослідження

Необхідно спроектувати та розробити програмний продукт, за допомогою якого можна наочно подивитися зображення фрактальної графіки. Програма повинна дозволяти розкрити сутність фракталу – багаторазове самоповторення (всього зображення чи певної його частини). Інтерфейс має бути максимально зрозумілим. Швидкість роботи мусить бути такою, щоб збалансувати продуктивність і якість, тобто. при цій швидкості промальовується досить наочне зображення. Необхідна також можливість збереження фрактального зображення. Програма має бути інтуїтивно зрозумілою та "не відштовхувати при першому погляді". Можливостями програми мають бути доступні промальовування щонайменше десяти алгебраїчних і щонайменше двох геометричних фракталів.

Дослідити структуру та особливості фракталів на основі побудови різних видів фракталів (наприклад крива Госпера, сніжинка Коха та інші) використовуючи JavaScript.

2.2. Обґрунтування методу конструктивно-продукційного моделювання як методу вирішення поставленої задачі

Особливостями конструктивно-продукційного моделювання із застосуванням є:

- атрибутивність елементів та операцій;
- носій, що розширюється;
- модель виконавця у вигляді його базових алгоритмів;
- зв'язок операцій із алгоритмами їх виконання.

Операції зв'язування елементів конструктора з'єднують окремі елементи конструкції або їх частини (проміжні форми). У класичних формальних граматиках

використовується одна бінарна операція зв'язування (конкатенації) над елементами термінального та нетермінального алфавітів, проте для спеціалізованих граматик можуть використовуватись різноманітні операції зв'язування: за умовою, багатомісні, графічні елементи та ін.

Як відомо, продукційні L-системи (Lindenmayer system) широко використовуються для моделювання різних систем і процесів, комп'ютерної графіки, біології, музики та ін. Основна відмінна риса L-систем щодо інших класичних граматик складається у відсутності нетерміналів, атрибутивності терміналів, виконанні «паралельної» підстановки, порядку формування безлічі конструкцій, що виводяться, і аксіоми у вигляді початкової конструкції.

2.3. Обґрунтування використання графічної бібліотеки THREE.JS як засобу побудови та відображення графічних об’єктів

Бібліотека Three.js полегшує роботу з WebGL. При використанні Three.js відпадає необхідність написання шейдерів (але можливість залишається), і з'являється можливість оперувати звичними поняттями. Над бібліотекою працює велика кількість розробників.

Моделювання графіки з використанням Three.js можна порівняти зі знімальним майданчиком, оскільки ми маємо можливість оперувати такими поняттями як сцена, світло, камера, об'єкти та їх матеріали.

Три основні властивості бібліотеки Three.js включають:

- Scene – своєрідна платформа, де розміщуються всі об'єкти, які ми створюємо;
- Camera – по суті – це "око", який буде спрямований на сцену. Камера знімає та відображає об'єкти, які розташовані на сцені;
- Renderer – візуалізатор, який дозволяє показувати сцену камери.

У Three.js існує кілька типів камери:

- Perspective Camera;
- Stereo Camera;
- Orthographic Camera;

– Cube Camera.

Найпоширеніші з них – це Perspective Camera та Orthographic Camera.

2.4. Визначення процесу формування фрактальних об’єктів

Головною особливістю фрактальних структур є їхня самоподібна ієрархічна організованість у відповідному метричному просторі. Властивість нескінченної самоподібності означає точну масштабну інваріантність геометрично регулярної фрактальної структури щодо набору послідовних операцій відображення подібності методом ітерацій. Формально в рамках ітераційного методу існують два принципово різні підходи до формування регулярної фрактальної структури F: ін'єктивний та сюр'єктивний.

Розглянемо фрактальну топологію об'єктів у геометричному 2D просторі. Тоді відповідно до уявлень теорії фрактальних множин можна висловити таке:

1 Принцип модулярної будови регулярних фрактальних структур: Будь-яка регулярна фрактальна структура може бути представлена з однакових мінімальних модулів, будова та форма яких містить структурну інформацію як про саму фрактальну структуру, так і про будь-який її передфрактал.

2 Принцип ієрархії модулів самоподібних регулярних фрактальних структур: Самоподібна регулярна фрактальна структура може бути представлена як модулярна з її предфракталів.

3 Принцип детерміністичності ін'єктивно одержаних фрактальних структур: Упорядкована безліч ідентичних фрактальних структур, одержаних ін'єктивним способом в одиничному осередку структурованого простору, являє собою детерміністичну фрактальну структуру.

4 Принцип необмеженого зростання (еволюціонування) сюр'єктивно одержуваних фрактальних структур: При ітеруванні генератора фракталу сюр'єктивним способом фрактальна структура необмежено еволюціонує з ініціального осередку в оточуючий осередок у відповідності зі своїм коефіцієнтом подібності.

Сформульовані вище принципи покладено основою еволюційних моделей формування детерміністичних фрактальних структур, упорядкованих у 2D просторі множин і мультимножин замкнутих фрактальних кривих і використані при цілеспрямованому пошуку та інтерпретації трибологічних властивостей поверхні композиційних матеріалів і покриттів.

2.5. Визначення структури системи

Визначення структури системи

- структура системи – це стійка впорядкованість її елементів та зв'язків;
- структура є формою уявлення деякого об'єкта як складових частин;
- структура – це безліч усіх можливих відносин між підсистемами та елементами всередині системи;
- під структурою розуміється сукупність елементів та зв'язків між ними, які визначаються, виходячи з розподілу функцій та цілей, поставлених перед системою;
- структура системи – це те, що залишається незмінним у системі при зміні її стану, при реалізації різних форм поведінки, при здійсненні системою операцій тощо.

У сукупності дані визначення досить добре відображають те головне, що є у будь-якій структурі: елементний склад, наявність зв'язків, інваріантність (незмінність) у часі. По суті лише остання властивість дозволяє розмежувати поняття системи та структури. Проте врахувати лише інваріантність структури ще недостатньо. Оскільки структура – це частина системи, необхідно чітко вказати, яка саме частина, які властивості та ознаки системи є структурними, а які – ні. Відповіді ці питання, звісно, залежить від цілей дослідження системи, що також необхідно враховувати. Тому далі під структурою розумітимемо сукупність тих властивостей системи, які є суттєвими з точки зору проведеного дослідження і мають інваріантність на всьому цікавому дослідника інтервалі функціонування або на кожному непересічним підмножині, на які розбитий інтервал функціонування.

Залежно від цілей вивчення дослідника цікавитимуть різні інваріантні у часі властивості системи. З визначення випливає, що для однієї і тієї ж системи можна побудувати різні структури та між системою та її структурою відсутня однозначна відповідність.

Підбиваючи підсумки, можна сказати, що формування структури є частиною вирішення загальної задачі побудови системи, причому такою, яка не визначає заздалегідь систему в цілому, а лише виявляє її конфігурацію. Отже, побудова структури – самостійне завдання, що передує синтез системи загалом і полегшує його.

Система виділяється людиною із зовнішнього "фону" за функціональними або просторовими ознаками (наприклад, живі та технічні системи – швидше за просторовим; економічні, організаційні – за функціональним).

Висновки до другого розділу

У цьому розділі розповідається про особливості методу конструктивно-продукційного моделювання, про визначення процесу формування фрактальних об’єктів, визначення структури системи. Також визначаюься переваги бібліотеки three.js як засобу побудови та відображення графічних об’єктів.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ФОРМУВАННЯ ДВОВИМІРНИХ ФРАКТАЛІВ У ПРОСТОРІ

3.1. Зовнішнє проектування системи

Для реалізації поставленої задачі було створено наступні файли:

– index.html (В ньому розташовано основні елементі коду для створення програми);
– style.scss (надає можливість гарного виду програми);
– index.js (реалізує всі функції для роботи програми).

Та папки з файлами:

– editor;
– fonts;
– icons;
– texture.

3.1.1. Розробка інтерфейсу користувача

Інтерфейс користувача виглядає наступним чином (рис. 3.1):

Рисунок 3.1 – Інтерфейс користувача

Інтерфейс користувача складається з двох частин:

– бокової панелі (панелі інструментів);
– зони побудови фракталів.

3.1.2. Формат вихідних даних

Вхідними параметрами є:

– ступень фракталу;
– довжина лінії;
– координати початку малювання фракталу.

3.2. Попередній етап проектування структури та архітектури системи

Конструктор трикутника Серпінського

Виконаємо конкретизацію конструктора C_{SERP}, що призначена для формування класичного фракталу – трикутника (серветка) Серпінського.

$$C_{GI} = \langle M_{GI}, \Sigma_{GI}, \Lambda_{GI} \rangle_K \mapsto C_{SERP} = \langle M_{GI}, \Sigma_{GI}, \Lambda_{SERP} \rangle,$$

де $\Lambda_{SERP} = \Lambda_{GI} \cup \Lambda_4$, частина ИОК Λ_4 визначає:

– початкові умови – початкова фігура (термінал) – ▲, з атрибутами розмірів $h = \sqrt{3}/2$, $l = 1$ та положенням центру $x_ц = 0, \quad y_ц = 0$;
– правило підстановки $\psi = < s, g >$, в якому s складається з єдиного відношення підстановки, представленого на рис. 3.2 та операцій над атрибутами $g = < \tau_2 /(h,2), \quad \tau_2 /(l,2) >$;
– розмір фігури у правій частині відношення такий самий, як і у лівій;
– умова завершення $N = \infty$.

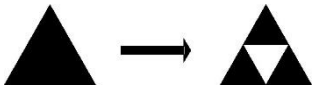

Рисунок 3.2 – Відношення підстановки конструктора

Реалізацію представимо у вигляді послідовності сентенціальних форм (рис. 3.3), зображення збільшено по відношенню до для більшої розбірливості).

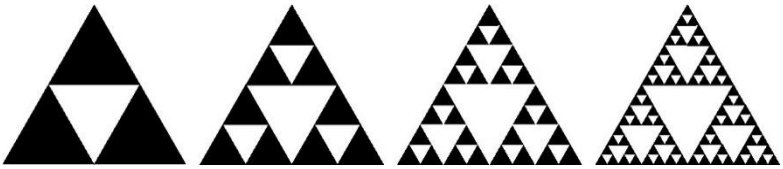

Рисунок 3.3 – Послідовність конструювання трикутника Серпінського

Реалізація виконується в такий спосіб. Операція часткового виведення здійснює перетворення форми f_i у f_{i+1}:

– початкову форму f_0 (▲), відповідно до відношення підстановки, одноразово застосовуючи операцію підстановки, перетворює на f_1:
– виконуючи операції над атрибутами, зменшує розміри трикутників лівої та правої частини відношення підстановки s.

Наступна форма f_2 отримана операцією часткового виведення з триразовим застосуванням операції підстановки з подальшим зменшенням ще вдвічі розмірів фігур у лівій та правій частині відношення s і так далі.

Операція повного виведення з багаторазовим застосуванням операції часткового виведення виконує перетворення від f_0 до f_∞.

Килим є об'єднання $N=3$ копій тих що істотно не перетинаються, зменшених вдвічі копій (коефіцієнт подоби $r=1/2$). Отже, представлені фрактали мають дрібну розмірність.

$$d=\frac{\log N}{\log 1/r}=\frac{\log 3}{\log 2}=1,585.$$

Зазначимо, що дробова розмірність отримана у припущенні, що зображене білим – порожньо. Вважаючи, що внутрішні трикутники білі, то отримаємо розмірність – два, у всіх сентенційних формах, у тому числі й у граничній (f_∞ при $n\to\infty$).

3.3. Виділення основних структурних та логічних сутностей та зв’язків між ними

Програмний засіб ділиться на логічні частини (контексти), що взаємодіють між собою в разі потреби та виконують кожен свою частину роботи. Кожен із логічних контекстів підписується на інші, відповідно до бізнес логіки програмного засобу. Підпис логічних контекстів відбувається в місці ініціалізації загального контексту програмного засобу.

Кожен контекст має можливість для внесення початкових даних, що потрібні йому для коректної роботи. Наприклад контекст інструментів, відповідальний за роботу панелі інструментів, що знаходиться в лівій частині інтерфейсу програмного засобу, виконую налаштування поточного інструменту на початку ініціалізації програмного засобу.

Кожен з контекстів програмного засобу виконую підпис на потрібні йому інші контексти відповідно до бізнес логіки програмного засобу.

Так, наприклад, контекст інтерфейсу підписується на:

1) події контексту інструментів, пов’язані зі зміною поточного інструменту для роботи з програмним засобом, для того щоб поновити вигляд кнопок на інтерфейсі користувача;

2) подію контексту вибраних елементів, пов’язану з вибором одного елементу на полотні, для того щоб відобразити кнопки інструментів доступні тільки в контексті одного вибраного елементу на полотні;

3) подію контексту вибраних елементів, пов’язану з вибором декількох елементів на полотні, для того щоб відобразити кнопки інструментів доступні тільки в контексті декількох вибраних елементів на полотні;

4) подію контексту вибраних елементів, пов’язану з відміною вибору всіх вибраних елементів, для того щоб відключити відображення елементів інтерфейсу користувача, пов’язаних з вибраними елементами;

5) власну подію натиску на кнопку генерації результату роботи L-системи, для того щоб підготувати форму з окремим полоном для відображення на ній побудованого зображення.

Контекст буферу обміну підписується на:

1) подію контексту вибраних елементів, пов’язану з натисканнями комбінації клавіш ctrl+c та реагує на неї внесенням вибраних елементів до буферу обміну;

2) подію контексту вибраних елементів, пов’язану з натисканням комбінації клавіш ctrl+v з наявними вибраними елементами збереженими в контексті буферу обміну, для того щоб вставити додати їх на полотно.

Контекст переміщення підписується на:

1) подію контексту полотна, пов’язану з натисканням лівої кнопки миші на елемент на полотні, для того щоб зареєструвати даний елемент як поточний елемент що переміщують;

2) на подію переміщення курсору миші на полотні, та посилає нову подію відповідно до того, чи було зареєстровано поточний елемент для переміщення, чи ні;

3) на подію перетину променів рейкастеру з буферною площиною, то поновлює координати об’єкту що переміщують, відповідно до нових координат;

4) на подію відпускання лівої клавіші миші, очищуючи інформацію про поточний елемент переміщення та повідомляє про це інші контексти породжуючи подію про завершення переміщення елементу.

Контекст вибору елементів на полотні підписується на:

1) подію контексту інтерфейсу користувача, пов’язану з натисканням на кнопку групування вибраних елементів, для сформування набору з вибраних елементів, та передачі їх в контекст полотна для логічного групування в один єдиний графічний елемент;

2) подію контексту інтерфейсу користувача, пов’язану з натисканням на кнопку розгрупування вибраних елементів, для передачі інформації про групу

елементів в контекст полотна для її логічного розгрупування на окремі частити(елементи);

3) подію полотна породжувану при натисканні на елемент комбінацією клавіш ctrl+MouseLeft, реагуючи на неї внесенням натиснутого елементу в контекст вибраних елементів, або навпаки видаленням елементу з нього, у випадку коли він вже був занесений в контекст вибраних елементів;

4) подію контексту інтерфейсу, породжувану при зміні даних в полі для вводу термінального символу, що буде присвоєно елементу, та передає інформацію контексту відповідальному за присвоєння цих даних елементу на полотні;

5) подію полотна, що надсилається при натисканні комбінації клавіш ctrl+c, та реагує на неї внесенням вибраних елементів до контексту буферу обміну.

Контекст полотна реагує на:

1) подію зміни інструменту контексту інструментів, вимикаючи або вмикаючи взаємодію з камерою залежно від обраного інструменту;

2) власну подію натиску лівою кнопкою миші на об’єкт на полотні та транслює її іншим контекстам з відповідною інформацією про комбінації клавіш при натиску на кнопку миші;

3) подію переміщення миші з наявним графічним елементом збереженим в контексті переміщення для оновлення інформації про позицію елементу;

4) подію переміщення миші без наявного графічного елементу збереженого в контексті переміщення, для оновлення інформації про площину, що потрібна рейкастеру;

5) власну подію натиску на полотно в місці де немає жодного елементу, для виконання скидання відображення допоміжних точок на вибраних елементах;

6) власну подію кліку по об’єкту полотна, для того щоб активувати відображення допоміжних точок взаємодії з об’єктом;

7) подію початку переміщення, виконуючи відключення модулю камери на момент переміщення об’єкту;

8) на подію завершення переміщення об’єкту контексту переміщення, для виконання повторною побудови площини, якщо об’єктом переміщення виступала допоміжна точка взаємодії з об’єктом, та увімкнення модулю камери;

9) подію групування об’єктів, виконуючи групування переданих об’єктів та їх вставку на полотно вже єдиною групою;

10) подію розгрупування об’єктів, виконуючи розгрупування переданої групи об’єктів та їх вставку на полотно як окремі елементи;

11) подію контексту інтерфейсу на створення полотна для виводу результату генерації L-системи, виконуючи внесення результату побудови на дане полотно;

12) подію що надходить від контексту буферу обміну, котра сигналізує про те, що була виконана спроба вставки елементів комбінацією клавіш ctrl+v, та реагує на неї додаванням даних елементів на полотно за їх наявності.

3.3.1. Опис базової архітектури

Архітектура програмного засобу базується на подійному підході (шаблон Event Emitter). Event Emitter це шаблон, який можна реалізувати різними способами. Основна ідея полягає в тому, щоб створити надійну та гнучку основу для управління подіями та реалізувати можливість будь-яким елементам «підписатися» на нього (і бути в курсі того, що відбувається). Основна ідея використання подійного підходу в реалізації програмного засобу для побудови фракталів це використання його гнучкості для досягнення простоти реалізації базового інструментарію програмного засобу та його легкого вдосконалення або внесення суттєвих правок в логіку роботи програмного засобу з мінімальними витратами часу та інших ресурсів.

Для розробки програмного засобу було використано середу розробки VS Code. Visual Studio Code – це абсолютно нова платформа і набір технологій, призначені докорінно змінити підхід до створення програм. Передбачається використання близько 30 мов програмування, які звертаються до єдиної ієрархії класів, які забезпечують базові послуги. Програми для Visual Studio Code не компілюються – вони перетворюються на якусь проміжну мову, відомий як

Microsoft Intermediate Language (MSIL), і виконуються під керуванням віртуальної машини, що має назву Common Language Runtime (CLR). Такий підхід має низку переваг, оскільки в даному випадку всі мови мають доступ до єдиного набору сервісів, а завдяки тому, що всі вони перетворюються на проміжний код, не виникає проблем з написанням окремих фрагментів програми тією чи іншою мовою програмування та з їх подальшою інтеграцією у єдине ціле.

У Visual Studio Code інтенсивно використовуються XML і протокол SOAP. Останній дозволяє створювати додатки, засновані не так на концепції використання прив'язаних до тієї чи іншої платформі компонентів чи об'єктів, але в концепції сервісів. Протокол SOAP (розроблений фірмами Microsoft, DevelopMentor і Userland Software та підтримуваний великою кількістю компаній, включаючи IBM та Oracle) та Web-сервіси є ключовими елементами платформи Visual Studio Code.

Огляд Web-сервісів та пов'язаних з ними технологій публікується в цьому номері – див. статтю «Web нового покоління – Web-сервіси».

З погляду архітектури Visual Studio Code може бути описана таким чином:

- існує широкий набір сервісів, доступних з різних мов програмування;
- ці послуги реалізовані як проміжного коду, незалежно від базової архітектури;
- сервіси виконуються під керуванням віртуальної машини CLR, яка також керує ресурсами та стежить за виконанням програм.

Однією з основних цілей створення платформи Visual Studio Code є надання розробникам засобу для створення сервіс-орієнтованих додатків, здатних працювати на будь-якій апаратній платформі: персональному комп'ютері, PDA, мобільному телефоні тощо.

Всі мови програмування, реалізовані для платформи Visual Studio Code, матимуть доступ до широкого набору сервісів, які включають базові API Windows, COM-сервіси, інтерфейси доступу до даних (ADO.NET), інтерфейси для реалізації Web-сервісів, засоби роботи XML та багато інших.

3.3.2. Опис пакетів програмного засобу

Пакетна структура проекту відповідає структурі базової архітектури програмного засобу та налічує 8 основних пакетів, що використовуються в програмі та взаємодіють один з одним. Схему взаємодії пакетів програмного засобу приведено на рисунку нижче.

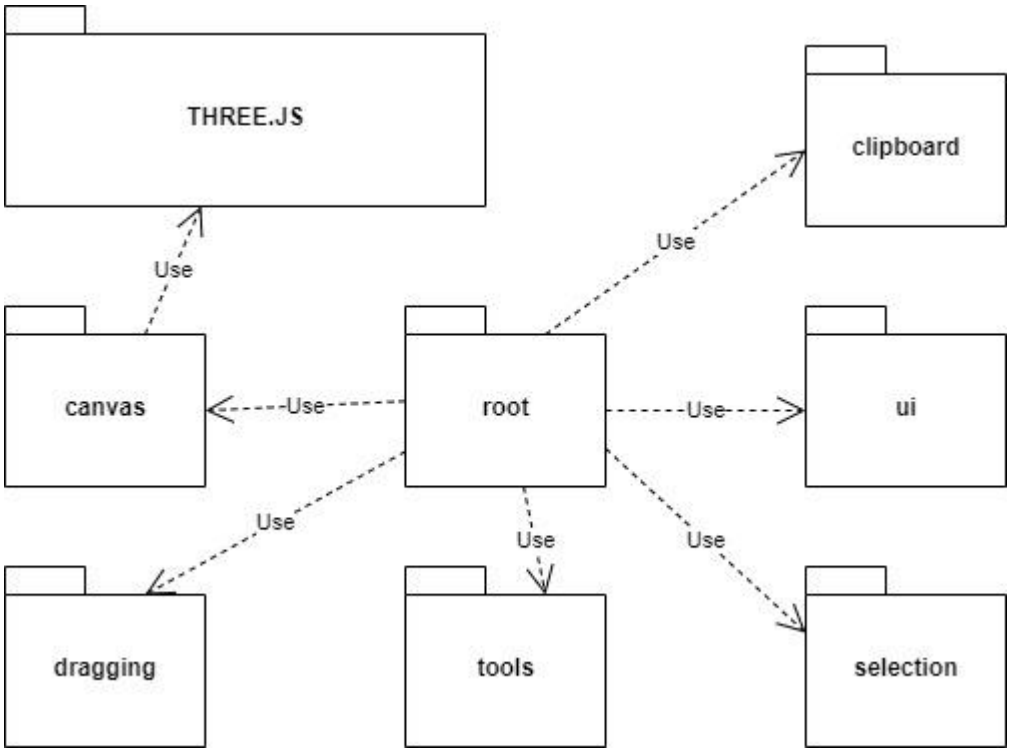

Рисунок 3.4 – UML–діаграма пакетів програмного засобу

3.3.3. Деталі архітектури програмної системи

Архітектура програмного засобу базується на шаблоні проектування EventEmitter. Логічні контексти програми доводять до відома іншим контекстам, що відбулася та чи інша подія, на яку можливо відреагувати.

Схема подійної архітектури програми наведено на рисунку нижче.

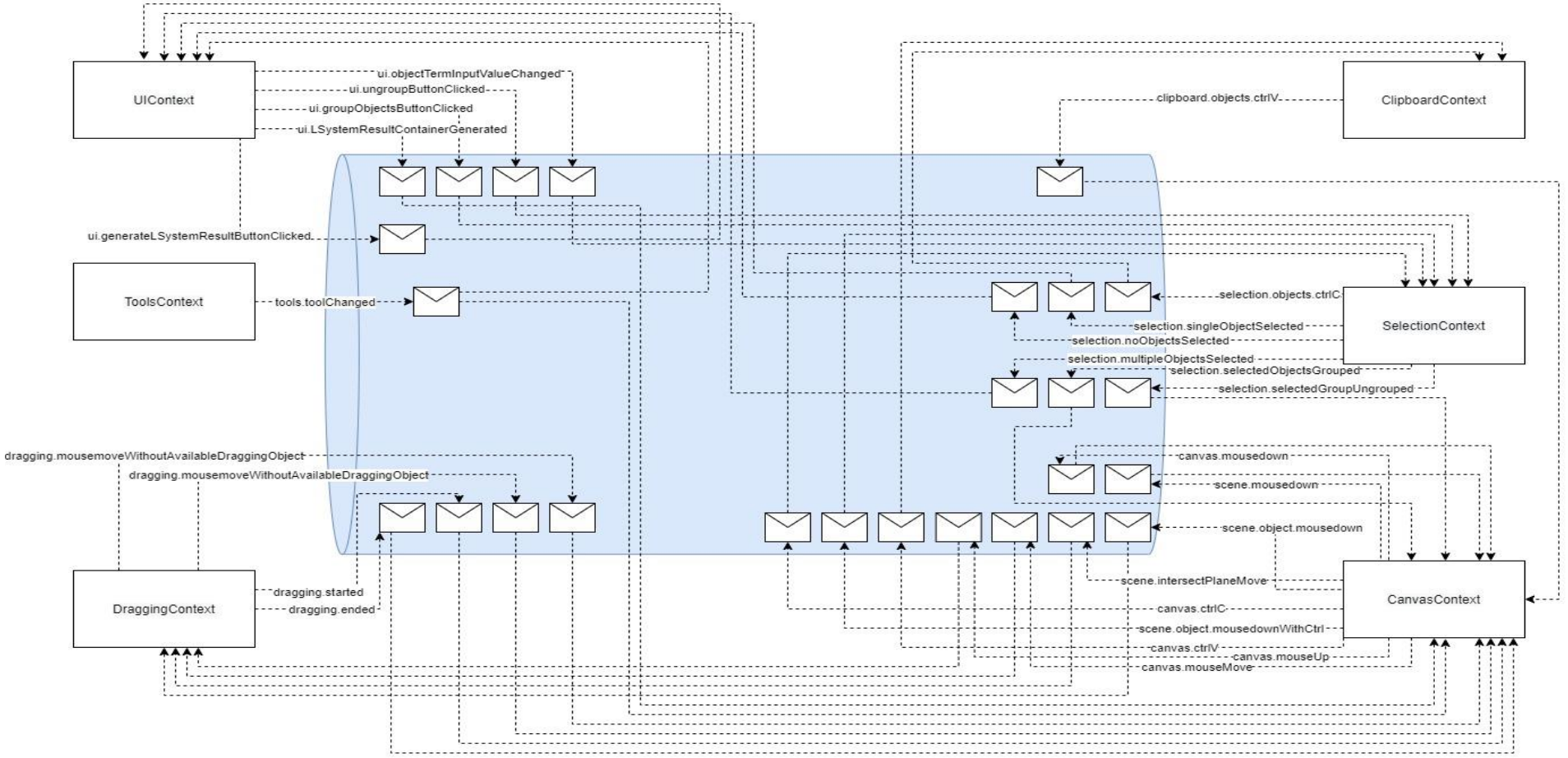

Рисунок 3.5 – Схема обміну подіями, організована в програмному засобі

3.4. Реалізація базового функціоналу

Загальноприйнята основа формування фракталів – стискаючі відображення. КПМ дозволяє формувати фрактали на основі відображень, що розширюються, але і одночасно стискаючих/розтискаючих. Розглянемо конструктор де $C_{13} = \langle M_{GI}, \Sigma_{GI}, \Lambda_{21} \rangle$, $\Lambda_{21} = \Lambda_{SERP} \setminus \{\psi\} \cup \{\psi_{19}, \psi_{20}, \psi_{21}, \psi_{22}\} \cup \Lambda_{22}$, Λ_{22}: позначимо f_i' і f_i'' – ліву і праву частину конструкції додається на -і-му кроці, $f_0' = f_0''$ – початковий трикутник, як в C_{SERP} ; $i = 1$; $l \leftarrow\!\lrcorner t_{21}$, $h \leftarrow\!\lrcorner t_{21}$ – висота і ширина трикутників в лівій і правій частині відносини $s \leftarrow\!\lrcorner \psi_{21}$, $l \leftarrow\!\lrcorner t_{22}$, $h \leftarrow\!\lrcorner t_{22} - s \leftarrow\!\lrcorner \psi_{22}$; $\psi_{19} =< s_{19}, g_4 >$, $\psi_{20} =< s_{20}, g_5 >$, $\psi_{21} =< s, g_6 >$, $\psi_{22} =< s, g_7 >$,

$s_{19} =< f_{i-1}' \to f_{i-1}' \square f_i' >$, $s_{20} =< f_{i-1}'' \to f_{i-1}'' \square f_i'' >$ $g_4 =<$ $_{\tau_2} -(x_{ц} \leftarrow\!\lrcorner f_i', l)$, $_{\tau_2}/(y_{ц} \leftarrow\!\lrcorner f_i', 1.25)$,

$_{\tau_2}/(l \leftarrow\!\lrcorner f_i', 1.25)$, $_{\tau_2}/(h \leftarrow\!\lrcorner f_i', 1.25) >$, $g_5 =<$ $_{\tau_2}/(y_{ц} \leftarrow\!\lrcorner f_i'', 0.8)$, $_{\tau_2}/(l \leftarrow\!\lrcorner f_i'', 0.8)$, $_{\tau_2}+(x_{ц} \leftarrow\!\lrcorner f_i'', l)$, $_{\tau_2}/(h \leftarrow\!\lrcorner f_i'', 0.8) >$,

$g_6 =<$ $_{\tau_0}/(l \leftarrow\!\lrcorner t_{21}, 2.5)$, $_{\tau_0}/(h \leftarrow\!\lrcorner t_{21}, 2.5) >$, $g_7 =<$ $_{\tau_0}/(l \leftarrow\!\lrcorner t_{22}, 1.6)$, $_{\tau_0}/(h \leftarrow\!\lrcorner t_{22}, 1.6) >$.

Сформований фрактал наведено на рис. 3.6.

Значення коефіцієнтів стиску в правилах $s \leftarrow\!\lrcorner \psi_{21}$ та $s \leftarrow\!\lrcorner \psi_{22}$ можуть бути як більші, так і менше 1.

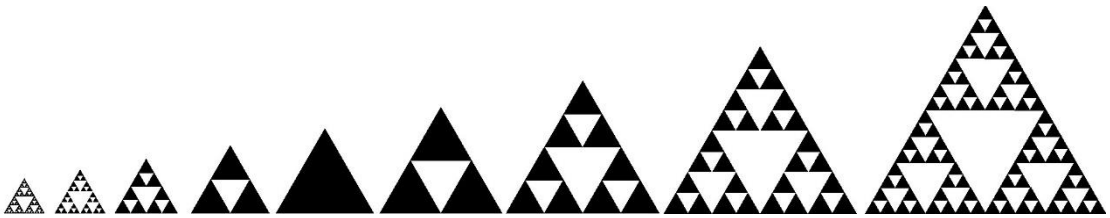

Рисунок 3.6 – Фрактал $K_G(C_{13})$

Таким чином, показано можливості формування нового, раніше невідомого та не вивченого класу фракталів. При відображенні, що розтискає, може бути отримана фігура, що заповнює всю площину, але з нульовою площею.

3.4.1. Формування зображень 2D в 3D терміналів, що будуть використані L-системою при формуванні кінцевого результату

Формування правил представлено на рис. 3.7.

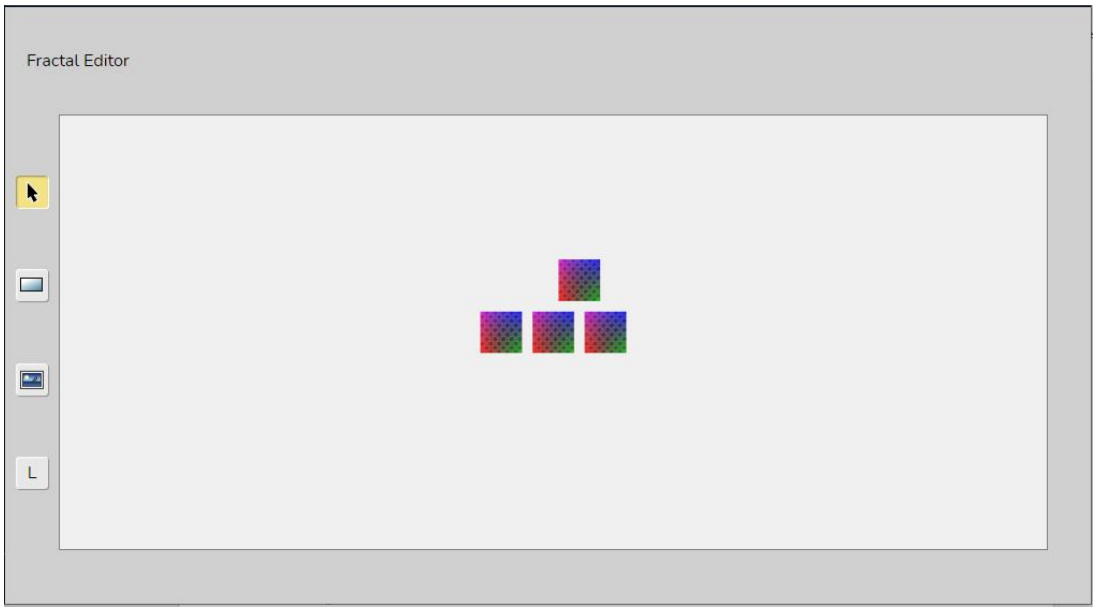

Рисунок 3.7 – Візуальне представлення аксіоми та правил

Інтерфейс користувача надає можливість розглянути побудовані об’єкти під іншим ракурсом. Приклад використання камери приведено на рис. 3.8.

Рисунок 3.8 – Використання камери на полотні

Переміщення об’єктів на полотні відбувається за допомогою утримання натиснутою лівої кнопки миші та переміщенням курсору у потрібному напрямку (рис. 3.9).

Рисунок 3.9 – Переміщення об’єкту на полотні

Наявна можливість приближення та віддалення просторової камери. Приклад використання наближення та віддалення приведено на рисунках нижче.

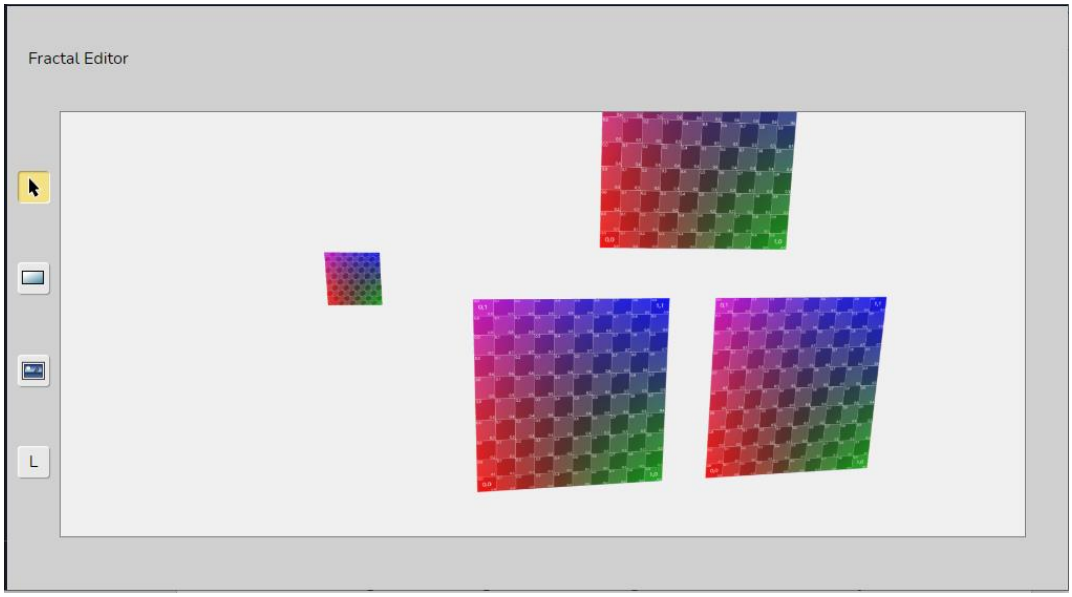

Рисунок 3.10 – Використання можливості наближення камери

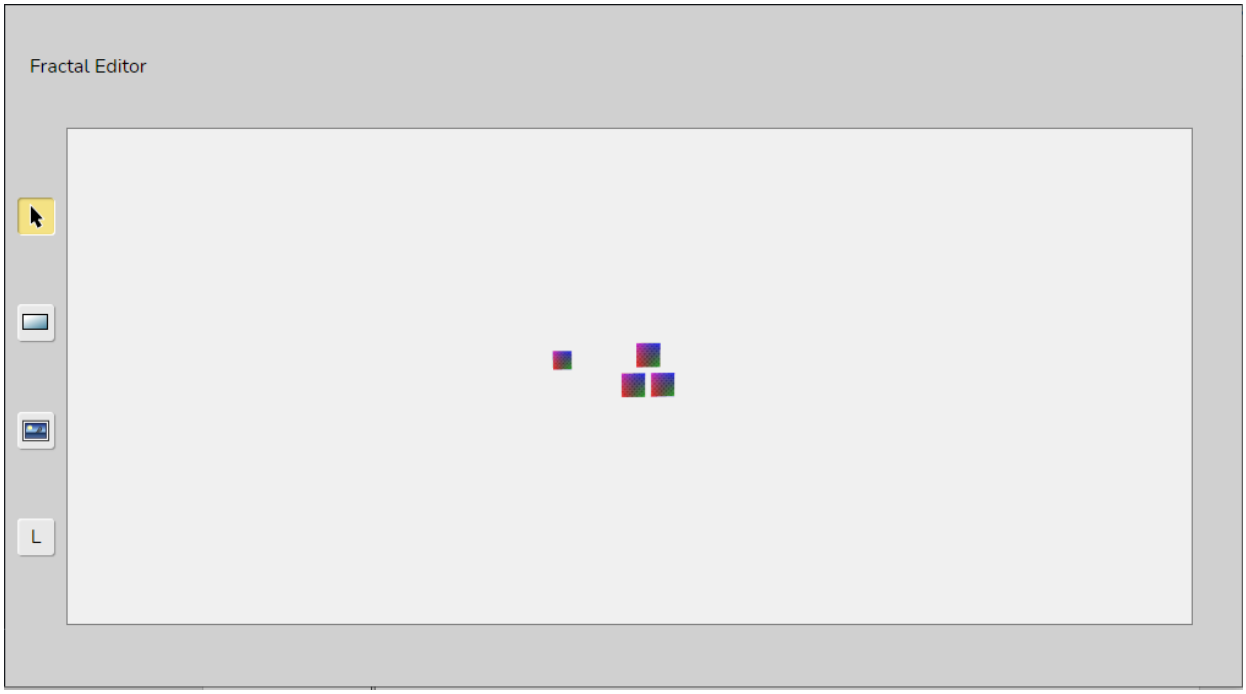

Рисунок 3.11 – Використання можливості віддалення камери

Для зміни форми площини потрібно скористатися опорними точками, що з’являються при виборі площини на полотні. Приклад деформації площини наведено на рис. 3.12.

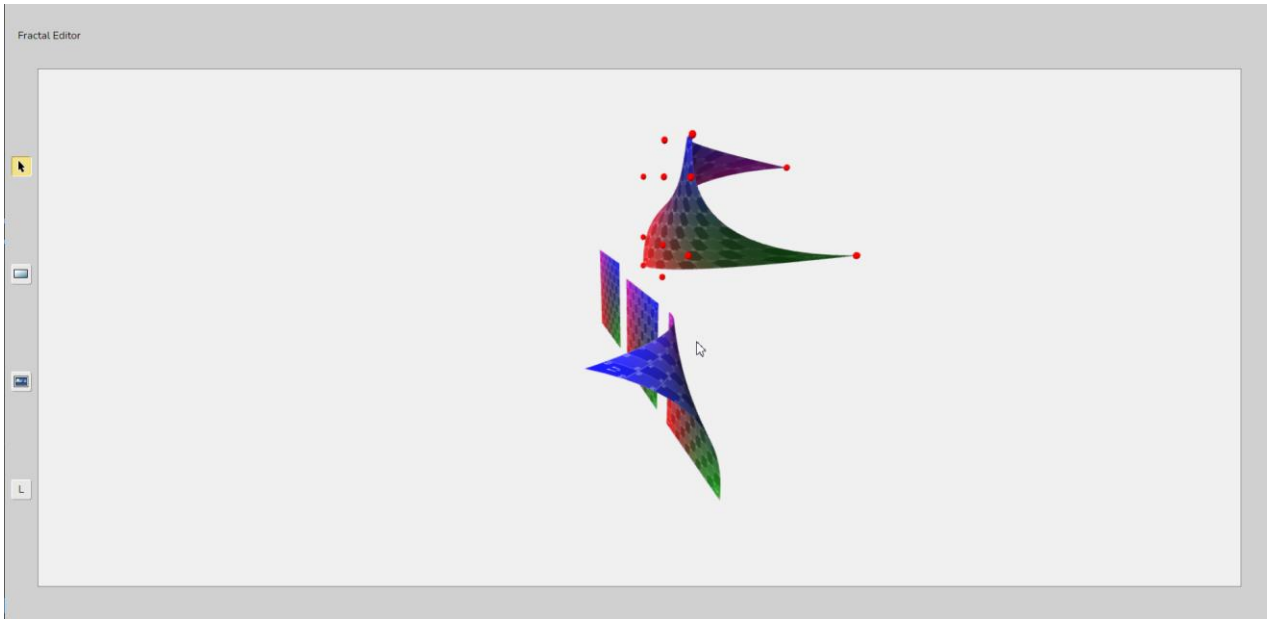

Рисунок 3.12 – Деформація площини

3.4.2. Присвоєння термінальних/нетермінальних символів побудованим об’єктам

Для спілкування з комп'ютерами розробляються спеціальні мови, які називаються штучними на відміну природних мов спілкування людей. Штучні мови повинні бути, з одного боку, зручними та зрозумілими для людини, а з іншого – повинні сприйматися пристроями. Поєднання цих вимог в одній мові є важким завданням, тому застосовуються засоби для перетворення текстів з мови, зрозумілої людині, на мову пристрою. Такі засоби називають трансляторами.

Транслятори, інтерпретатори та компілятори

Транслятор може бути типу, що інтерпретує або компілює. У першому випадку його називають інтерпретатором вхідної мови, а в другому компілятором.

Інтерпретатор послідовно читає речення вхідної мови, аналізує їх і одразу виконує. Компілятор не виконує пропозиції мови, а будує програму, яка може бути запущена для отримання результату.

На вхід компілятора подається текст, написаний мовою зрозумілій людині, а в результаті роботи компілятора створюється текст мовою, зручною для пристрою.

3.4.3. Генерація потрібного N покоління L-системою

Для того, щоб отримати результат побудови (результуючий фрактал) потрібно після створення правил L-системою задати потрібну кількість ітерацій, яка буде використана при запуску процесу генерації.

Для прикладу побудови візьмемо L=4;

Натиснувши на кнопку «L» для генерації результату обчислень, у нас з’явиться представлене на рисунку нижче.

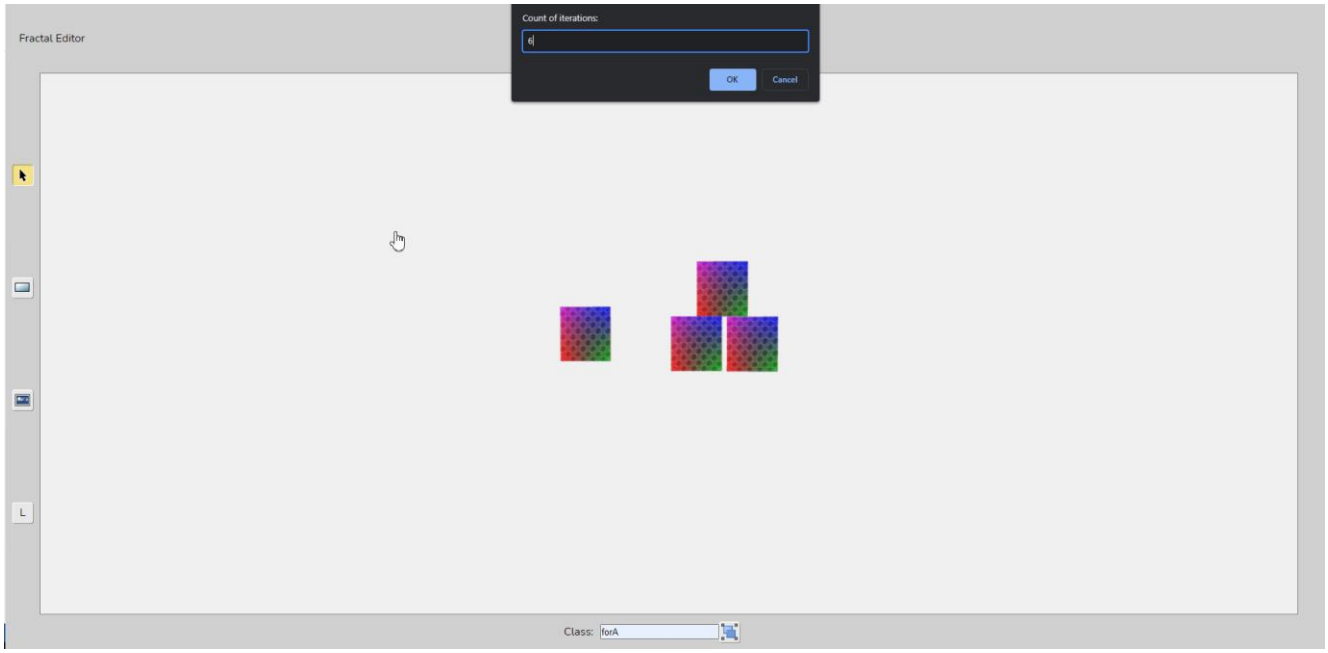

Рисунок 3.13 – Введення числа ітерацій роботи L–системи (L=4)

Результат генерації відображається на окремій формі, призначеній для огляду результату побудови фракталу (рис. 3.14).

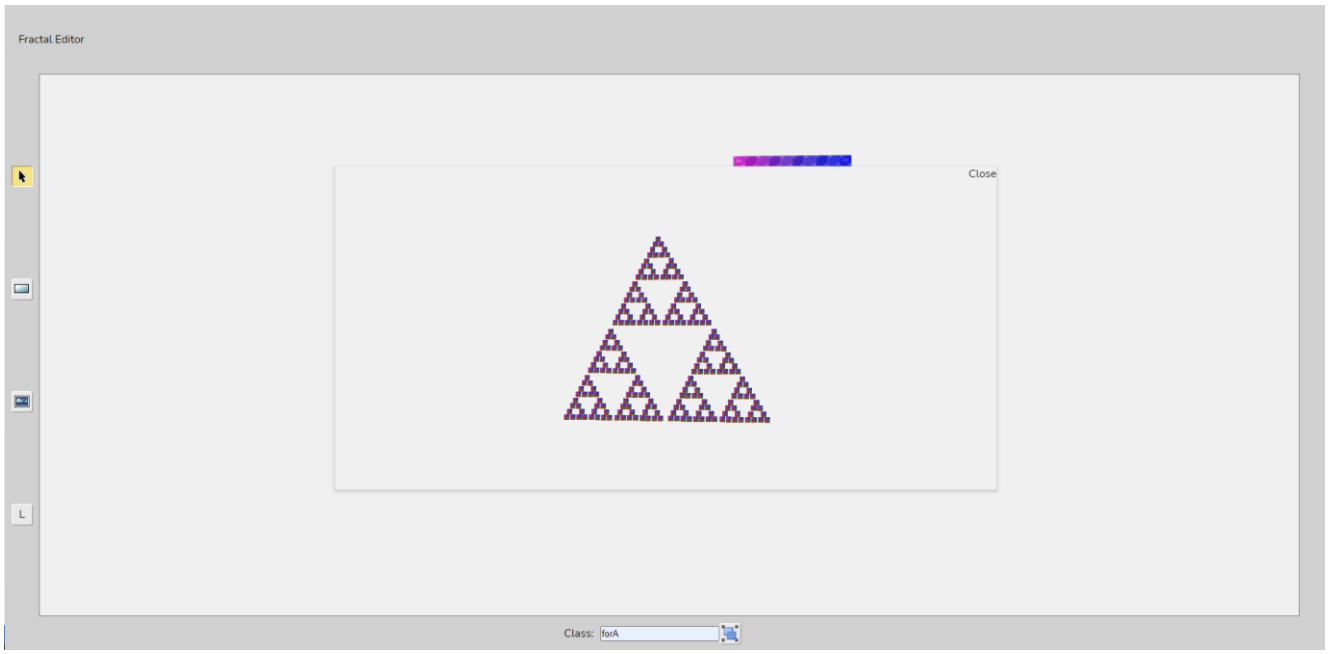

Рисунок 3.14 – Результат генерації відповідно до правил (L=4)

На рисунках 3.15, 3.16 представлено приклад огляду триманого результату побудови фракталу.

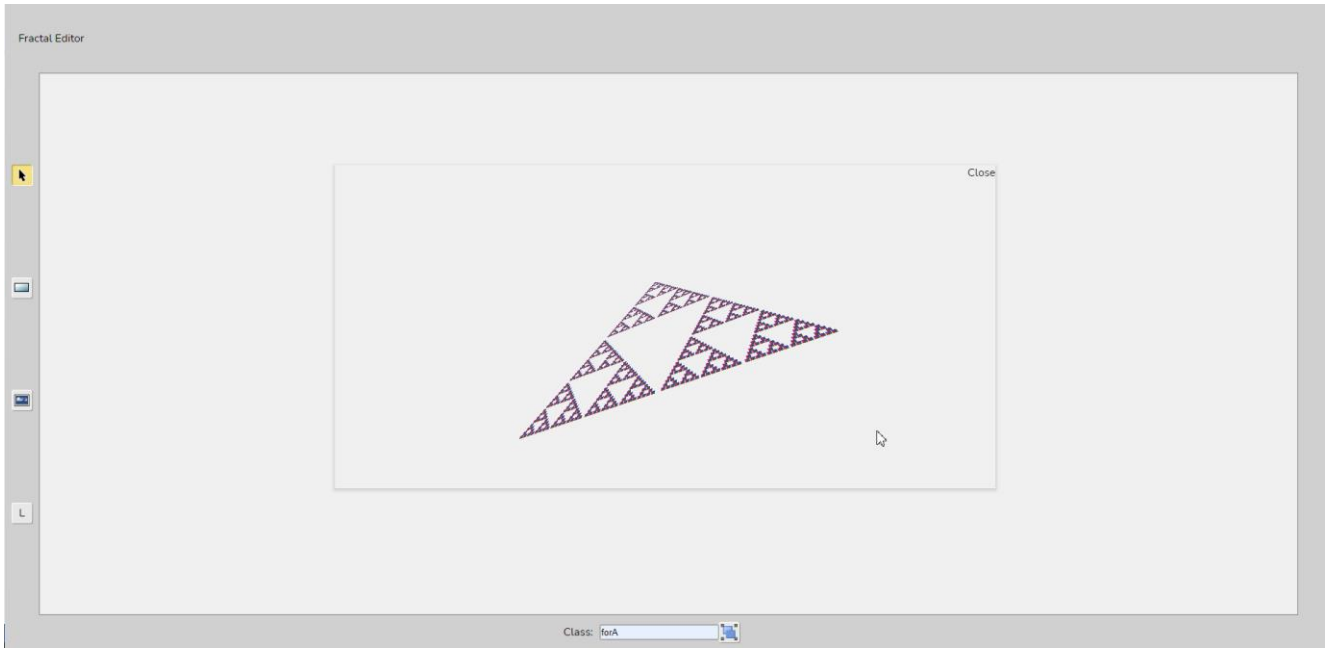

Рисунок 3.15 – Огляд результату генерації

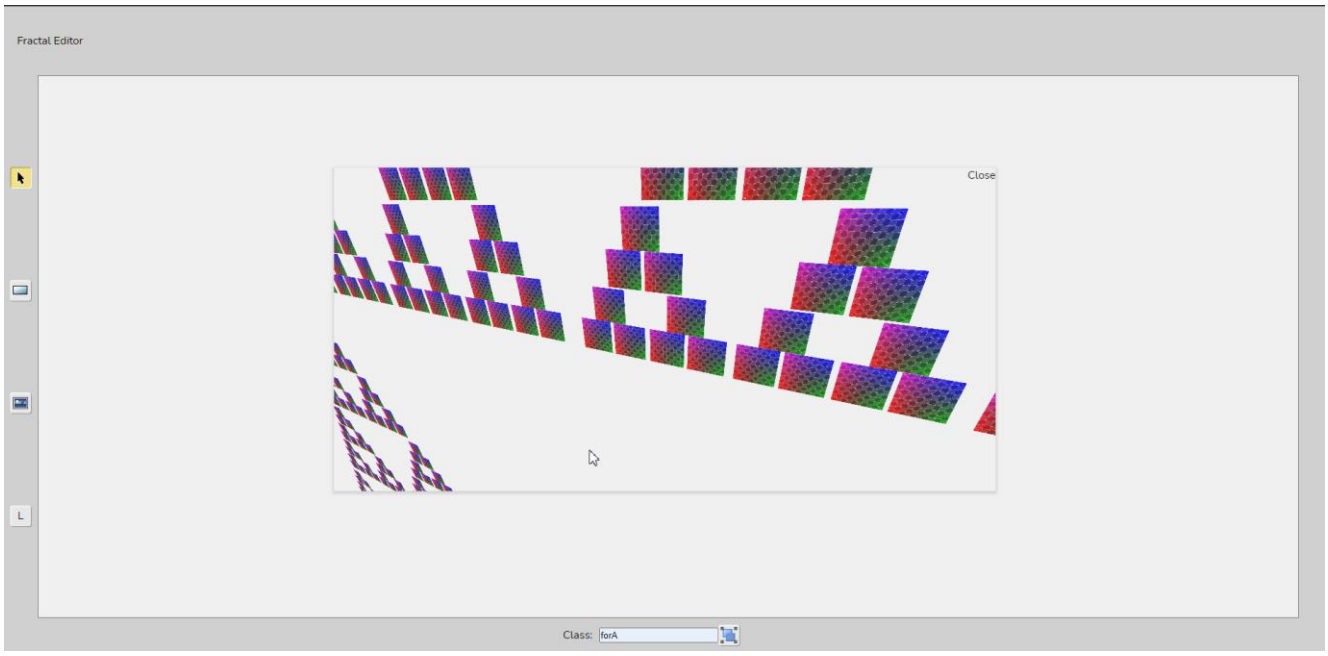

Рисунок 3.16 – Огляд результату генерації (наближення)

Натиснувши на слово Close фрактал закриється і повернеться до початкового екрану. Приклад даної події наведено на рисунку нижче.

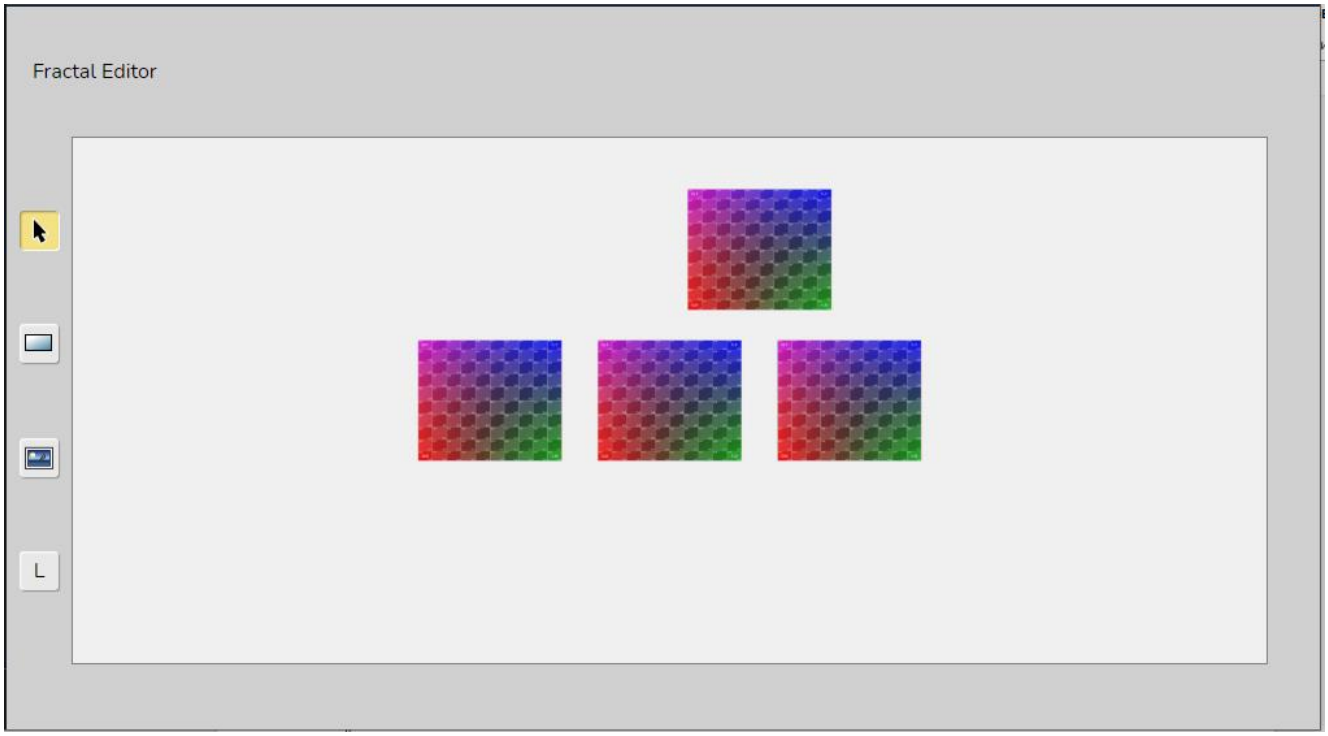

Рисунок 3.17 – Вигляд інтерфейсу після закриття форми з результатами

Висновки до третього розділу

В цьому розділі було розглянуто проектування та розробка програмного забезпечення. Розповідається про інтерфейс користувача, про вхідні данні. Опис базової архітектури, розроблені діаграми та описано реалізацію програми. Згенеровано фрактали N-го покоління.

4 ДОСЛІДЖЕННЯ МЕТОДІВ ПОБУДОВИ ДВОВИМІНИХ ФРАКТАЛІВ У ПРОСТОРІ

4.1. Особливості побудови векторної графіки

Векторна графіка описує зображення з використанням прямих і вигнутих ліній, які називають векторами, а також параметрів, що описують кольори та розташування. Наприклад, зображення деревного листа описується точками, якими проходить лінія, створюючи цим контур листа. Колір листа задається кольором контуру та області усередині цього контуру.

Принцип кодування графічної інформації у векторній графіці принципово відрізняється від растрової. У векторній графіці всі зображення описуються як математичних об'єктів – контурів. Кожен контур є незалежним об'єктом, який можна переміщати, масштабувати, змінювати безліч разів. Усі лінії визначаються початковими точками та формулами, що описують самі лінії. Тому при зміні розміру малюнка пропорції та контури завжди точно витримуються. Векторну графіку часто називають також об'єктно-орієнтованою графікою, оскільки зображення складається з окремих об'єктів – прямих і кривих ліній, замкнутих і розімкнених фігур, прямокутників, еліпсів тощо, кожен з яких має свої характеристики кольору, товщини контуру, стилю лінії. і т.д.

При редагуванні елементів векторної графіки Ви змінюєте параметри прямих і вигнутих ліній, що описують форму цих елементів. Ви можете переносити елементи, змінювати їх розмір, форму і колір, але це не позначиться на якості візуального представлення. Векторна графіка залежить від дозволу, тобто. може бути показана у різноманітних вихідних пристроях з різною роздільною здатністю без втрати якості.

Векторна графіка описує зображення за допомогою математичних формул. За своєю суттю будь-яке зображення можна розкласти на безліч простих об'єктів, як контури, графічні примітиви і т.д. Будь-який такий простий об'єкт складається з

контуру (на малюнку зображений червоним кольором) та заливки (синій колір). Векторне уявлення полягає в описі елементів зображення математичними кривими із зазначенням їх кольорів та заповнюваності (згадайте, коло та коло – різні фігури). Червоний еліпс на білому тлі буде описаний всього двома математичними формулами – прямокутника та еліпса відповідних кольорів, розмірів та розташування. Очевидно, такий опис займе значно менше місця, ніж у першому випадку.

Ще одна перевага – якісне масштабування у будь-який бік. Збільшення чи зменшення об'єктів здійснюється збільшенням чи зменшенням відповідних коефіцієнтів у математичних формулах. На жаль, векторний формат стає невигідним при передачі зображень з великою кількістю відтінків або дрібних деталей (наприклад, фотографій). Адже кожен найменший відблиск у цьому випадку представлятиметься не сукупністю одноколірних точок, а найскладнішою математичною формулою або сукупністю графічних примітивів, кожен з яких є формулою. Це призводить до обтяження файлу.

Крім того, переведення зображення з растрового у векторний формат (наприклад, програмою Adobe Strime Line або Corel OCR-TRACE) призводить до спадкування останнім неможливості коректного масштабування у велику сторону. Від збільшення лінійних розмірів кількість деталей чи відтінків на одиницю площі не стає. Це обмеження накладається роздільною здатністю вступних пристроїв (сканерів, цифрових фотокамер та ін.). Векторне зображення Векторне зображення є сукупністю відрізків кривих ліній, які описують математичними виразами, і кольорових заливок. Інакше кажучи, щоб комп'ютер намалював пряму лінію, потрібні координати двох точок, які з'єднуються найкоротшим шляхом; для дуги задаються координати центру кола та радіус, і т. д.

Таким чином, векторна ілюстрація – це набір геометричних примітивів (найпростіших об'єктів, таких, як лінії, кола, багатогранники тощо) – сплайнів, що використовуються для створення більш складних зображень. Звідси і основна перевага

векторних форматів – компактність одержуваних файлів і висока якість зображень, причому незалежно від роздільної здатності пристрою відображення.

4.1.1. Переваги(в контексті поставленої задачі)

До переваг векторної графіки відноситься те, що вона економна в плані обсягів дискового простору, необхідного для зберігання зображень. Це з тим, що зберігається саме зображення, лише деякі основні дані, використовуючи які, програма щоразу відтворює зображення заново. Крім того, опис колірних характеристик майже не збільшує розмір файлу.

Об'єкти векторної графіки легко трансформуються, причому практично без шкоди якості зображення. Векторні зображення, які не містять растрових об'єктів, займають відносно невеликий обсяг пам'яті комп'ютера. Навіть векторні малюнки, що складаються з тисяч примітивів, потребують пам'яті, обсяг якої не перевищує кількох сотень кілобайтів. Для аналогічного растрового малюнка необхідна у 10-1000 разів більша пам'ять.

Векторні об'єкти задаються за допомогою опису. Тому, щоб змінити розмір векторного малюнка, необхідно виправити його опис. Наприклад, для збільшення або зменшення еліпса достатньо змінити координати лівого верхнього та правого нижнього кутів прямокутника, що обмежує цей еліпс. І знову для малювання об'єкта буде використовуватися максимально можливе число елементів (відеоописів або точок).

Отже, векторні зображення можна легко масштабувати без втрати якості.

4.1.2. Недоліки(в контексті поставленої задачі)

Прямі лінії, кола, еліпси та дуги є основними компонентами векторних малюнків. Тому донедавна векторна графіка використовувалася для побудови креслень, діаграм, графіків, а також створення технічних ілюстрацій.

З розвитком комп'ютерних технологій ситуація дещо змінилася: сьогоднішні векторні зображення за якістю наближаються до реалістичних. Однак, векторна

графіка не дозволяє отримувати зображення фотографічної якості. Справа в тому, що фотографія – мозаїка з дуже складним розподілом кольорів та яскравостей пікселів та уявлення такої мозаїки у вигляді сукупності векторних примітивів – досить складне завдання.

Векторні зображення описуються десятками, інколи ж і тисячами команд. У процесі друку ці команди передаються пристрою виведення (наприклад, лазерний принтер). При цьому може статися так, що на папері зображення виглядатиме зовсім інакше, ніж хотілося користувачеві або взагалі не роздруковується. Справа в тому, що принтери містять власні процесори, які інтерпретують передані їм команди. Тому спочатку потрібно перевірити, чи принтер розуміє векторні команди даного стандарту, надрукувавши якийсь простий векторний малюнок. Після успішного завершення друку можна вже друкувати складне зображення. Якщо ж принтер не може розпізнати будь-який примітив, слід замінити його іншим – схожим, зрозумілим принтеру.

Таким чином, векторні зображення іноді не друкуються або виглядають на папері не так, як хотілося б.

4.2. Особливості побудови растрової графіки

Растрові зображення використовують бітові карти для зберігання інформації. Це означає, що великий файл потребує великого растрового зображення. Чим більше зображення, тим більше місця на диску займе файл образу. Наприклад, зображення розміром 640 x 480 вимагає збереження інформації для 307 200 пікселів, тоді як зображення розміром 3072 x 2048 (з цифрової камери 6,3 мегапікселя) має зберігати інформацію для величезних 6 291 456 пікселів. Ми використовуємо алгоритми, які стискають зображення, щоб зменшити ці розміри файлів. Формати зображень, такі як jpeg і gif, є поширеними форматами стиснутих зображень. Зменшити ці зображення легко, але збільшення растрового зображення робить його піксельним або просто

розмитим. Тому для зображень, які потрібно масштабувати до різних розмірів, ми використовуємо векторну графіку.

4.2.1. Переваги растрової графіки (в контексті поставленої задачі)

Трасування зображень у обчислювальній техніці можна віднести до векторизації, і це просто перетворення растрових зображень у векторні зображення. Цікавим застосуванням векторизації є оновлення зображень і відновлення роботи. Векторизація може бути використана для отримання інформації, яку ми втратили. Paint в Microsoft Windows створює вихідний файл растрового зображення. У Paint легко помітити нерівні лінії. При такому перетворенні розмір зображення різко зменшується. Як наслідок, точне перетворення неможливе в цьому сценарії. Через різні апроксимації та редагування, які виконуються в процесі перетворення, перетворені зображення не мають гарної якості.

4.2.2. Недоліки растрової графіки (в контексті поставленої задачі)

Максимальне можливе наближення зображення при використанні растової графіки напряму залежить від технічних можливостей робочої машини на якій виконується побудова фрактальних зображень. Тобто можливості дослідження побудованих фрактальних зображень дещо обмежені на відміну від наближення при використанні векторної графіки.

Принтери та пристрої відображення є растровими пристроями. В результаті нам потрібно перетворити векторні зображення в растровий формат, перш ніж їх можна буде використовувати, тобто відобразити або роздрукувати. Необхідна роздільна здатність відіграє важливу роль у визначенні розміру створюваного растрового файлу. Тут важливо зазначити, що розмір векторного зображення, яке потрібно конвертувати, завжди залишається незмінним. Зручно конвертувати векторний файл у ряд форматів растрових/растрових файлів, але йти протилежним шляхом важче. (бо інколи нам потрібно редагувати зображення під час перетворення з растрового у векторний).

4.3. Вибір типу графіки та пояснення вибору

В роботі було обрано растровий тип графіки.

Для роботи з побудовою фракталів у просторі вибір типу використовуваної графіки не міг лежати дуже гостро, тому що використання векторної графіки (щонайменше на даних момент) не дає ніяких можливостей для побудови плоских фракталів у тривимірному просторі.

Хоча існують програми, що забезпечують трасування растру у вектор, але коректно, скажімо, перевести повнокольорову фотографію людського обличчя у векторне зображення вони не можуть. У будь-якому випадку отриманий вектор не зможе передати всю тонкість та глибину фарб повнокольорового растрового зображення.

Навіть якщо при перекладі у вектор встановити налаштування, що точно передають дрібні деталі і градації кольору, все одно при неосяжному розмірі векторного файлу результат буде однаково не ідеальним. Положення змінюється, коли вектор експортується в растрове зображення. Тут майже немає меж для величини дозволу растру, і при цьому він залишається однаково якісним. Тобто. векторне зображення будується на примітивних графічних об'єктах, побудованих із векторів: лінія, прямокутник, коло, дуга, замкнута лінія тощо.

Наприклад, основою більшості складних 3D-фігур є трикутник, з безлічі якого складається вся об'ємна фігура. Група примітивів і є векторний малюнок. Нині дуже поширена тривимірна графіка (3D). На основі тривимірних векторних редакторів будуються найскладніші сцени. Цю область безперечно не можна замінити ні чим іншим. Як би талановиті та посидючі ви не були, намалювати пензлем растрового редактора зображення тривимірного об'єкта неможливо. Є чимало людей, які намагаються це спростовувати, але це не тема для розмови. Просто потрібно цінувати і розуміти, що різні технології комп'ютерної графіки спеціалізовані в різних напрямках і

несмачно змішувати їх, або замінювати одну іншу – дурна впертість. А ось грамотно комбінувати їх можна і потрібно.

Саме формування комп'ютерних об'єктів, регулювання колірного балансу, створення будь-яких колірних та об'ємних ефектів роблять зображення яскравим та неповторним. Сцена 3D-моделей будується на пакетах тривимірного моделювання і надалі може візуалізуватися з будь-яких точок перегляду 2D-зображення. При цьому є можливість змін освітлення, форм об'єктів, перспективних деформацій, регулювання параметрів матеріалів та атмосферних ефектів комп'ютерної тривимірної сцени. Можна створити не тільки тривимірні стандартні об'єкти – куб, чарка і т.п., а й складніші об'єкти, скажімо, звірят, а також різних персонажів і т.п.

Висновки до четвертого розділу

Усі комп'ютерні зображення, всі формати для їх зберігання і всі програми для їх обробки поділяються на два великі класи – векторні та растрові, – що відрізняються, перш за все, рівнем абстракції, застосованої до зображення. Можна сказати, що якщо векторна графіка намагається імітувати сприйняття зображень людиною, то растровий формат зберігає графіку в тому вигляді, в якому вона найлегше перетравлюється комп'ютером. Відповідно, векторна графіка здебільшого створюється людиною з нуля прямо у векторному редакторі, а спроби генерувати її автоматично рідко коли призводять до задовільного результату. І навпаки, головний постачальник растрових зображень – фотографії, тобто. у суттєвій своїй частині автоматичний процес з результатами, що легко оцифровуються.

Векторне зображення складається з об'єктів – геометричних форм, складених з прямих, дуг кола та кривих Безьє. У всіх векторних форматах об'єкти можуть варіювати товщину та колір контуру, а замкнуті об'єкти – ще й колір заливки. Об'єкти можуть накладатися, частково або повністю затуляючи один одного.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1. Вимоги до безпеки при виконанні робіт на робочому місці

Вимоги стосовно умов праці трактуються законодавством України, а саме Законом України «Про охорону праці» від 2002р [34].

Робочі місця працівників з екранними пристроями мають бути спроектовані і мати такі розміри, щоб працівники мали простір зміни робочого становища і рухів.

Для забезпечення безпеки та захисту здоров'я працівників все випромінювання від екранних пристроїв має бути зведене до гранично допустимого рівня (вплив на людину факторів середовища – шуму, вібрації, забруднювачів, температури тощо, що не викликає соматичних чи психічних розладів, а також змін стану здоров'я, працездатності, поведінки, що виходять за межі пристосувальних реакцій) з точки зору безпеки та охорони здоров'я працівників [35].

Організація робочого місця працівника з екранними пристроями повинна забезпечувати відповідність всіх елементів робочого місця та їх розташування ергономічним, антропологічним, психофізіологічним вимогам, а також характером виконуваних робіт.

Висвітлення робочого місця працівника з екранними пристроями має створювати відповідний контраст між екраном та довкіллям (з урахуванням виду роботи) та відповідати вимогам ДСанПІН 3.3.2.007-98 [36].

Робочий стіл або робоча поверхня повинні бути достатнього розміру та мати поверхню з низькою відбивною здатністю, допускати гнучкість при розміщенні екрана, клавіатури, документів та відповідного обладнання.

Робоче крісло має бути стійким і дозволяти працівникові з екранними пристроями легко рухатися та займати зручне положення.

Сидіння має регулюватися по висоті, спинка сидіння – як по висоті, так і по нахилу. Слід передбачати підніжку для тих, кому це потрібно для зручності [37].

Мінімальні вимоги безпеки під час роботи з екранними пристроями:

- щодня перед початком роботи необхідно очищати екранні пристрої від пилу та інших забруднень;
- після завершення роботи екранні пристрої слід відключати від електричної мережі;
- у разі аварійної ситуації необхідно негайно відключити пристрій від електричної мережі.

Не допускається:

- виконувати технічне обслуговування, ремонт та налагодження екранних пристроїв безпосередньо на робочому місці працівника під час роботи з екранними пристроями [38];
- відключати захисні пристрої, самовільно проводити зміни у конструкції та складі екранних пристроїв або їх технічне налагодження;
- працювати з екранними пристроями, де під час роботи виникають нехарактерні сигнали, нестабільне зображення на екрані та інші несправності.

При виконанні робіт операторського типу, пов'язаних з нервово-емоційною напругою, у приміщеннях при роботі з екранними пристроями, на пультах та постах управління технологічними процесами та в інших приміщеннях повинні дотримуватися оптимальних умов мікроклімату відповідно до вимог ДСН 3.3.6.042-99.

Мінімальні вимоги безпеки до екранних пристроїв:

- все випромінювання, за винятком видимої частини електромагнітного спектру, має бути зведене до незначного рівня з погляду безпеки та охорони здоров'я працівників;
- символи на екранних пристроях мають бути чіткими, відповідного розміру. Між символами та рядками символів має бути належна відстань;
- зображення на екрані має бути стабільним, без миготіння або інших видів нестабільності;

- яскравість та/або контрастність символів повинні легко регулюватися працівником під час роботи з екранними пристроями, а також швидко адаптуватися до навколишніх умов;
- вибираючи екрани, слід віддавати перевагу таким екранам, які легко та вільно обертаються та нахиляються відповідно до потреби працівника;
- за потреби можна використовувати окрему підставку або регульований стіл для розміщення екрана;
- екран не повинен відображати світло, щоб не викликати дискомфорт у працівника під час роботи з екранними пристроями;
- вибираючи клавіатуру, слід віддавати перевагу такій клавіатурі, яка відкидається і є автономною (відокремленою від екрану), щоб працівник міг вибрати зручну робочу позу та уникнути втоми рук (кисті та верхньої частини руки);
- поверхня клавіатури має бути матовою, щоб уникнути відображення. Розташування клавіш і клавіші повинні полегшувати роботу з клавіатурою. Позначення клавіш має бути досить контрастним та розбірливим;
- обладнання, що входить до робочої станції, не повинно виділяти надлишкового тепла, що може завдати незручності працівникам під час роботи з екранними пристроями;
- при розробці, виборі, замовленні та модифікації програмного забезпечення, а також при розробці завдань, що передбачають використання обладнання з екранними пристроями, роботодавець повинен керуватися таким програмним забезпеченням, яке відповідає розв'язуваним завданням і просто у використанні, а де необхідно адаптованим до рівня знань та досвіду працівника.

5.2. Шкідливі виробничі фактори на робочому місці

5.2.1. Мікроклімат приміщення

Мікроклімат виробничих приміщень з робочими місцями працівників з екранними пристроями повинен підтримуватися на постійному рівні та відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99, затверджених постановою Головного державного санітарного лікаря України від 0.12.2004 № 2.2.1999 -99) [39].

Показниками, що характеризують метеорологічні умови у закритих виробничих приміщеннях (мікроклімат), є:

- температура повітря;
- відносна вологість повітря;
- швидкість руху повітря;
- інтенсивність теплового виромінення.

Оптимальні показники мікроклімату поширюються всю робочу зону виробничих приміщень без розмежування робочих місць на постійні і непостійні. Допустимі показники встановлюються на постійних та непостійних робочих місцях робочої зони. Оптимальні та допустимі показники температури, відносної вологості та швидкості руху повітря у робочій зоні виробничих приміщень повинні відповідати величинам.

Допустимі величини показників мікроклімату встановлюються у випадках, коли з технологічних вимог виробництва, технічних і економічних причин ще неможливо забезпечити оптимальні норми.

У кабінах, на пультах та постах управління технологічними процесами, в залах обчислювальної техніки, а також в інших приміщеннях при виконанні робіт операторського типу, пов'язаних з нервово-емоційною напругою, повинні дотримуватися оптимальних величин температури повітря (22 – 24 °С), його відносної

вологості (60 – 40%) та швидкості руху (не більше 0,1 м/с). Перелік інших виробничих приміщень, у яких повинні дотримуватись оптимальні норми мікроклімату, визначається галузевою документацією, погодженою з органами санітарного нагляду в установленому порядку.

При забезпеченні оптимальних показників мікроклімату температура внутрішніх поверхонь, що огороджують робочу зону конструкцій (стін, підлоги, стелі) або пристроїв (екранів тощо), а також температура зовнішніх поверхонь технологічного обладнання або його пристроїв не повинні виходити більш ніж на 2 градуси. С за межі оптимальних величин температури повітря, встановлених для окремих категорій робіт. При температурі внутрішніх поверхонь огороджувальних конструкцій нижче або вище оптимальних величин температури повітря робочі місця повинні бути віддалені від них на відстань не менше 1 м. Перепади температури повітря по висоті та горизонталі робочої зони її зміна протягом зміни не повинні виходити за межі оптимальних температур, для окремих категорій робіт.

У холодний період року необхідно передбачати заходи захисту робочих місць від радіаційного охолодження від засклених поверхонь віконних отворів, у теплий період – від потрапляння прямих сонячних променів.

При забезпеченні допустимих величин показників мікроклімату температура внутрішніх поверхонь, що огороджують робочу зону конструкцій (стін, підлоги, стелі) або пристроїв (екранів тощо), повинна виходити межі допустимих величин температури повітря, встановлених для окремих категорій робіт. Перепад температури повітря за висотою робочої зони за всіх категорій робіт допускається до 3 °С.

Зміни температури повітря горизонталлю робочої зони, а також протягом зміни допускаються до 4 °С – при легких роботах, до 5 °С – при роботах середньої тяжкості та до 6 °С – при важких роботах, при цьому абсолютні значення температури повітря, виміряної на різній висоті та в різних ділянках приміщень протягом зміни, не повинні виходити за межі допустимих величин.

У холодний період року необхідно передбачати заходи захисту робочих місць від радіаційного охолодження від засклених поверхонь віконних отворів, у теплий період – від потрапляння прямих сонячних променів.

Вимоги до температури внутрішніх поверхонь огороджувальних конструкцій та пристроїв не поширюються на загальні та місцеві системи опалення та охолодження приміщень та робочих місць.

Інтенсивність теплового опромінення працюючих від нагрітих поверхонь технологічного обладнання, освітлювальних приладів, інсоляції на постійних та непостійних робочих місцях не повинна перевищувати 35 Вт/кв. м при опроміненні 50% та більше поверхні тіла, 70 Вт/кв. м при величині поверхні, що опромінюється від 25 до 50% і 100 Вт/кв. м – при опроміненні трохи більше 25% поверхні тіла.

Інтенсивність теплового опромінення працюючих від відкритих джерел (нагрітий метал, скло, відкрите полум'я) не повинна перевищувати 140 Вт/кв. м при опроміненні не більше 25% поверхні тіла та обов'язковому використанні засобів індивідуального захисту, у тому числі засобів захисту обличчя та очей.

При цьому на постійних робочих місцях температура повітря не повинна перевищувати верхні межі оптимальних значень для теплої пори року; на непостійних робочих місцях – верхні межі допустимих значень для постійних робочих місць.

У четвертій будівельно-кліматичній зоні, згідно з СНіП 2.01-01-82 "Будівельна кліматологія та геофізика", у виробничих приміщеннях за дотримання вимог щодо попередження перегрівання допустимих, що працюють на верхній кордон.

5.2.2. Освітлення робочого місця

Нормативними документами передбачені норми освітленості під час роботи з комп'ютером. Все можна взяти зі СНіП ДБН В.2.5-28: Для офісних працівників достатньо 300 люксів.

Якщо організації працюють конструктора, то норматив освітленості 500 лк. Для залів, де проводяться наради – 200 лк. Для архівів – 75 лк.

Діючі норми містять такі відомості: Коефіцієнт природного освітлення не повинен бути менше 1,5 – 1,2%.

Світло повинне падати на робочу поверхню зліва.

За рахунок штучного підсвічування регулюється рівномірність освітлення. Потік світла для робочого столу має сягнути від 300 до 500 лк. Щоб це забезпечити, достатньо вирішити проблему застосуванням місцевого освітлення. До такого освітлення теж висуваються свої вимоги:

- від екрана не повинно відбивати відблисків;
- потік світла має бути 300 лк;
- близькість має бути обмежена.

Показник яскравості повинен перевищувати 200 кд на квадратний метр. Необхідно пам'ятати, що поверхня, що відображається, обмежується, але при цьому застосовуються настільні лампи.

Дані параметри належать до застосування штучного освітлення. Інші вимоги: Показник яскравості відблисків на моніторі щонайменше 40 кд на квадратний метр. Осліплення для світильників – 20 одиниць.

Для адміністративно-побутових приміщень достатньо 40 одиниць засліплення. Показник яскравості між поверхнями визначається відносинами 3 до 1, або 5 до 1.

Між поверхнями обладнання та стін цей показник – 10:1. Щоб організувати оптимальні умови праці, достатньо керуватися цими нормативами. Тільки тоді можна досягти непоганих результатів та зберегти зір.

5.2.3. Шум та вібрація

Розрізняють два види нормування виробничого шуму: гігієнічне нормування та технічне [40].

Під гігієнічним нормуванням розуміють обмеження емісії шуму, тобто. обмеження рівнів шуму, що впливає людину, що у зоні дії джерел шуму. Мета гігієнічного нормування – обґрунтування допустимих рівнів та комплексу гігієнічних вимог, що забезпечують попередження функціональних розладів та захворювань.

Предметом технічного нормування є обмеження інтенсивності випромінювання джерел шуму з умов забезпечення допустимих рівнів шуму робочих місцях. Мета технічного нормування – надання можливості проектувальникам виробничих приміщень та споживачам машинобудівної продукції підбирати машини та обладнання з необхідними акустичними характеристиками, а творцям нового обладнання ще на стадії проектування визначати необхідність проведення технічних та організаційних заходів боротьби з шумом.

За тимчасовими характеристиками шум слід поділяти на:

постійний, рівень звуку якого за 8-годинний робочий день (робочу зміну) змінюється в часі не більше ніж на 5 ДБА при вимірах на часовій характеристиці за ДБН В.1.1-31:2013;

непостійний, рівень звуку якого за 8-годинний робочий день (робочу зміну) змінюється у часі більш ніж на 5 ДБА при вимірах на часовій характеристиці за ДБН В.1.1-31:2013.

Непостійний шум поділяють на:

- коливається у часі, рівень звуку якого безперервно змінюється у часі;
- переривчастий, рівень звуку якого поступово змінюється (на 5 ДБА і більше), причому тривалість інтервалів, протягом яких рівень залишається постійним, становить 1 сек. і більше;
- імпульсний, що складається з одного або декількох звукових сигналів, кожен тривалістю менше 1 сек.

5.3. Дії працівників в надзвичайних ситуаціях

5.3.1. Дії у разі пожежі.

Кожен працівник при виявленні вогнища загоряння або ознак горіння (задимлення, запах гару, підвищення температури тощо) повинен:

негайно повідомити про це телефоном «101» (для мобільного зв'язку). У цьому назвати найменування об'єкта, місце вибуху, пожежі, і навіть своє прізвище; вжити заходів щодо евакуації людей, гасіння пожежі та збереження матеріальних цінностей.

Вимоги щодо використання первинних засобів пожежогасіння:

Вуглекислотні вогнегасники (ОУ-2, ОУ-3, ОУ-5, ОУ-6, ОУ-7 тощо)

призначені для гасіння загоряння різних горючих речовин, за винятком тих, горіння яких відбувається без доступу повітря, а також застосовуються для гасіння електроустановок, що знаходяться під напругою до 1000В. Вогнегасна речовина – двоокис вуглецю.

Для приведення в дію вуглекислотних вогнегасників необхідно розтруб направити на палаючий предмет, зірвати пломбу, висмикнути чеку, натиснути на важіль (або повернути маховик вентиля вліво до відмови), направити струмінь на полум'я. Тримати вогнегасник вертикально, перевертати його не потрібно.

Щоб уникнути обморожування, не торкатися металевої частини розтруба оголеними частинами тіла. При гасінні електроустановок, що знаходяться під напругою, не допускається підводити до них розтруб ближче 1м.

Внутрішні пожежні крани (ПК) призначені для подачі води при гасінні твердих матеріалів і горючих рідин. Внутрішній ПК вводиться в роботу двома працівниками: один прокладає рукав і тримає напоготові пожежний ствол для подачі води у вогнище горіння, другий – перевіряє приєднання пожежного рукава ПК та відкриває вентиль для надходження води.

Асбестове полотно, повсть (кошма) використовуються для гасіння невеликих вогнищ загоряння будь-яких речовин та матеріалів, горіння яких не може відбуватися без доступу повітря. Осередок загоряння накривається азбестовим або повстяним полотном для припинення повітря.

Пісок застосовується для механічного збивання полум'я та ізоляції палаючого або тліючого матеріалу від доступу повітря. Подається в осередок пожежі лопатою або совком.

5.3.2. Дії у разі ураження електричним струмом

Відповідно до порядку надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою затвердженому наказом Міністерства охорони здоров’я України №398 [41] послідовність дій надання первинної допомоги повинна бути наступна:

1) переконатися у відсутності небезпеки;
2) при перебуванні постраждалого під дією електричного струму, спробувати, при можливості, припинити його дію;
3) провести огляд постраждалого;
4) викликати швидку допомогу;
5) якщо постраждалий не дихає, розпочати процес серцево-легеневої реанімації;
6) якщо постраждалий дихає, але без свідомості, забезпечити йому стабільне положення;
7) накласти чисті повязки на місця опіку;
8) забезпечити потерпілому постійний нагляд до прибуття швидкої допомоги.

5.3.3. Порядок надання домедичної допомоги

Надання першої допомоги при різних травмах і станах – це невідкладний порядок дій до надання медичної допомоги, який спрямований на усунення фактора,

що несе загрозу життю або здоров'ю потерпілого, на зняття болю та мінімізації ризику ускладнень [42].

У разі виникнення рани: якщо рана сильно кровоточить, спочатку треба зупинити кровотечу. Рану зверху накрити чистою марлею, перев'язати всю рану бинтом. Якщо у розпорядженні є настойка йоду, спирт етиловий, то шкіру навколо рани спочатку двічі або тричі протерти марлею або ватою, змоченою цим розчином.

У разі виникнення забитих місць: необхідно накласти пов'язку, що давить, холод (хустку, змочену холодною водою, сніг або лід у целофановому пакеті) на забите місце.

У разі розтягнення зв'язок: необхідно накласти тугу пов'язку, холод.

У разі вивиху: створити кінцівці максимальний спокій. Вправляти вивихи повинен лише медичний працівник

У разі перелому: накласти шину (наприклад, з дошки, фанери, палиць, картону), зафіксувати два найближчих суглоби. При відкритих переломах перед тим як накласти шину, треба накласти стерильну пов'язку на рану. Навіть за підозри на перелом фіксація кінцівки обов'язкова.

У разі опіку: при термічному опіку треба усунути причину, що спричинила опік, промити місце опіку холодною водою (при опіку без порушення цілісності опікових бульбашок), накласти стерильну пов'язку, при можливості покласти сніг, лід чи інший холод на 15-20 хвилин.

У разі хімічного опіку необхідно промити місце опіку холодною водою або молоком, сечею, мильною водою, слабким розчином питної соди.

У разі відмороження необхідне повільне та поступове зігрівання потерпілого (перенесення в тепле приміщення), накладення пов'язок, тепле питво (чай, кава), примус до руху.

У разі електротравм: необхідно дотримуватися заходів особистої безпеки, припинити дію струму на організм (наприклад вимкнути рубильник, вивернути

запобіжні пробки на щиті, відтягнути дріт сухою дерев'яною палицею, сухою мотузкою або відтягнути потерпілого, використовуючи при цьому діелектричні рукавички або підручні ізолюючі засоби: суху мотузку , дошку, прогумований плащ, гумовий килимок) [43]. На область опіку накласти суху пов'язку, забезпечити постраждалому повний спокій та викликати лікаря.

При відсутності у потерпілого дихання та пульсу необхідно розстебнути одяг, почати штучну вентиляцію легень та зовнішній масаж серця до відновлення самостійного дихання та серцебиття.

У разі непритомності необхідно укласти потерпілій в горизонтальне положення, підняти ноги, розстебнути одяг, що стискує, забезпечити доступ свіжого повітря, обличчя оббризкати холодною водою, дати понюхати нашатирний спирт або оцет на ватці, натерти цими засобами віскі, натиснути больову точку під носом.

При тепловому (сонячному) ударі перенести потерпілого в тінь, розстебнути одяг і покласти з піднятою головою, накласти на голову холодний компрес, напоїти холодною водою. Застосувати холодне обгортання (наприклад, мокрим простирадлом). За необхідності провести зовнішній масаж серця та штучну вентиляцію легень.

5.3.4. Пожежна безпека

Перелік основних журналів з питань пожежної безпеки для підприємств, установ та організацій України [44]:

- журнал обліку вогнегасників на об'єкті;
- журнал реєстрації інструктажів з питань пожежної безпеки;
- журнал обслуговування установок пожежної автоматики;
- журнал обліку технічного обслуговування внутрішніх пожежних кранів (у разі наявності внутрішнього протипожежного водопостачання).

Чинні нормативно-правові акти з питань пожежної безпеки обов'язковість нумерації, прошнурування та скріплення печаткою журналів, що регламентують Вашу діяльність у галузі пожежної безпеки, не визначають.

Виходячи з наданих характеристик вашої організації, а саме орендований офіс у бізнес-центрі, влаштовувати у приміщенні стенд із пожежної безпеки не обов'язково. Достатньо вивісити на видному місці інструкцію щодо заходів пожежної безпеки (пункт 3.5 Правил пожежної безпеки в Україні).

Вогнегасники слід встановлювати у легкодоступних та помітних місцях (коридорах, біля входів або виходів з приміщень), а також у пожежонебезпечних місцях, де найімовірніша поява вогнищ пожежі. При цьому необхідно забезпечити їх захист від потрапляння прямих сонячних променів та безпосередньої (без загороджувальних щитків) дії опалювальних та нагрівальних приладів.

Вибір типу та необхідна кількість вогнегасників визначається відповідно до Типових норм належності вогнегасників (далі — Типові норми), затверджених наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 2 квітня 2004 р. № 151.

Відповідно до пункту 3.8 Типових норм, громадські та адміністративно-побутові будівлі на кожному поверсі повинні мати не менше двох переносних (порошкових, водопінних або водних) вогнегасників із масою заряду вогнегасної речовини 5 кг та більше. Крім того, слід передбачати по одному вуглекислотному вогнегаснику з величиною заряду вогнегасної речовини 3 кг і більше:

- на 20 м2 площі підлоги в офісних приміщеннях з ПОЕМ, комор, електрощитових, вентиляційних камер і технічних приміщеннях;
- на 50 м2 площі підлоги приміщень архівів, машзалів, бібліотек, музеїв.

Приміщення, в яких розміщені ПЕОМ, слід оснащувати переносними вуглекислотними вогнегасниками з розрахунку один вогнегасник ВВК-1,4 (старе позначення ОУ-2) ВВК-2 (старе позначення ОУ-3) або один ВВПА-400 на три ПЕОМ,

але не менше одного вогнегасника зазначених типів на приміщення (п. 3.10 Типових норм).

Ця вимога стосується також будівель, споруд та приміщень, що обладнані будь-якими типами установок пожежогасіння, пожежної сигналізації або внутрішніми пожежними кранами (п. 6.4.8 Правил пожежної безпеки в Україні).

Відповідно до п. 6.4.23 Правил пожежної безпеки в Україні, відповідальними особами за своєчасне та повне оснащення об'єктів вогнегасниками та іншими засобами пожежогасіння, забезпечення їх технічного обслуговування, навчання працівників за правилами користування вогнегасниками є власники цих об'єктів (або орендарі згідно з договором оренди).

Висновки до п’ятого розділу

В цьому розділі ми ознайомилися з Охороною праці та безпекою в надзвичайних ситуаціях. Дізналися, про вимоги до безпеки при виконання робіт на робочому місці. Розібрали шкідливі виробничі фактори. Яке має бути освітлення на робочому місці. Що потрібно робити у надзвичайній ситуації та яку медичну допомогу потрібно надати. Також ознайомилися з правилами пожежної безпеки.

ВИСНОВКИ

Розроблено сімейство параметричних конструкторів формування мультисимвольних фракталів та пов'язане з ним сімейство параметричних конструкторів-перетворювачів мультисимвольних у лінійні плоскі геометричні фрактали. Обидва сімейства базуються на ідеях L-систем. Процес формування фракталів з різною елементною базою побудований таким чином, що на основі розбору (аналізу) одних з них формуються інші. Очевидно встановлюється зв'язок між ними. Автором розроблено програму, що реалізує представлені в даній роботі сімейства параметричних конструкторів.

В роботі отримав розвиток конструктивно-продукційний підхід формалізації конструкцій і конструктивних процесів, що допускають їх визначення і моделювання. Представлений у статті підхід до моделювання процесів на основі узагальненого конструктора дозволяє ефективно та просто будувати детерміновані та стохастичні лінійні геометричні фрактали на базі продукційних L – систем. Запропонована варіативність на основі сімейства параметричних конструкторів дозволяє формалізувати взаємнооднозначну відповідність між конструкціями та різною природою, що відкриває нові можливості їх вивчення. Розроблений формалізм, разом з іншими роботами з конструктивно-продукційного моделювання, дозволяє створювати більш універсальні програмні засоби конструювання та оптимізації конструкцій та конструктивних процесів різної природи.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Мандельброт Б. Б. Фрактали та хаос. Безліч Мандельброта та інші дива. – М., НДЦ "Регулярна та хаотична динаміка", 2009. – 392 с. ISBN: 978-5-93972-772-3
2. Б. Мандельброт Фрактальна геометрія природи. – Москва: Інститут комп'ютерних досліджень, 2002. – 656 с.
3. Річард М. Кроновер Фрактали та хаос у динамічних системах. – М., Постмаркет, 2000. – 352 с.
4. Шредер М. Фрактали, хаос, статечні закони. Мініатюри із нескінченного раю. – Іжевськ: НДЦ «Регулярна та хаотична динаміка». 2001. – 528 с.
5. М.Газалі. Гномон. Від фараонів до фракталів. -Москва-Іжевськ: Інститут комп'ютерних досліджень, 2002. – 272 с.
6. Х.-О. Пайтген, П. Х. Рітхер Краса фракталів. – М. Світ, 1993. – 176 с.
7. Божокін СВ., Паршин В.А. Фрактали та мультифрактали. – Іжевськ: НДЦ «Регулярна та хаотична динаміка», 2001. – 128 с.
8. Є.Федер Фрактали. – М., Світ, 1991. – 261 с.
9. Морозов А.Д. Введення у теорію фракталів. – Москва-Іжевськ: Інститут комп'ютерних досліджень, 2002. – 160 с.
10. е. Петерс Фрактальний аналіз фінансових ринків: Застосування теорії Хаосу в інвестиціях та економіці. М: Інтернет-тренд, 2004 – 304 с
11. Могилевський Е.І. Фрактали на сонці. – М.: ФІЗМАТЛІТ, 2001. – 152 с. – ISBN 5-9221-0179-X.
12. В.В. Ісаєва, Ю.А. Каретін, А.В. Чернишов, Д.Ю. Шкурат Фрактали і хаос в біологічному морфогенезі. – Владивосток, 2004. – 128 с.
13. Ultra Fractal [електронний ресурс] Режим доступу: https://www.ultrafractal.com
14. XenoDream [електронний ресурс] Режим доступу: https://www.xenodream.com
15. Jux [електронний ресурс] Режим доступу: https://studiojux.com

16. Fractal Explorer [електронний ресурс] Режим доступу: http://zuxcel.com/fractal-explorer/
17. Xaos [електронний ресурс] Режим доступу: https://xaos-project.github.io
18. Apophysis [електронний ресурс] Режим доступу: https://sourceforge.net/projects/apophysis/
19. Fractal Science Kit [електронний ресурс] Режим доступу: http://www.fractalsciencekit.com
20. ChaosPro [електронний ресурс] Режим доступу: http://www.chaospro.de/download.php
21. Мандельброт Б. Б. Фрактали та хаос. Безліч Мандельброта та інші дива. – М., НДЦ "Регулярна та хаотична динаміка", 2009. – 392 с. ISBN: 978-5-93972-772-3
22. Б. Мандельброт Фрактальна геометрія природи. – Москва: Інститут комп'ютерних досліджень, 2002. – 656 с.
23. Річард М. Кроновер Фрактали та хаос у динамічних системах. – М., Постмаркет, 2000. – 352 с.
24. Шредер М. Фрактали, хаос, статечні закони. Мініатюри із нескінченного раю. – Іжевськ: НДЦ «Регулярна та хаотична динаміка». 2001. – 528 с.
25. Мандельброт Б. Фрактальная геометрия природы. Москва: Институт компьютерных исследований, 2002. 656 с.
26. Кроновер Р. Фрактал и хаос в динамических системах. Москва: Техносфера, 2006. 484 с.
27. Falconer K. Fractal Geometry: Mathematical Foundations and Applications. John Wiley & Sons. 1999. – 288 с.
28. Божокин С. В., Паршин Д. А. Фракталы и мультифракталы. Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001. – 128 с.
29. Морозов А. Д. Введение в теорию фракталов. Москва-Ижевск: Институт компьютерных исследований, 2002. – 160 с.

30. Рысцов И. К. Аффинные автоматы и классические фракталы. Кибернетика и системный анализ. 2018. №1. С. 13-23.
31. Lindenmayer A. Mathematical models for cellular interaction in development. J. Theoret. Biology. 1968. №18. P. 280-315.
32. Шинкаренко В. И., Ильман В. М. Конструктивно-продукционные структуры и их грамматические интерпретации. I. Обобщенная формальная конструктивно-продукционная структура. Кибернетика и системный анализ. 2014. №5. С. 8-16.
33. Шинкаренко В. И., Ильман В. М. Конструктивно-продукционные структуры и их грамматические интерпретации. II. Уточняющие преобразования. Кибернетика и системный анализ. 2014. №6. С. 15-28.
34. Закон України «Про охорону праці» прийнятий від 14 жовтня 1992 року № 2695-12;
35. НПАОП 0.700-7.15-18 Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями, затверджені від 14.02.2018 р. № 207;
36. ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджені від 10.12.1998 р. № 7;
37. НПАОП 80.0-1.12-04 Правила безпеки під час навчання в кабінетах інформатики навчальних закладів системи загальної середньої освіти, затверджений 16.03.2004 р. № 81;
38. Інструкція щодо надання послуг з ремонту побутової радіоелектронної апаратури, затверджені від 27.08.2000 р. № 20;
39. ДСН 3.3.6.042-99, Державні санітарні норми мікроклімату виробничих приміщень затверджені від 01.12.99 р. № 42;
40. ДСН 2.3.6.037-99, Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку затверджені від 01.12.99 р. № 37;
41. Наказ МОЗУ 16.06.2014 № 398 ПОРЯДОК надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою;

42. Наказ №398 Про затвердження порядків надання домедичної допомоги особам при невідкладних станах від 16.06.2014;
43. Порядку надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою, затверджений від 16.06.2014 р. № 398;
44. НАПБ А.01.001-2014 Правила пожежної безпеки в Україні, затверджений від 30.12.2014 Наказом № 1417

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Дніпровського
національного університету
залізничного транспорту
імені академіка В. Лазаряна
Борис БОДНАР

КОНСТРУКТИВНІ ПРОСТОРОВІ ПЕРЕТВОРЕННЯ ДВОВИМІРНИХ ФРАКТАЛІВ

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

1116130.01207-01-ЛЗ

Завідувач кафедри КІТ
Вадим ГОРЯЧКІН

Керівник розробки
Віктор ШИНКАРЕНКО

Виконавець
Владислав МОСІЄНКО

Нормоконтролер
Олена КУРОП'ЯТНИК

2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

1116130.01207-01

КОНСТРУКТИВНІ ПРОСТОРОВІ ПЕРЕТВОРЕННЯ ДВОВИМІРНИХ ФРАКТАЛІВ

Технічне завдання

Аркушів 21

2021

АНОТАЦІЯ

Документ 1116130.01207-01 «Конструктивні просторові перетворення двовимірних фракталів. Технічне завдання» входить до складу документації програмного засобу, який надає можливість конструювати двовимірні фрактали у 3D просторі, переглядати отримані фрактали та використовувати результат при побудові інших фракталів.

Даний документ описує приклади застосування програмного рішення, основні вимоги, календарний план розробки проекту, технічні та економіко-технічні розрахунки для програмного продукту.

ЗМІСТ

ВСТУП

Фрактали охоплюють широку та різноманітну сферу застосування у таких галузях людської діяльності, як інформатика, наука, радіотехніка, тощо. Фрактали мають практичне застосування в комп’ютерній графіці для задачі побудови зображень природних об’єктів, таких як чагарники, дерева, гірські ландшафти, поверхня моря, тощо. Французький математик Б. Мандельброт увів термін «фрактал» у 1975 році. Цей термін складається з двох латинських слів: frangere – ламати й fractus – дробовий. Одне з визначень фракталу звучить так: фракталом називається структура, що складається із частин, які в якомусь змісті подібні до цілого.

Програмне рішення «Конструктивні просторові перетворення двовимірних фракталів» дозволяє будувати двовимірні фрактали у тривимірному просторі та здійснювати перетворення над ними.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ № 690 ст. від 18.11.2020 ректора Дніпровського національного університету залізничного транспорту імені академіка В. Лазаряна «Про затвердження керівників та затвердження тем магістерських робіт».

Тема дипломної роботи: Конструктивні просторові перетворення двовимірних фракталів.

Керівник дипломної роботи – Шинкаренко В. І.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Функціональне призначення

Мета даного програмного рішення полягає у розробці додатку, що дозволяє здійснювати просторові перетворення двовимірних фракталів.

2.2 Експлуатаційне призначення

Удосконалення процесу дослідження двовимірних фракталів у просторі.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик наступні:

- створення графічних елементів площин;
- зміна координат площини у просторі;
- зміна форми площини з використанням опорних точок побудованої площини;
- копіювання створених елементів;
- вставка елементів занесених до буферу обміну;
- маркування побудованих елементів термінальними символами;
- маркування побудованих елементів символами правил;
- побудова виводу L-системи відповідно до заданих правил;
- перегляд побудованого фракталу, можливість його копіювання.

Вхідні дані:

- побудовані графічні елементи;
- правила L-системи;
- кількість ітерацій виводу L-системи.

Вихідні дані:

- згенерований графічний результат виводу відповідно до правил L-системи.

3.2 Вимоги до надійності

Вимоги до надійності програмного додатку наступні:

- програма повинна не допускати витіків пам‘яті;
- програма повинна мати максимальну продуктивність алгоритму побудови результату;
- програма повинна видавати коректний результат виводу відповідно до введеної кількості ітерацій;
- критичні ситуації в програмі повинні повідомляти користувача про помилку, та не приводити до аварійного завершення програми.

3.3 Умови експлуатації

Програмний засіб може функціонувати в умовах, відповідних до умов, які описані в документі [1].

Для стабільного функціонування програмного засобу необхідно дотримуватись таких умов:

- ознайомитися з керівництвом користувача;
- ЕОМ, які використовуються для роботи програмного продукту, повинні відповідати чинним вимогам та стандартам України, нормативним актам з охорони праці [2];
- програмний комплекс повинен використовуватись в приміщеннях, призначених для роботи ЕОМ з наступними кліматичними умовами: температура – 21-25° С, відносна вологість повітря 40-60%.

3.4 Вимоги до складу і параметрів технічних засобів

Для роботи з додатком необхідний ПК, що має наступні мінімальні характеристики:

- процесор: чотири-ядерний 32-, 64- або 86-бітний процесор з тактовою не менш ніж частотою 1.9 ГГц;
- оперативна пам’ять: не менш ніж 4гб;
- вільний простір на жорсткому диску: 100мб;
- периферійні пристрої: монітор, клавіатура, миша.

3.5 Вимоги до інформаційної і програмної сумісності

Програмний продукт має бути розроблений з використанням мови програмування JS за допомогою IDE Visual Studio Code.

Робота з додатком проводиться через браузер, що повинен бути встановлений на відповідній машині завчасно.

3.6 Вимоги до маркування і упаковки

Програмного продукт не потребує випуску у фізичному форматі, так я являє собою веб-сайт. У разі необхідності фізичного відображення посилань на веб-

додаток, можливо згенерувати qr-код та використовувати його як засіб для отримання доступу до веб-додатку.

Приклад маркування приведений на рис. 3.1:

Конструктивні просторові перетворення двовимірних
фракталів Мосієнко В.С.
ДНУЗТ Каф. КІТ
2021

Рисунок 3.1 – Приклад маркування

3.7 Вимоги до транспортування і зберігання

Так як програмний засіб представляє собою веб-додаток, то особливих вимог до транспортування не має, тому що програмний засіб фактично знаходиться в мережі інтернет. Код програми також потенційно зберігається на хмарному репозиторії, тож зібрати виробничу версію програмного додатку можливо звідки завгодно.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація представляє собою перелік наступних документів:

- специфікація;
- текст програми;
- опис програми;
- керівництво користувача. Керівництво з побудови двовимірних фракталів у просторі.

Вся документація до програми повинна задовольняти вимогам державного стандарту до оформлення програмних документів ГОСТ 19.101-77 «Єдина система програмної документації. Види програм та програмних документів» [3].

5 ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Оцінка розміру програмного забезпечення є початковим етапом розрахунку величини трудових витрат розробників. Основні відмінності методик, що застосовуються в оцінці трудовитрат, полягають у використовуваному типі критерію оцінки якості (кількісний або якісний) [4].

За моделлю COCOMO, розмір проекту S вимірюється в рядках коду LOC (KLOC), а трудовитрати в людино-місяцях [5].

$$E = a \cdot S^b \cdot EAF, \quad (1)$$

де E – витрати праці на проект (в людино-місяцях);

S^b – розмір коду (в KLOC);

EAF – фактор уточнення витрат (effort adjustment factor).

Для простих систем, a = 2,4; b = 1,05

Припустимо, що розмір програмного коду програмного засобу – 1405 рядків:

$$E = 2{,}4 \cdot 1{,}4^{1{,}05} \cdot 1 = 3.4 \quad (2)$$

Отже, згідно моделі COCOMO, орієнтовні трудовитрати на проект складуть приблизно 3,4 людино-місяці.

Наведемо розрахунки вартості розробки проекту «Конструктивні просторові перетворення двовимірних фракталів». Основними статтями витрат прийняті:

– основна заробітна плата;

– накладні витрати;

– відрахування на соціальні потреби;

– витрати на персональний комп’ютер і ліцензійні базові програмні засоби.

Основна заробітна плата (ОЗП) визначається виходячи з кількості розробників, часу виконання розробки (годин), а також заробітної плати в розрахунку на одну годину. Розрахунок заробітної платні проводиться по формі табл. 5.1.

Таблиця 5.1 – Фонд місячної заробітної плати

№ п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати грн
			чол	місяців	
1	інженер-програміст	16,248	1	3,4	55,243

Описаний в проекті програмний продукт буде розроблений одним програмістом в період з 01.07.20 до 15.10.21, що складає 107 днів або 22 робочих тижні. Витрати робочого часу прийняті за 40 годин у тиждень. Погодинна ставка кваліфікованого інженера–програміста складає 101,55 грн/год[6]. Таким чином, витрачено робочого часу:

$$t_{\text{розробки}} = N_{\text{чол}} \cdot N_{\text{тиж}} \cdot N_{\text{год}}, \quad (3)$$

де $N_{\text{чол}}$ – кількість виконавців, *чол*;

$N_{\text{тиж}}$ – тривалість розробки;

$N_{\text{год}}$ – витрати робочого часу, год;

$$t_{\text{розробки}} = 1 \cdot 22 \cdot 40 = 880 \text{ чол/год}. \quad (4)$$

ОЗП визначається за формулою:

$$\text{ОЗП} = t_{\text{розробки}} \cdot N \cdot K_{KB}, \quad (5)$$

де $t_{\text{розробки}}$ – витрати праці у чол/год;

N – погодинна ставка;

K_{KB} – коефіцієнт кваліфікації програміста, приймається 0,75.

ОЗП складає:

$$\text{ОЗП} = 880 \cdot 101{,}55 \cdot 0{,}75 = 67023\text{грн}. \quad (6)$$

Відрахування на соціальні потреби встановлюються у відсотках від суми заробітної плати[7]:

$$C_{\text{соц}} = \frac{\text{ОЗП} \cdot 22\%}{100\%}$$

$$C_{\text{соц}} = \frac{67023 \cdot 22\%}{100\%} = 13404\text{грн}. \quad (7)$$

Отримані результати за (6) та (7) підсумовуються. Вони складають 80427грн. та визначають основні прямі витрати.

Накладні витрати враховують загальногосподарчі витрати по забезпеченню проведення роботи: витрати на опалення, електроенергію, амортизація будівель, зарплату адміністративного персоналу та інше. Вони визначаються в процентах (30 – 40%) від суми прямих витрат:

$$С_{накл} = \frac{(ОЗП+С_{соц})\cdot 35\%}{100\%}; \quad (8)$$

$$С_{накл} = \frac{(67023+13404)\cdot 35\%}{100\%} = 28149 \text{ грн.} \quad (9)$$

На протязі усього терміну використання нової техніки підприємство щорічно витрачає певні кошти, пов’язані з її експлуатацією.

Експлуатаційні витрати на персональний комп’ютер визначаються протягом терміну розробки програмного засобу в залежності від вартості комп’ютеру. В експлуатаційні витрати входять:

– вартість витратних матеріалів;

– витрати на ремонт;

– заробітна плата ремонтника;

– оренда приміщення;

– додаткові витрати – прибирання приміщення, охорона, оренда, комунальні послуги;

– амортизаційні витрати на персональний комп'ютер і програмне забезпечення;

– витрати на електроенергію ($С_{ел}$),

які визначаються за формулою:

$$С_{ел} = P \cdot B \cdot T_{розр}, \quad (10)$$

де Р – потужність комп’ютера та допоміжних споживачів електричної енергії, приймається 0,45 кВт/год;

B – вартість 1 кВт/година, складає 1,68 грн; [8]

$T_{розр}$ – час роботи з ЕВМ, приймається рівним робочому часу.

Витрати на електроенергію визначаються так:

$$C_{\text{ел}} = 0{,}45 \cdot 1{,}68 \cdot 880 = 1{,}874 \text{ грн.} \quad (11)$$

Витрати на витратні матеріали ($\text{С}_{\text{вм}}$) протягом всього терміну експлуатації приблизно 10% від вартості комп'ютеру. Вартість робочої станції приймається 18 000 грн. (Lenovo IdeaPad 5)[9], термін експлуатації – 5 років. Можна визначити витрати за період створення програмного засобу:

$$\text{С}_{\text{вм}} = \text{В}_{\text{ком}} \cdot \frac{N_{\text{Д}}}{N_{\text{експ}} \cdot 365} \cdot \frac{10\%}{100\%}, \quad (12)$$

де $B_{КОМ}$ – вартість персонального комп'ютеру;

$N_{\text{Д}}$ – кількість днів розробки програмного продукту;

$N_{\text{експ}}$ – термін експлуатації персонального комп'ютеру.

Витрати на витратні матеріали визначаються так:

$$C_{\text{вм}} = 18000 \cdot \frac{107}{5 \cdot 365} \cdot \frac{10}{100} = 105{,}48 \text{ грн.} \quad (13)$$

Заробітна плата ремонтника ($C_{\text{рем}}$) визначена наступним чином: на ремонт 50 комп'ютерів потрібен один інженер-системотехнік. Його середньомісячна заробітна плата приймається 9000 грн. Тоді в перерахунку на один комп'ютер його заробітна плата за період розробки програмного продукту складає:

$$C_{\text{рем}} = \frac{C_{\text{рем}}'}{N_{КОМ}} \cdot T_{\text{міс}}, \quad (14)$$

де $C_{\text{рем}}'$ – середньомісячна заробітна плата;

$N_{КОМ}$ – кількість комп'ютерів на одного ремонтника.

$T_{мес}$ – час розробки програмного продукта, міс.

Заробітна плата ремонтника ($C_{\text{рем}}$) буде складати:

$$C_{\text{рем}} = \frac{9000}{50} \cdot 2 = 360 \text{ грн.}$$

За статистикою витрати на комплектуючи вироби ($\text{С}_{КОМ}$) для ремонту персонального комп'ютера складає 10% від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали:

$$C_{КОМ} = C_{ВМ} = 105{,}48 \text{ грн.} \quad (16)$$

Амортизаційні відрахування на персональний комп'ютер (АПК) визначені з положення, що амортизаційний період в даний час дорівнює терміну морального

старіння обчислювальної техніки і складає 3 роки. За 3 роки амортизаційні відрахування на персональний комп'ютер дорівнюють вартості комп'ютера. За період проектування амортизаційні відрахування складуть:

$$\text{АКП} = \text{В}_{КОМ} \cdot \frac{N_{\text{Д}}}{N_{\text{експ}} \cdot 365}; \quad (17)$$

$$\text{АКП} = 18000 \cdot \frac{107}{5 \cdot 365} = 1055$$

Нехай термін морального старіння для Windows дорівнює 5 років та MATLAB – 1 рік тоді амортизаційні відрахування на програмне забезпечення дорівнюють його вартості.

Для функціонування персонального комп'ютера використовувалася операційна система Windows 10, для написання програмного забезпечення – програмне середовище Visual Studio Code.

$$\text{АКП}_w = 13800 \cdot \frac{107}{5 \cdot 365} = 809$$

Розрахунок амортизаційних відрахувань на програмне забезпечення зведений в табл. 5.2.

Таблиця 5.2 – Використовуване програмне забезпечення

Найменування програмного забезпечення	Вартість програмного забезпечення, *грн*	Джерело придбання	Амортизаційні відрахування, *грн*
WINDOWS 10 PROFESSIONAL	13800	http://mtsoft.kiev.ua/product/windows-10-professional	809
Visual Studio Code	-	https://code.visualstudio.com/	-
Всього:	13800	-	809

Додаткові витрати ($C_{\text{дод}}$): прибирання приміщень, охорона, комунальні послуги важко оцінити точно і прийняти рівними 50% заробітної плати інженера-програміст, тобто 13540 гривень на місяць.

Оренду приміщень приймемо рівною 2500 гривень на місяць базуючись на варіантах представлених на сайті для пошуку приміщень в оренду [10].

Сумарні експлуатаційні витрати на один такий персональний комп'ютер складають:

$$C_{\text{експ}} = C_{\text{ел}} + C_{\text{ВМ}} + C_{\text{рем}} + \text{АПК} + \text{АПО} + C_{\text{ор}} + C_{\text{дод}}; \quad (18)$$

$$C_{\text{експ}} = 1874 + 105{,}48 + 360 + 1055 + 809 + 2500 + 8585 = \\ = 15288\text{грн}. \quad (19)$$

Результати розрахунків зведено у табл. 5.3.

Таблиця 5.3 – Експлуатаційні витрати на ПК і ПЗ.

Найменування витрат	Витрати, грн
Витрати на електроенергію	259,2
Вартість витратних матеріалів	105,48
Витрати на ремонт	1874
Амортизація персонального комп'ютера	1055
Амортизація програмного забезпечення	809
Оренда приміщення	2500
Додаткові витрати	8585
Всього	**15288**

Таким чином, витрати на створення даного програмного продукту складають:

$$C_{\text{розробки}} = \text{ОЗП} + C_{\text{соц}} + C_{\text{накл}} + C_{\text{експ}}; \quad (20)$$

$$C_{\text{розробки}} = 67023 + 13404 + 28149 + 15288 = \\ = 123864 \text{ грн}. \quad (21)$$

Розрахунок витрат зведено у табл. 5.4.

Таблиця 5.4 – Кошторис витрат на розробку програмного засобу

Найменування витрат	Витрати, грн
Основна заробітна плата	67023
Відрахування на соціальні потреби	13404
Накладні витрати	28149
Експлуатаційні витрати	15288
Всього	123864

За отриманими значеннями техніко-економічних показників проекту складено кошторис витрат на розробку сучасного програмного забезпечення для оцінки схожості програм. За результатами розрахунків, приблизна вартість розробки складає 123864 грн.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії розробки наведені в табл. 6.1.

Таблиця 6.1 – Стадії та етапи розробки

Стадії	Зміст робіт	Терміни виконання
1	2	3
1. Технічне завдання	Постановка завдання Збір початкових матеріалів Вибір і обґрунтування критеріїв ефективності і якості програми, що розробляється Обґрунтування необхідності проведення науководослідних робіт Визначення структури вхідних і вихідних даних Попередній вибір методів рішення задач Обґрунтування доцільності застосування раніше розроблених програм Визначення вимог до технічних засобів Обґрунтування принципової можливості рішення поставленої задачі Визначення вимог до програми Техніко-економічне обґрунтування розробки програми Визначення стадій, етапів і термінів розробки програми і документації на неї Вибір мов програмування Визначення необхідності проведення науководослідних робіт на подальших стадіях Узгодження і затвердження технічного завдання	25.01.2021 – 10.03.2021

Продовження таблиці 6.1

1	2	3
2. Робочий проект	Програмування і налагодження програми Розробка програмних документів відповідно до вимог ГОСТ 19.101-77 Розробка, узгодження і затвердження програми і методики випробувань Проведення попередніх і інших видів випробувань Коригування програми і програмної документації за наслідками випробувань	11.03.2021 – 24.05.2021
3. Впровадження	Підготовка і передача програми і програмної документації для супроводу і (або) виготовлення	25.05.2021 – 11.06.2021

7 ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Контроль за виконанням роботи виконує керівник розробки.

Прийом програмного продукту здійснюється прийомною комісією і керівником розробки.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. ДСанПіН 3.3.2-007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Текст] / Постанова Головного державного санітарного лікаря України від 10 грудня 1998 р. № 7 – К.,1998.
2. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень [Текст]/ Постанова Головного Державного санітарного лікаря України від 01.12.1999 № 42 – К., 1999.
3. ГОСТ 19.101-77. Виды програм и программных документов [Текст]/ Постановление Государственного комитета стандартов Совета Министров СССР от 20 мая 1977 г. – М., 1977.
4. Борисенков, Евгений. «Методики оценки трудозатрат по разработке программного обеспечения информационных систем.»
5. «Методики оценки трудозатрат по разработке программного обеспечения информационных систем» [Ел. ресурс]. Available: http://repository.enu.kz/
6. «Статистика зарплат програмістів, тестувальників і PM в Україні | DOU» Available: https://jobs.dou.ua/salaries/#period=jun2021&city=all&title=Software%20Engineer&language=JavaScript&spec=&exp1=0&exp2=1
7. «Методики оценки трудозатрат» [Ел. ресурс]. Available: http://www.hups.mil.gov.ua/periodic-app/article/11953/soi_2014_8_33.pdf.
8. «Тарифи на електроенергію для населення в Україні» [Ел. ресурс]. Available: https://bankchart.com.ua/spravochniki/indikatory_rynka/electric_tariff.
9. «Лучшие ноутбуки для программирования 2021 года: рейтинг для студентов, программистов, веб и фронтенд разработчиков». [Ел. ресурс]. Available: https://speedcamupdates.ru/prilozheniya/top-noutbukov-dlya-programmiro-vaniya.html
10. «Сайт оголошень OLX» [Ел. ресурс]. Available: https://www.olx.ua/nedvizhimost/kommercheskaya-nedvizhimost/arenda-

kommercheskoy-nedvizhimosti/ofisnye-pomescheniya/dnepr/?search%5Bfilter_float_price%3Afrom%5D=10&search%5Bfilter_float_price%3Ato%5D=100&search%5Border%5D=filter_float_price%3Aasc

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Дніпровського
національного університету
залізничного транспорту
імені академіка В. Лазаряна
Борис БОДНАР

КОНСТРУКТИВНІ ПРОСТОРОВІ ПЕРЕТВОРЕННЯ ДВОВИМІРНИХ ФРАКТАЛІВ

Робочий проект
1116130.01207-01-ЛЗ
ЛИСТ ЗАТВЕРДЖЕННЯ

Завідувач кафедри КІТ
Вадим ГОРЯЧКІН

Керівник розробки
Олена КУРОП'ЯТНИК

Виконавець
Владислав МОСІЄНКО

Нормоконтролер
Олена КУРОП'ЯТНИК

2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

1116130.01207-01

КОНСТРУКТИВНІ ПРОСТОРОВІ ПЕРЕТВОРЕННЯ ДВОВИМІРНИХ ФРАКТАЛІВ

Специфікація

Аркушів 2

2021

Позначення	Найменування	Примітка
1116130.01207-01-ЛЗ	Лист затвердження	
1116130.01207-01 12 01-ЛЗ	Лист затвердження	
1116130.01207-01 12 01	Текст програми	
1116130.01207-01 13 01-ЛЗ	Лист затвердження	
1116130.01207-01 13 01	Опис програми	
1116130.01207-01 І3 01-ЛЗ	Лист затвердження	
1116130.01185-01 І3 01	Керівництво користувача. Керівництво з використання системи визначення відповідності тексту програми графічному представленню алгоритму	

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

1116130.01207-01 13 01

КОНСТРУКТИВНІ ПРОСТОРОВІ ПЕРЕТВОРЕННЯ ДВОВИМІРНИХ ФРАКТАЛІВ

Опис програми

Аркушів 15

2021

АНОТАЦІЯ

Документ 1116130.01207-01 13 01 ««Конструктивні просторові перетворення двовимірних фракталів». Опис програми» входить до складу програмної документації додатку, що надає можливість будувати та досліджувати двовимірні фрактали у тривимірному просторі.

У даному документі представлений опис програми клієнтської частини системи: функціональне призначення, опис логічної структури, використані технічні засоби, виклик і завантаження, вхідні і вихідні дані, опис інтерфейсу користувача, порядок роботи з програмою.

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Програмний додаток «Конструктивні просторові перетворення двовимірних фракталів» являє собою інструмент для побудови, редагування та дослідження двовимірних фракталів у тривимірному просторі.

Для коректного функціонування програми потрібно мати ПК з встановленою операційною систему та веб-браузером.

Програма реалізована на мові програмування JavaScript у IDE Visual Studio Code, а також з використанням мови гіпертекстової розмітки HTML та формальної мова опису зовнішнього вигляду документа CSS.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Метою програмного додатку є побудова двовимірних фракталів у просторі, їх модифікація, перевикористання та дослідження.

Функціональні вимоги до додатку:

- створення двовимірного фракталу;
- переміщення двовимірного фракталу;
- огляд результуючого фракталу;
- використання результуючого фракталу в правилах для побудови інших фракталів.

Відповідно до функціональних вимог додатку було визначено наступні вимоги до даних програми:

- кількість ітерацій для роботи L-системи повинна бути позитивним цілим числом.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

3.1. Методи та алгоритми програмного засобу

Програмного засобу базується на подійному підході (шаблон Event Emitter). Основна ідея використання подійного підходу в реалізації програмного засобу для побудови фракталів – використання його гнучкості для досягнення простоти реалізації базового інструментарію програмного засобу та його легкого вдосконалення або внесення суттєвих правок в логіку роботи програмного засобу з мінімальними витратами часу та інших ресурсів.

Програмний засіб ділиться на контексти, що взаємодіють між собою та виконують кожен свою частину роботи. Логічні контексти підписуються на інші, відповідно до бізнес логіки програмного засобу. Підпис відбувається в місці ініціалізації загального контексту програмного засобу.

Кожен контекст має можливість для ініціалізації початкових даних, що потрібні йому для коректної роботи. Наприклад, контекст інструментів, відповідальний за роботу панелі інструментів, що знаходиться в лівій частині інтерфейсу програмного засобу, виконує налаштування поточного інструменту на початку ініціалізації програмного засобу.

Кожен з контекстів програмного засобу виконую підпис на потрібні йому інші контексти відповідно до бізнес логіки програмного засобу.

Для побудови графічних об'єктів використовується бібліотека THREE.JS. За її ініціалізацією та використання відповідає контекст полотна.

Для побудови фракталів було використано конструктивно-продукційне моелювання.

3.2. Структура програми з описом функцій складових частин і зв’язки між ними

Приведемо опис основних складових частин програми та зв’язків між ними.

CanvasContext – контекст, що відповідає за роботу з графікою, полотном, камерою та інших речей, що стосуються графіки.

DraggingContext – контекст, що відповідає за збереження інформації про об’єкти переміщення.

ToolsContext – контекст, що відповідає за логіку панелі інструментів, що використовуються для побудови графічних об’єктів та взаємодії з полотном.

SelectionContext – контекст, відповідальний за збереження інформації про вибрані елементи та пересилку інформації про дії над вибраними елементами на полотні.

UiContext – контекст, що відповідає за оновлення інтерфейсу користувача та обробку реакцій на події, що з нього надходять.

ClipboardContext – контекст, відповідальний за копіювання об’єктів полотна.

3.3. Залежності програмного засобу від інших програм або систем

Для коректної роботи програмного засобу необхідні:

- операційна система зі встановленим веб-браузером
- Node.js 16^.

4 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Для коректного функціонування програмного засобу достатньо мати робочий персональний комп’ютер, що має наступні технічні характеристики:

- процесор: чотири-ядерний 32-, 64- або 86-бітний процесор з тактовою не менш ніж частотою 1.9 ГГц;
- не менше 256 Мб ОЗУ;
- оперативна пам’ять: не менш ніж 4гб;
- вільний простір на жорсткому диску: 100мб;
- периферійні пристрої: монітор, клавіатура, миша.

5 ВИКЛИК ТА ЗАВАНТАЖЕННЯ

Програмний засіб приводиться в дію одним з двох способів. Перший спосіб: запуск командами npm install, npm start для запуску в режимі розробника. Другий спосіб: відкриття веб-сайту з додатком, якщо додаток доступний мережі інтернет. Файли, що завантажує програмний додаток, зберігаються у папці, що є базовою для завантаження файлів браузером з мережі інтернет.

6 ВХІДНІ ДАНІ

Вхідні дані програмного продукту:

- аксіома L-системи у вигляді графічних об’єктів на полотні;
- правила L-системи у вигляді графічних об’єктів на полотні;
- потрібна кількість ітерацій роботи L-системи.

7 ВИХІДНІ ДАНІ

Вихідні дані програмного продукту:

– побудований двовимірний фрактал відповідно до представлених аксіоми та правил.

8 ОПИС ІНТЕРФЕЙСУ КОРИСТУВАЧА

8.1. Опис станів програми

Можливі стани програми наведено у табл. 8.1.

Таблиця 8.1 – Стани програми

№ стану	Назва стану	Опис стану	Рекомендовані дії
1	Відкриття вкладки з додатком	Додаток запускається.	Очікування завантаження вкладки з додатком
2	Створення аксіоми для L-системи	Вибір графічних елементів, додавання їх на полотно, зміна за потреби, формування аксіоми L-системи	Створити потрібну аксіому для L-системи
3	Створення правил для L-системи	Вибір графічних елементів, додавання їх до полотна, зміна за потреби, формування правил L-системи	Створити потрібні правила для L-системи
4	Введення кількості ітерацій роботи L-системи	Вводиться кількість ітерацій роботи L-системи, у вигляді цілого додатного числа	Ввести ціле додатне число ітерацій роботи L-системи
5	Генерація результату роботи L-системи	Побудова фракталу відповідно до заданих аксіоми та правил L-системи	Натиснути кнопку “L” панелі інтструментів та очікувати завершення генерації результатів
6	Огляд результатів	Огляд генерованих результатів роботи L-системи за допомогою можливостей камери	Вибрати інструмент “Переміщення” на панелі інструментів та обертати, приближати, віддаляти камеру для огляду результуючого фракталу

8.2. Опис переходів між станами програми

Схема переходів програми наведена на рис. 8.1, де цифри є номером стану програми (табл. 8.1). Вийти з програми можливо під час усіх станів крім першого стану.

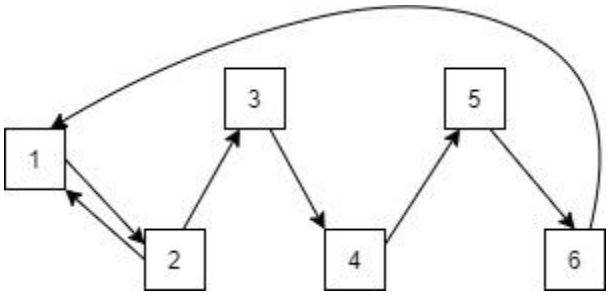

Рисунок 8.1 – Схема переходів між станами програми

8.3. Опис керування діалогом

Керування діалогом відбувається за допомогою екранних форм та панелі інструментів програмного засобу.

8.4. Форматування екрана

Головне вікно програми можна форматувати, а саме змінювати його розмір, змінюючи розмір вікна браузера, або вибравши емульований пристрій відкривши панель розробника в вікні браузера. Інші засоби форматування не передбачені.

9 ПОРЯДОК РОБОТИ З ПРОГРАМОЮ

Порядок роботи з програмою передбачає наступну послідовність операцій:

– формування аксіоми L-системи у графічному вигляді;
– формування правил L-системи у графічному вигляді;
– генерація результуючого фракталу відповідно до описаних аксіоми та правил L-системи;
– дослідження отриманого результату.

10 ПОВІДОМЛЕННЯ

Повідомлення користувачу, які можуть з’явитися під час роботи з програмою описані в таблиці 10.1.

Таблиця 10.1 – Повідомлення користувачу

Текст повідомлення	Опис ситуації	Рекомендовані дії
Не створена аксіома L-системи	Генерація результатів не можлива через відсутність аксіоми L-системи	Створити аксіому L-системи
Кількість ітерацій роботи L-системи введено не вірно	Значення поля для введення кількості ітерацій роботи L-системи не відповідає вимогам	Ввести коректне значення для кількості кроків роботи L-системи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

1116130.01207-01 І3 01

КОНСТРУКТИВНІ ПРОСТОРОВІ ПЕРЕТВОРЕННЯ ДВОВИМІРНИХ ФРАКТАЛІВ

Керівництво користувача. Керівництво з використання системи конструктивно просторових перетворень двовимірних фракталів у просторі

Аркушів 12

2021

АНОТАЦІЯ

Документ 1116130.01207-01 І3 01 ««Конструктивні просторові перетворення двовимірних фракталів». Керівництво користувача. Керівництво з використання системи конструктивно просторових перетворень двовимірних фракталів у просторі» входить до складу програмної документації додатку, що надає можливість будувати та досліджувати двовимірні фрактали у тривимірному просторі.

У даному документі описано призначення та умови застосування програмного засобу, інструкції з підготовки до роботи, а також опис основних операцій та аварійних ситуацій програмного додатку.

ЗМІСТ

ВСТУП

Програмний додаток «Конструктивні просторові перетворення двовимірних фракталів» використовується студентами, що навчаються за спеціальності Інженерія програмного забезпечення, а також викладачами та в потенціалі інших наукових робітників для дослідження двовимірних фракталів у просторі.

Програмний додаток дає можливості для конструювання та дослідження двовимірних фракталів у тривимірному просторі.

Для використання програми необхідні мінімальні навички роботи з ПК, операційною системою та веб-браузером.

Для правильної роботи програмного додатку необхідно ознайомитися з описом програми та керівництвом користувача.

1 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

1.1. Функціональне призначення програми

Основною метою даного програмного засобу є надання можливостей для побудови, редагування та дослідження двовимірних фракталів у просторі.

До функціональних можливостей програмного додатку входить:

- створення площин;
- редагування площин;
- задання аксіоми та правил L–системи Ліденмайера;
- генерація результуючого фракталу відповідно до заданих аксіоми та правил;
- огляд результуючого фракталу.

Основне призначення програмного додатку полягає наданні можливостей для дослідження двовимірних фракталів у просторі.

1.2. Вимоги до складу і параметрів технічних засобів

Для коректного функціонування програмного забезпечення достатньо мати робочий персональний комп’ютер, що має наступні технічні характеристики:

- процесор: чотири-ядерний 32-, 64- або 86-бітний процесор з тактовою не менш ніж частотою 1.9 ГГц;
- не менше 256 Мб ОЗУ;
- оперативна пам’ять: не менш ніж 4гб;
- вільний простір на жорсткому диску: 100мб;
- периферійні пристрої: монітор, клавіатура, миша.

1.3. Вимоги до інформаційної і програмної сумісності

Для правильного функціонування програмного засобу необхідно щоб на персональному комп’ютері було встановлено операційну систему, веб-браузер та Node.js 16 версії або вище.

2 ПІДГОТОВКА ДО РОБОТИ

Програмний засіб приводиться в дію одним з двох способів. Перший спосіб: запуск виконанням послідовності команд npm install, npm start, виконуючи їх з кореневого каталогу проекту, для запуску програмного засобу в режимі розробника. Другий спосіб: відкриття веб-сайту з додатком, якщо додаток доступний мережі інтернет. Файли, що завантажує програмний додаток, зберігаються у папці, що є базовою для завантаження файлів браузером з мережі інтернет.

3 ОПИС ОПЕРАЦІЙ

На початку роботи з програмним засобом можливі наступні операції:

1. Вибір інструменту для роботи з переміщенням (кнопка з іконкою курсору на панелі інструментів);
2. Вибір інструменту для створення площин (кнопка з іконкою площини на панелі інструментів);
3. Переміщення створених об'єктів натиснувши лівою кнопкою миші по елементу та утримуючи її, виконувати переміщення об’єкту;
4. Копіювання створених елементів комбінаціями клавіш “ctrl+c”;
5. Вставка копійованих елементів комбінаціями клавіш “ctrl+v”;
6. Генерація результату обчислень (кнопка з іконкою L);
7. Огляд результатів генерації за допомогою обертання камери, затиснувши ліву кнопку миші на пустому місці полотна та переміщуючи курсор миші;

4 АВАРІЙНІ СИТУАЦІЇ

У разі виникнення аварійних ситуацій, що призводять до невідворотних запинок в програмі, необхідно закрити вкладку з програмним засобом та відкрити його знов, або оновити сторінку з програмним засобом та спробувати знову.

У разі виявлення інших аварійних ситуацій, необхідно закрити програмний додаток та зв’язатися з персоналом, що супроводжує даний додаток.

5 РЕКОМЕНДАЦІЇ ЩОДО ЗАСВОЄННЯ

Інтерфейс програмного засобу зображено на рис. 5.1.

Рисунок 5.1 – Інтерфейс програмного засобу

Для створення об’єкту площини виберемо інструмент створення площин та натиснумо лівою кнопкою миші по полотну (рис. 5.2).

Рисунок 5.2 – Створений об’єкт площини

Для переміщення створеного об’єкту виконаємо переміщення курсору миші затиснувши курсор над об’єктом (рис. 5.3).

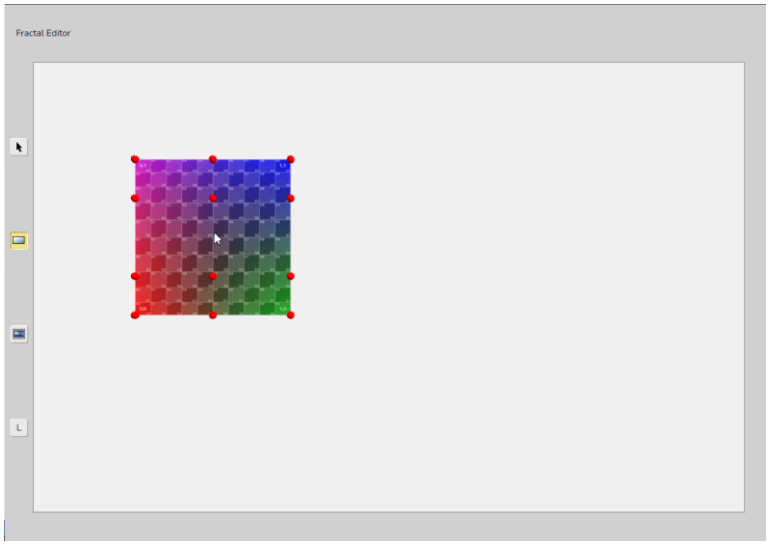

Рисунок 5.3 – Переміщення створеного елементу на полотні

Редагування створеного об’єкту відбувається шляхом взаємодії з опорними точками побудованого об’єкту (рис. 5.4). Виконаємо деформацію площини затиснувши курсор на одній з опорних точок та перемістивши курсор у потрібному напрямі.

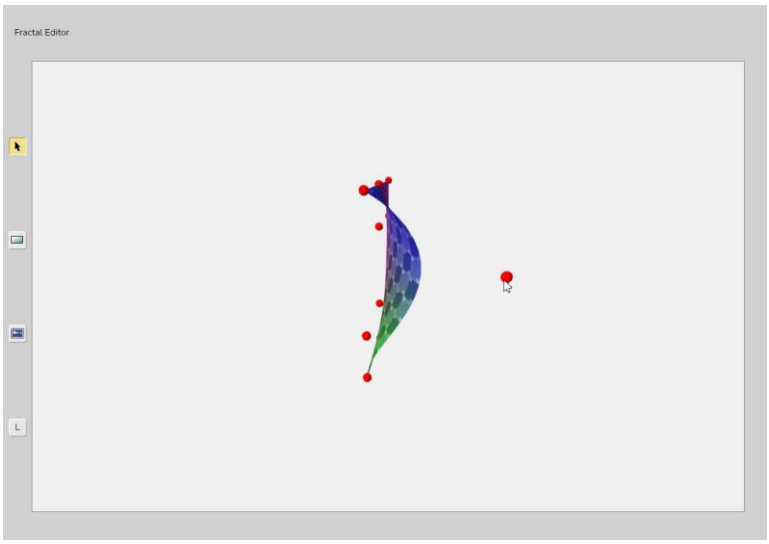

Рисунок 5.4 – Деформація створеної площини на полотні

Задання термінальних символів побудованим об’єктам відбувається за допомогою вибору потрібного об’єкта та внесенням потрібного символу до поля вводу, призначеного для вводу термінальних символів (рис. 5.5).

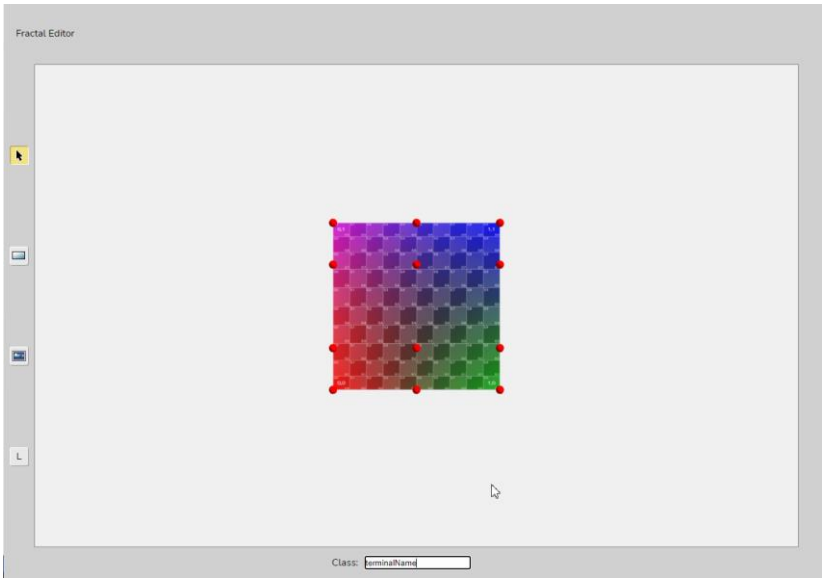

Рисунок 5.5 – Задання термінального символу для об’єкту / групи об’єктів

Для генерації результуючого фракталу натиснемо на відповідно кнопку в панелі інструментів та введемо потрібну кількість ітерацій роботи L–системи (рис. 5.6).

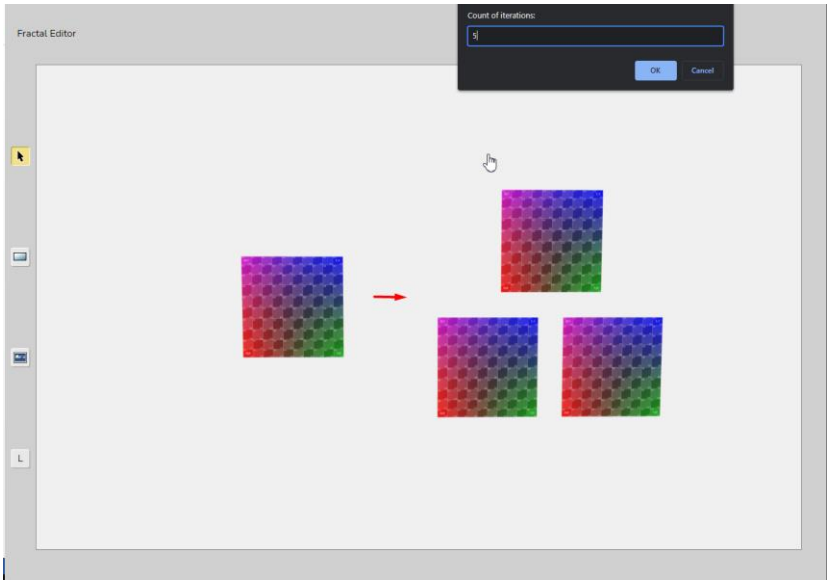

Рисунок 5.6 – Спливаюче вікно для введення кількості ітерацій роботи L–системи Ліденмайера

Після завершення процесу генерації буде виведено результуючий фрактал на окремій формі, призначеній для перегляду та дослідженні результатів (рис. 5.7).

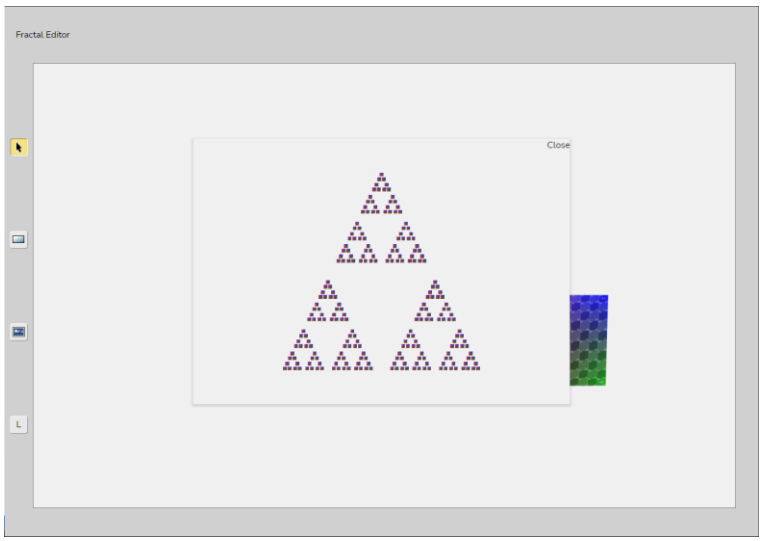

Рисунок 5.7 – Результат генерації відповідно до правил та аксіоми заданих L–системі

Виконаємо огляд результатів за допомогою модуля камери, утримуючи натиснутою ліву кнопку миші на полотні та переміщуючи курсор в потрібному напрямі (рис. 5.8).

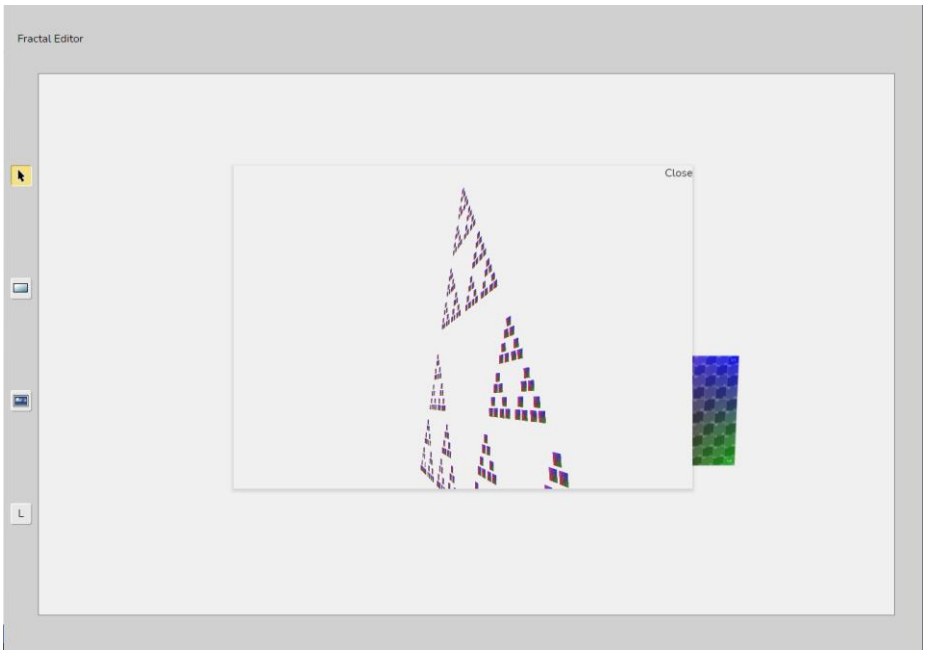

Рисунок 5.8 – Огляд результатів генерації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

1116130.01207-01 І3 01

КОНСТРУКТИВНІ ПРОСТОРОВІ ПЕРЕТВОРЕННЯ ДВОВИМІРНИХ ФРАКТАЛІВ

Текст програми

Аркушів 26

2021

АНОТАЦІЯ

Документ 1116130.01207-01 І3 01 ««Конструктивні просторові перетворення двовимірних фракталів». Текст програми» входить до складу програмної документації засобу, що надає можливість будувати та досліджувати двовимірні фрактали у тривимірному просторі.

Документ містить текст програми. Програма реалізована на мові JavaScript у програмному середовищі Visual Studio Code.

ЗМІСТ

1 СТРУКТУРА ПРОГРАМИ

Основа структури проекту базується на подієвому підході та налічує 7 логічних контекстів, повністю ізольованих один від одного.

На рис. 1.1 зображено основні програмні модулі логіки програмного засобу та взаємозв‘язки між ними.

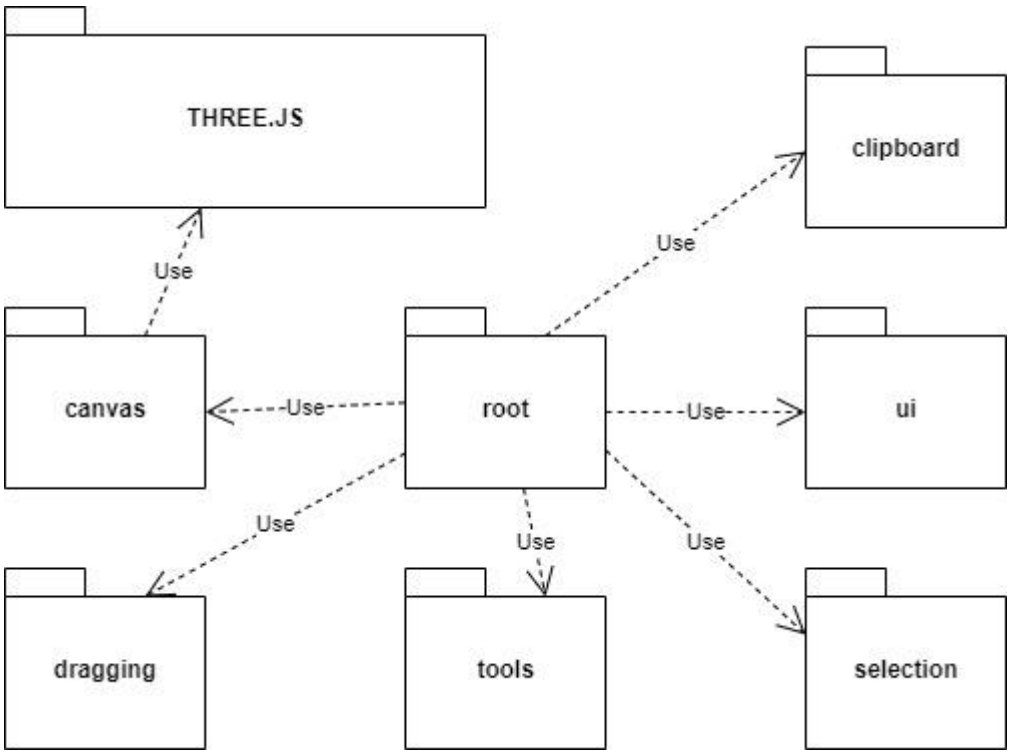

Рисунок 1.1 – UML – діаграма пакетів програмного засобу

Приведемо опис відповідних програмних модулів.

canvas.js – модуль, відповідальний за побудову графічних об’єктів, роботу з полотном, камерою та інших речей, що стосуються графіки.

dragging.js – модуль, відповідальний за фіксування інформації про об’єкти що переміщуються.

tools.js – модуль, відповідальний за логіку роботи вибору інструментів, потрібних для побудови графічних об’єктів та взаємодії з полотном.

selection.js – модуль, відповідальний за збереження інформації про вибрані елементи.

ui.js – модуль, відповідальний за роботу з інтерфейсом користувача – оновлення тих чи інших його частин, видання повідомлень та інше.

clipboard.js – модуль, відповідальний за збереження копійованих графічних елементів полотна.

root.js – модуль, відповідальний за зв’язку та запуск процесу ініціалізації усіх інших модулів.

three.js – модуль, відповідальний за роботу з графікою

Файлова структура проекту зображена на рис. 1.2.

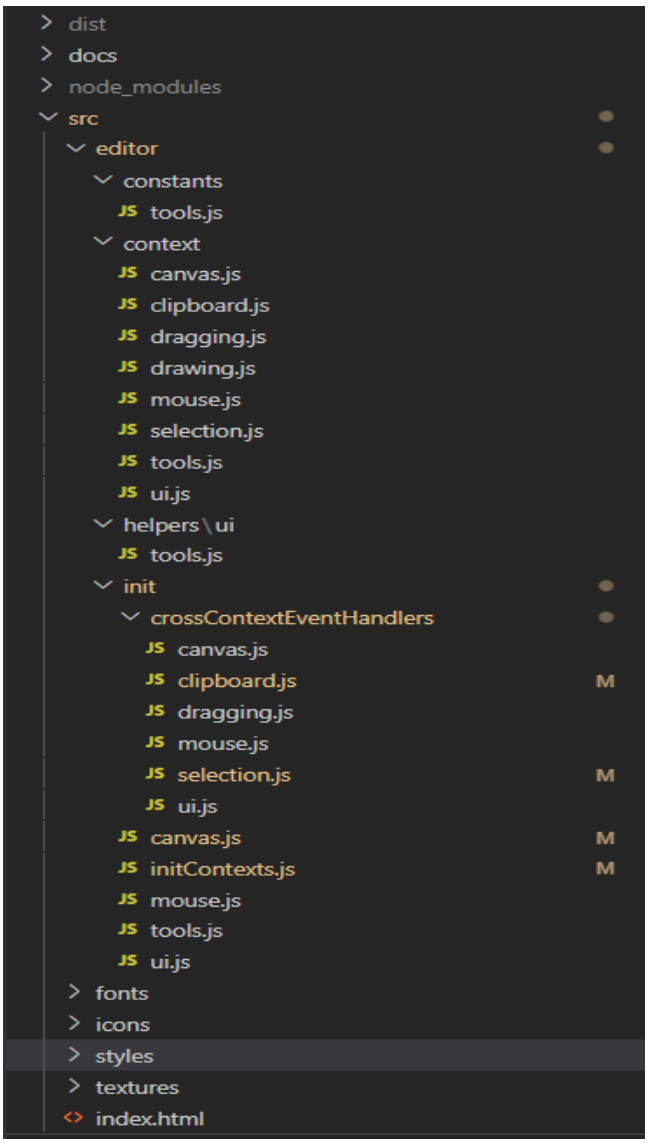

Рисунок 1.2 – Файлова структура проекту

2 ТЕКСТ ПРОГРАМИ

editor/context/canvas.js

```
import * as THREE from 'three';
import { OrbitControls } from
'three/examples/jsm/controls/OrbitControls.js
';
import { NURBSSurface } from
'three/examples/jsm/curves/NURBSSurface.js';

THREE.Object3D.prototype.getObjectsByCustomPr
operty = function (
  property = '',
  propertyValue,
  resultArray = []
) {
  // check the current object
  if (this.userData[property] ===
propertyValue) resultArray.push(this);

  // check children
  for (var i = 0; i < this.children.length;
i++) {
    var child = this.children[i];

child.getObjectsByCustomProperty(property,
propertyValue, resultArray);
  }

  return resultArray;
};

THREE.Object3D.prototype.getObjectsByFilterCu
stomProperty = function (
  property = '',
  predicate = () => {},
  resultArray = []
) {
  // check the current object
  if (predicate(this.userData[property]))
resultArray.push(this);

  // check children
  for (var i = 0; i < this.children.length;
i++) {
    var child = this.children[i];

child.getObjectsByFilterCustomProperty(proper
ty, predicate, resultArray);
  }

  return resultArray;
};

export class CanvasContext {
  constructor() {
    this.canvas =
document.getElementById('canvas');
    this.renderer = new THREE.WebGLRenderer({
      canvas: this.canvas,
      antialias: true,
    });

    this.animate = this.animate.bind(this);
    this.addObject =
this.addObject.bind(this);
    this.createSurface =
this.createSurface.bind(this);
    this.removeObject =
this.removeObject.bind(this);

    this.renderer.setPixelRatio(5);

    /**@type {THREE.Scene} */
    this.scene = new THREE.Scene();

    this.scene.background = new
THREE.Color(0xf0f0f0);
```

```
    this.scene.add(new
THREE.AmbientLight(0x808080));
    const light = new
THREE.DirectionalLight(0xffffff, 1);
    light.position.set(1, 1, 1);
    this.scene.add(light);

    this.camera = new
THREE.PerspectiveCamera(
      60,
      this.canvas.clientWidth /
this.canvas.clientHeight,
      0.1,
      10000
    );
    this.camera.position.set(0, 0, 1000);

    this.orbitControls = new OrbitControls(
      this.camera,
      this.renderer.domElement
    );

    this.animate();
  }

  animate() {
    this.render();
    requestAnimationFrame(this.animate);
  }

  resizeRendererToDisplaySize(renderer) {
    const canvas = renderer.domElement;
    const width = canvas.clientWidth;
    const height = canvas.clientHeight;
    const needResize = canvas.width !== width
|| canvas.height !== height;
    if (needResize) {
      renderer.setSize(width, height, false);
    }
    return needResize;
  }

  render() {
    this.renderer.render(this.scene,
this.camera);
  }

  addObject(object) {
    this.scene.add(object);
  }

  removeObject(object) {
    object.parent.remove(object);
  }

  createSurface(
    nsControlPoints = [
      [
        new THREE.Vector3(-200, -200, 0),
        new THREE.Vector3(-200, -100, 0),
        new THREE.Vector3(-200, 100, 0),
        new THREE.Vector3(-200, 200, 0),
      ],
      [
        new THREE.Vector3(0, -200, 0),
        new THREE.Vector3(0, -100, 0),
        new THREE.Vector3(0, 100, 0),
        new THREE.Vector3(0, 200, 0),
      ],
      [
        new THREE.Vector3(200, -200, 0),
        new THREE.Vector3(200, -100, 0),
        new THREE.Vector3(200, 100, 0),
        new THREE.Vector3(200, 200, 0),
      ],
    ],
    position
  ) {
    // NURBS surface
    const degree1 = 2;
    const degree2 = 3;
    const knots1 = [0, 0, 0, 1, 1, 1];
```

```
    const knots2 = [0, 0, 0, 0, 1, 1, 1, 1];
    const nurbsSurface = new NURBSSurface(
      degree1,
      degree2,
      knots1,
      knots2,
      nsControlPoints
    );

    const textureImage =
require('../../textures/uv_grid_opengl.jpg');
    const map = new
THREE.TextureLoader().load(textureImage);
    map.wrapS = map.wrapT =
THREE.RepeatWrapping;
    map.anisotropy = 16;

    function getSurfacePoint(u, v, target) {
      return nurbsSurface.getPoint(u, v,
target);
    }

    const geometry = new
THREE.ParametricBufferGeometry(
      getSurfacePoint,
      20,
      20
    );
    const material = new
THREE.MeshLambertMaterial({
      map: map,
      side: THREE.DoubleSide,
    });
    const object = new THREE.Mesh(geometry,
material);
    object.position.set(position.x,
position.y, position.z);

    object.userData.type = 'Surface';

    for (var i = 0; i <
nsControlPoints.length; i++) {
      for (var j = 0; j <
nsControlPoints[i].length; j++) {
        var cpGeometry = new
THREE.SphereGeometry(10);
        var cpMaterial = new
THREE.MeshLambertMaterial({ color: 0xff0000
});
        var cpMesh = new
THREE.Mesh(cpGeometry, cpMaterial);

        cpMesh.position.set(
          nsControlPoints[i][j].x,
          nsControlPoints[i][j].y,
          nsControlPoints[i][j].z
        );

cpMesh.userData.cpMarkerOfSurfaceWithId =
object.id;
        cpMesh.userData.type =
'ControlPointMarker';

        cpMesh.visible = false;

        object.add(cpMesh);
      }
    }

    return object;
  }

  enableCamera() {
    this.orbitControls.enabled = true;
  }

  disableCamera() {
    this.orbitControls.enabled = false;
  }
}
```

editor/context/clipboard.js

```
export class ClipboardContext {
  constructor() {
    this.savedObjects = [];
  }

  setObjects(objects) {
    this.savedObjects = objects;
  }

  getObjects() {
    return this.savedObjects;
  }
}
```

editor/context/dragging.js

```
export class DraggingContext {
  constructor() {
    this.draggingObject = null;
  }

  setDraggingObject(object) {
    this.draggingObject = object;
  }

  clearDraggingObject() {
    this.draggingObject = null;
  }
}
```

editor/context/selection.js

```
import { eventEmitter } from
'../init/initContexts';

export class SelectionContext {
  constructor() {
    /**
     * @private
     * @type {[]}
     */
    this.selectedElements = [];
  }

  addElementToSelection(element) {
    this.selectedElements.push(element);

    if (this.selectedElements.length === 1) {

eventEmitter.emit('selection.singleObjectSele
cted', {
        term:
this.selectedElements[0].userData.term,
        isGroup:
this.selectedElements[0].type === 'Object3D',
      });
    } else if (this.selectedElements.length >
1) {

eventEmitter.emit('selection.multipleObjectsS
elected', {});
    } else {

eventEmitter.emit('selection.noObjectsSelecte
d', {});
    }
  }

  removeElementFromSelection(removePredicate)
{
    this.selectedElements =
this.selectedElements.filter(
      (el) => !removePredicate(el)
    );

    if (this.selectedElements.length === 1) {
```

```
eventEmitter.emit('selection.singleObjectSele
cted', {
        term:
this.selectedElements[0].userData.term,
        isGroup:
this.selectedElements[0].type === 'Object3D',
      });
    } else if (this.selectedElements.length >
1) {

eventEmitter.emit('selection.multipleObjectsS
elected', {});
    } else {

eventEmitter.emit('selection.noObjectsSelecte
d', {});
    }
  }
```

```
  isObjectSelected(predicate = () => false) {
    return
this.selectedElements.some(predicate);
  }

  getSelectedObjects() {
    return this.selectedElements;
  }

  clearSelection() {
    this.selectedElements = [];


eventEmitter.emit('selection.noObjectsSelecte
d', {});
  }
}
```

editor/context/tools.js

```
import { TOOLS } from '../constants/tools';
import { eventEmitter } from
'../init/initContexts';

export class ToolsContext {
  constructor() {
    /**
     * @private
     */
    this.currentTool = null;
  }

  /**
   * @param {string} tool
   */
  selectCurrentTool(tool) {
    this.currentTool = tool;

    eventEmitter.emit('tools.toolChanged', {
tool });
  }

  /**
   * @returns {boolean}
   */
  isCurrentToolForDrawing() {
    return (
      this.currentTool !== TOOLS.SELECTION &&
      this.currentTool !==
TOOLS.BUILD_L_SYSTEM_RESULT
    );
  }

  /**
   * @returns {string}
   */
  getCurrentTool() {
    return this.currentTool;
  }
}
```

editor/context/ui.js

```
export class UIContext {
  constructor() {}

  /**
   *
   * @param {string} elementId
   * @param {MouseEvent} eventHandler
   */
  mountClickEventToElementById(elementId,
eventHandler) {
    const element =
document.getElementById(elementId);
    element.onclick = eventHandler;
  }

  /**
   *
   * @param {string} elementId
   * @param {KeyboardEvent} eventHandler
   */
  mountOnInputEventToElementById(elementId,
eventHandler) {
    const element =
document.getElementById(elementId);
    element.oninput = eventHandler;
  }
}
```

editor/helpers/ui/tools.js

```
import { TOOLS } from
'../../constants/tools';

export function
updateUiOnToolChanged(targetTool) {
  const toolElem =
document.getElementById(getToolIdByToolType(t
argetTool));
  // unselect previously selected tool
  const toolsList =
document.getElementById('tools-list');
  for (let i = 0; i <
toolsList.children.length; i++) {
    const child = toolsList.children[i];
    if (child.classList.contains('current-
tool')) {
      child.classList.remove('current-tool');
    }
  }

  toolElem.classList.add('current-tool');
}

export function getToolIdByToolType(tool) {
  switch (tool) {
    case TOOLS.SELECTION:
      return 'selection-tool-button';
    case TOOLS.RECTANGLE:
      return 'rect-tool-button';
    case TOOLS.IMAGE:
      return 'image-tool-button';

    default:
      break;
  }
}
```

editor/init/canvas.js

```
import { eventEmitter } from
'./initContexts';

/**
 *
 * @param {import('./initContexts').Contexts}
contexts
 */
export function
canvasContextPreconfigurations({
canvasContext }) {

canvasContext.canvas.addEventListener('click'
, (e) => {
    eventEmitter.emit('canvas.click', e);
  });
```

```
canvasContext.canvas.addEventListener('pointe
rdown', (e) => {
    e.preventDefault();
    eventEmitter.emit('canvas.mousedown', e);
  });



canvasContext.canvas.addEventListener('pointe
rup', (e) => {
    e.preventDefault();
    eventEmitter.emit('canvas.mouseup', e);
  });



canvasContext.canvas.addEventListener('pointe
rmove', (e) => {
    e.preventDefault();
    eventEmitter.emit('canvas.mousemove', e);
  });

  document.addEventListener('keydown', (e) =>
{
    if (e.ctrlKey && e.key === 'c') {
      eventEmitter.emit('canvas.ctrlC', e);
    }
  });

  document.addEventListener('keydown', (e) =>
{
    if (e.ctrlKey && e.key === 'v') {
      eventEmitter.emit('canvas.ctrlV', e);
    }
  });
}
```

editor/init/initContexts.js

```
import { CanvasContext } from
'../context/canvas';
import { UIContext } from '../context/ui';
import { ToolsContext } from
'../context/tools';
import { MouseContext } from
'../context/mouse';
import { DrawingContext } from
'../context/drawing';
import { toolsContextPreconfigurations } from
'./tools';
import { uiContextPreconfigurations } from
'./ui';
import { configureMouseContextSubscriptions }
from './crossContextEventHandlers/mouse';
import { configureUIContextSubscriptions }
from './crossContextEventHandlers/ui';
import { configureCanvasContextSubscriptions
} from './crossContextEventHandlers/canvas';
import { SelectionContext } from
'../context/selection';
import {
configureSelectionContextSubscriptions } from
'./crossContextEventHandlers/selection';
import { EventEmitter } from 'eventemitter3';
import { mouseContextPreconfigurations } from
'./mouse';
import { canvasContextPreconfigurations }
from './canvas';
import { DraggingContext } from
'../context/dragging';
import {
configureDraggingContextSubscriptions } from
'./crossContextEventHandlers/dragging';
import { ClipboardContext } from
'../context/clipboard';
import {
configureClipboardContextSubscriptions } from
'./crossContextEventHandlers/clipboard';
import * as THREE from 'three';

export const eventEmitter = new
EventEmitter();
```

```
export function initContexts() {
  const canvasContext = new CanvasContext();
  const toolsContext = new ToolsContext();
  const selectionContext = new
SelectionContext();
  const drawingContext = new
DrawingContext();
  const mouseContext = new MouseContext();
  const uiContext = new UIContext();
  const draggingContext = new
DraggingContext();
  const clipboardContext = new
ClipboardContext();

  initialAppSetup({
    canvasContext,
    toolsContext,
    drawingContext,
    mouseContext,
    uiContext,
    selectionContext,
    draggingContext,
    clipboardContext,
  });
}

/**
 * @typedef {Object} Contexts
 * @property {CanvasContext}
contexts.canvasContext
 * @property {SelectionContext}
contexts.selectionContext
 * @property {DrawingContext}
contexts.drawingContext
 * @property {ToolsContext}
contexts.toolsContext
 * @property {MouseContext}
contexts.mouseContext
 * @property {UIContext} contexts.uiContext
 * @property {DraggingContext}
contexts.draggingContext
 * @property {ClipboardContext}
contexts.clipboardContext
 *
 * @param {Contexts} contexts
 */
function initialAppSetup(contexts) {
  // TODO: check there is no order problems
with this approach
  // cross context event listeners
subscription
  //
configureLayersContextSubscriptions(contexts)
;

configureMouseContextSubscriptions(contexts);
  configureUIContextSubscriptions(contexts);

configureCanvasContextSubscriptions(contexts)
;

  // contexts own pre-configurations.
  canvasContextPreconfigurations(contexts);
  mouseContextPreconfigurations(contexts);
  uiContextPreconfigurations(contexts);
  toolsContextPreconfigurations(contexts);

configureSelectionContextSubscriptions(contex
ts);

configureDraggingContextSubscriptions(context
s);

configureClipboardContextSubscriptions(contex
ts);

  //
prepareSerpinskiTriangleExample(contexts);
}
```

```
/**
 *
 * @param {Contexts} contexts
 */
function
prepareSerpinskiTriangleExample(contexts) {
  const el1 =
contexts.canvasContext.createSurface(
    undefined,
    new THREE.Vector3(-500, 0, 0)
  );
  el1.userData.term = 'A';
  contexts.canvasContext.addObject(el1);

  const group = new THREE.Object3D();
  group.userData.term = 'forA';

  const el2 =
contexts.canvasContext.createSurface(
    undefined,
    new THREE.Vector3(0, 0, 0)
  );
  el2.userData.term = 'A';
  group.add(el2);

  const el3 =
contexts.canvasContext.createSurface(
    undefined,
    new THREE.Vector3(500, 0, 0)
  );
  el3.userData.term = 'A';
  group.add(el3);

  const el4 =
contexts.canvasContext.createSurface(
    undefined,
    new THREE.Vector3(250, 500, 0)
  );
  el4.userData.term = 'A';
  group.add(el4);

  contexts.canvasContext.addObject(group);
}
```

editor/init/tools.js

```
import { TOOLS } from '../constants/tools';

/**
 *
 * @param {import('./initContexts').Contexts}
contexts
 */
export function
toolsContextPreconfigurations({ toolsContext
}) {
  // choose selection tool as current by
default

toolsContext.selectCurrentTool(TOOLS.SELECTIO
N);
}
```

editor/init/ui.js

```
import { TOOLS } from '../constants/tools';
import { eventEmitter } from
'./initContexts';

/**
 *
 * @param {import('./initContexts').Contexts}
contexts
 */
export function uiContextPreconfigurations({
toolsContext, uiContext }) {
  // mount tools selection buttons
```

```
uiContext.mountClickEventToElementById('selec
tion-tool-button', () => {

toolsContext.selectCurrentTool(TOOLS.SELECTIO
N);
  });


uiContext.mountClickEventToElementById('rect-
tool-button', () => {

toolsContext.selectCurrentTool(TOOLS.RECTANGL
E);
  });


uiContext.mountClickEventToElementById('image
-tool-button', () => {

toolsContext.selectCurrentTool(TOOLS.IMAGE);
  });

  uiContext.mountOnInputEventToElementById(
    'selected-elem-classname-input',
    (e) => {

eventEmitter.emit('ui.objectTermInputValueCha
nged', {
        term: e.target.value,
      });
    }
  );
```

```
uiContext.mountClickEventToElementById('group
-elements-tool', () => {

eventEmitter.emit('ui.groupObjectsButtonClick
ed');
  });


uiContext.mountClickEventToElementById('ungro
up-elements-tool', () => {

eventEmitter.emit('ui.ungroupButtonClicked');
  });


uiContext.mountClickEventToElementById('lsyst
em-tool-button', () => {
    const iterationsCount = Number.parseInt(
      prompt('Count of iterations: ', '1')
    );

    if (iterationsCount) {

eventEmitter.emit('ui.generateLSystemResultBu
ttonClicked', {
        iterationsCount,
      });
    }
  });
}
```

editor/init/crossContextEventHandlers/canvas.js

```
import { TOOLS } from
'../../constants/tools';
import { eventEmitter } from
'../initContexts';
import * as THREE from 'three';
```

```
import { OrbitControls } from
'three/examples/jsm/controls/OrbitControls.js
';

/**
```

```
 * @param
{import('../initContexts').Contexts} contexts
 */
export function
configureCanvasContextSubscriptions({
  canvasContext,
  toolsContext,
}) {
  var _plane = new THREE.Plane();
  var _raycaster = new THREE.Raycaster();
  var _mouse = new THREE.Vector2();
  var _offset = new THREE.Vector3();
  var _intersection = new THREE.Vector3();
  var _camera = canvasContext.camera;
  const canvasBounds = canvasContext.renderer
    .getContext()
    .canvas.getBoundingClientRect();

  // subscribe on tool change event, to
disable dragging elements, if we tries to
draw(on them)
  eventEmitter.on('tools.toolChanged', ({
tool }) => {
    if (tool === TOOLS.SELECTION) {
      canvasContext.enableCamera();
    } else {
      canvasContext.disableCamera();
    }
  });

  eventEmitter.on('canvas.mousedown',
(mouseEvent) => {
    _mouse.x =
      ((mouseEvent.clientX -
canvasBounds.left) /
        (canvasBounds.right -
canvasBounds.left)) *
        2 -
      1;

    _mouse.y =
      -(
        (mouseEvent.clientY -
canvasBounds.top) /
        (canvasBounds.bottom -
canvasBounds.top)
      ) *
        2 +
      1;
    _raycaster.setFromCamera(_mouse,
_camera);

    var intersects =
_raycaster.intersectObjects(
      canvasContext.scene.children,
      true
    );

    if (intersects.length > 0) {
      var object = intersects[0].object;
      while (object.parent.type ===
'Object3D') {
        object = object.parent;
      }

eventEmitter.emit('scene.object.mousedown', {
object });
      if (mouseEvent.ctrlKey) {

eventEmitter.emit('scene.object.mousedownWith
Ctrl', { object });
      }

      if
(_raycaster.ray.intersectPlane(_plane,
_intersection)) {

_offset.copy(_intersection).sub(object.positi
on);
      }
    } else {
      eventEmitter.emit('scene.mousedown');
```

```
    }
  });

  eventEmitter.on(

'dragging.mousemoveWithAvailableDraggingObjec
t',
    ({ mouseEvent }) => {
      _mouse.x =
        ((mouseEvent.clientX -
canvasBounds.left) /
          (canvasBounds.right -
canvasBounds.left)) *
          2 -
        1;

      _mouse.y =
        -(
          (mouseEvent.clientY -
canvasBounds.top) /
          (canvasBounds.bottom -
canvasBounds.top)
        ) *
          2 +
        1;

      _raycaster.setFromCamera(_mouse,
_camera);

      if
(_raycaster.ray.intersectPlane(_plane,
_intersection)) {

eventEmitter.emit('scene.intersectPlaneMove',
{
          newPosition:
_intersection.sub(_offset),
        });
      }
    }
  );

  eventEmitter.on(

'dragging.mousemoveWithoutAvailableDraggingOb
ject',
    (mouseEvent) => {
      _mouse.x =
        ((mouseEvent.clientX -
canvasBounds.left) /
          (canvasBounds.right -
canvasBounds.left)) *
          2 -
        1;

      _mouse.y =
        -(
          (mouseEvent.clientY -
canvasBounds.top) /
          (canvasBounds.bottom -
canvasBounds.top)
        ) *
          2 +
        1;

      _raycaster.setFromCamera(_mouse,
_camera);

      var intersects =
_raycaster.intersectObjects(
        canvasContext.scene.children,
        true
      );

      if (intersects.length > 0) {
        var object = intersects[0].object;

        _plane.setFromNormalAndCoplanarPoint(

_camera.getWorldDirection(_plane.normal),
          object.position
        );
```

```
      }
    }
  );

  eventEmitter.on('scene.mousedown', () => {
    const cpMarkers =
canvasContext.scene.getObjectsByCustomPropert
y(
      'type',
      'ControlPointMarker'
    );
    cpMarkers.forEach((cp) => {
      cp.visible = false;
    });

    if (toolsContext.getCurrentTool() ===
TOOLS.RECTANGLE) {
      canvasContext.addObject(

canvasContext.createSurface(undefined, new
THREE.Vector3(0, 0, 0))
      );
    }
  });

  eventEmitter.on('scene.object.mousedown',
({ object }) => {
    object.children.forEach((child) => {
      if (child.userData.type ===
'ControlPointMarker') {
        child.visible = true;
      }
    });
  });

  eventEmitter.on('dragging.started', () => {
    canvasContext.disableCamera();
  });

  eventEmitter.on('dragging.ended', ({ object
}) => {
    canvasContext.enableCamera();

    if (object.userData.type ===
'ControlPointMarker') {
      const res =
canvasContext.scene.getObjectsByCustomPropert
y(
        'cpMarkerOfSurfaceWithId',

object.userData.cpMarkerOfSurfaceWithId
      );
      const cpArray = [
        [res[0].position, res[1].position,
res[2].position, res[3].position],
        [res[4].position, res[5].position,
res[6].position, res[7].position],
        [res[8].position, res[9].position,
res[10].position, res[11].position],
      ];
      //find old surface
      const oldSurface =
canvasContext.scene.getObjectById(

object.userData.cpMarkerOfSurfaceWithId
      );

      //create new one with updates made
      canvasContext.addObject(
        canvasContext.createSurface(cpArray,
oldSurface.position)
      );
      //remove old surface
      canvasContext.removeObject(oldSurface);
    }
  });


eventEmitter.on('selection.selectedObjectsGro
uped', ({ group }) => {
    canvasContext.addObject(group);
  });
```

```
eventEmitter.on('selection.selectedGroupUngro
uped', ({ group }) => {
    const children = [...group.children];

    children.forEach((element) => {
      canvasContext.scene.attach(element);
    });

    canvasContext.removeObject(group);
  });

  eventEmitter.on(
    'ui.LSystemResultContainerGenerated',
    ({ canvasElement, iterationsCount }) => {
      const canvas = canvasElement;
      const renderer = new
THREE.WebGLRenderer({
        canvas: canvas,
        antialias: true,
      });

      renderer.setPixelRatio(5);

      const LSystemResultScene = new
THREE.Scene();

      LSystemResultScene.background = new
THREE.Color(0xf0f0f0);

      LSystemResultScene.add(new
THREE.AmbientLight(0x808080));
      const light = new
THREE.DirectionalLight(0xffffff, 1);
      light.position.set(1, 1, 1);
      LSystemResultScene.add(light);

      const camera = new
THREE.PerspectiveCamera(
        60,
        canvas.clientWidth /
canvas.clientHeight,
        0.1,
        10000
      );
      camera.position.set(0, 0, 1000);

      const orbitControls = new
OrbitControls(camera, renderer.domElement);
      orbitControls.enabled = true;

      animate();

      function animate() {
        render();
        requestAnimationFrame(animate);
      }

      function
resizeRendererToDisplaySize(renderer) {
        const canvas = renderer.domElement;
        const width = canvas.clientWidth;
        const height = canvas.clientHeight;
        const needResize = canvas.width !==
width || canvas.height !== height;
        if (needResize) {
          renderer.setSize(width, height,
false);
        }
        return needResize;
      }

      function render() {
        if
(resizeRendererToDisplaySize(renderer)) {
          const canvas = renderer.domElement;
          camera.aspect = canvas.clientWidth
/ canvas.clientHeight;
          camera.updateProjectionMatrix();
        }
```

```
        renderer.render(LSystemResultScene,
camera);
      }

      const axioms =
canvasContext.scene.getObjectsByFilterCustomP
roperty(
        'term',
        (term) => {
          return !!term &&
!term.includes('for');
        }
      );
      const axiomObj = (axioms.length > 0 &&
axioms[0].clone()) || null;

      const resultAsGroup = new
THREE.Object3D();
      LSystemResultScene.add(resultAsGroup);

      if (axiomObj) {
        resultAsGroup.add(axiomObj);

        for (let i = 0; i < iterationsCount;
++i) {
          const resultSceneChildren =
[...resultAsGroup.children];


resultSceneChildren.forEach((element) => {
            const potentialReplacements =

canvasContext.scene.getObjectsByCustomPropert
y(
                'term',
                `for${element.userData.term}`
              );

            const potentialReplacementElement
=
              potentialReplacements.length >
0
                ?
potentialReplacements[0].clone()
                : null;

            if (potentialReplacementElement)
{

resultAsGroup.add(potentialReplacementElement
);


potentialReplacementElement.position.copy(
                new THREE.Vector3(
                  element.position.x,
                  element.position.y,
                  element.position.z
                )
              );

              var box = new THREE.Box3();
              var size1 = box

.setFromObject(potentialReplacementElement)
                .getSize(size1);
              var size2 =
box.setFromObject(element).getSize(size2);


potentialReplacementElement.scale.divide(
                new THREE.Vector3(
                  size1.x / size2.x,
                  size1.x / size2.x,
                  size1.z / size2.z
                )
              );

              if
(potentialReplacementElement.type ===
'Object3D') {
```

```
                const arr =
[...potentialReplacementElement.children];
                arr.forEach((element) => {

resultAsGroup.attach(element);

resultAsGroup.remove(potentialReplacementElem
ent);
                });
              }

              resultAsGroup.remove(element);
            }
          });
        }

        var box = new THREE.Box3();
        box.setFromObject(resultAsGroup);

        resultAsGroup.position.setX(
          resultAsGroup.position.x -
box.getCenter().x
        );
        resultAsGroup.position.setY(
          resultAsGroup.position.y -
box.getCenter().y
        );

        // resultAsGroup.position.copy(new
THREE.Vector3(0, 0, 0));
        // console.log(
        //   'file: canvas.js ~ line 329 ~
resultAsGroup',
        //   resultAsGroup
        // );
      }
    }
  );

  eventEmitter.on('clipboard.objects.ctrlV',
({ objects }) => {
    objects.forEach((obj) => {
      canvasContext.addObject(obj);
    });
  });
}
```

editor/init/crossContextEventHandlers/clipboard.js

```
import { eventEmitter } from
'../initContexts';

/**
 * @param
{import('../initContexts').Contexts} contexts
 */
export function
configureClipboardContextSubscriptions({
clipboardContext }) {
  eventEmitter.on('selection.objects.ctrlC',
({ objects }) => {
    clipboardContext.setObjects(objects);
  });

  eventEmitter.on('canvas.ctrlV', () => {
    const savedObjects =
clipboardContext.getObjects();
    if (savedObjects.length > 0) {

eventEmitter.emit('clipboard.objects.ctrlV',
{ objects: savedObjects });
    }
  });
}
```

editor/init/crossContextEventHandlers/dragging.js

```
import { eventEmitter } from
'../initContexts';

/**
```

```
 * @param
{import('../initContexts').Contexts} contexts
 */
export function
configureDraggingContextSubscriptions({
draggingContext }) {
  eventEmitter.on('scene.object.mousedown',
({ object }) => {

draggingContext.setDraggingObject(object);
    eventEmitter.emit('dragging.started');
  });

  eventEmitter.on('scene.intersectPlaneMove',
({ newPosition }) => {

draggingContext.draggingObject.position.copy(
newPosition);
  });

  eventEmitter.on('canvas.mousemove', (e) =>
{
    if (draggingContext.draggingObject) {

eventEmitter.emit('dragging.mousemoveWithAvai
lableDraggingObject', {
        mouseEvent: e,
        object:
draggingContext.draggingObject,
      });
    } else {

eventEmitter.emit('dragging.mousemoveWithoutA
vailableDraggingObject', e);
    }
  });

  eventEmitter.on('canvas.mouseup', () => {
    if (draggingContext.draggingObject) {
      eventEmitter.emit('dragging.ended', {
        object:
draggingContext.draggingObject,
      });
      draggingContext.clearDraggingObject();
    }
  });
}
```

editor/init/crossContextEventHandlers/selection.js

```
import { eventEmitter } from
'../initContexts';
import * as THREE from 'three';

/**
 * @param
{import('../initContexts').Contexts} contexts
 */
export function
configureSelectionContextSubscriptions({
selectionContext }) {

eventEmitter.on('ui.groupObjectsButtonClicked
', () => {
    const selectedObjects =
selectionContext.getSelectedObjects();
    if (selectedObjects.length > 1) {
      const group = new THREE.Object3D();
      selectedObjects.forEach((obj) => {
        group.add(obj);
      });

eventEmitter.emit('selection.selectedObjectsG
rouped', { group: group });
      selectionContext.clearSelection();
    }
  });

  eventEmitter.on('ui.ungroupButtonClicked',
() => {
    const selectedObjects =
selectionContext.getSelectedObjects();
```

```
    if (selectedObjects.length === 1) {
      const group = selectedObjects[0];
      // const groupedObjects =
group.children;

eventEmitter.emit('selection.selectedGroupUng
rouped', { group: group });
      selectionContext.clearSelection();
    }
  });



eventEmitter.on('scene.object.mousedownWithCt
rl', ({ object }) => {
    if (
      selectionContext.isObjectSelected((obj)
=> {
        return obj.id === object.id;
      })
    ) {

selectionContext.removeElementFromSelection((
obj) => {
        return obj.id === object.id;
      });
    } else {

selectionContext.addElementToSelection(object
);
    }
  });



eventEmitter.on('ui.objectTermInputValueChang
ed', ({ term }) => {
    const selectedObjects =
selectionContext.getSelectedObjects();
    if (selectedObjects.length === 1) {
      const obj = selectedObjects[0];
      obj.userData.term = term;
    }
  });

  eventEmitter.on('canvas.ctrlC', () => {
    const selectedObjects =
selectionContext.getSelectedObjects();
    if (selectedObjects.length > 0) {
      const selectedObjectsCopies =
selectedObjects.map((obj) => {
        return obj.clone();
      });

eventEmitter.emit('selection.objects.ctrlC',
{
        objects: selectedObjectsCopies,
      });
    }
  });
}
```

editor/init/crossContextEventHandlers/ui.js

```
import { updateUiOnToolChanged } from
'../../helpers/ui/tools';
import { eventEmitter } from
'../initContexts';

/**
 *
 * @param
{import('../initContexts').Contexts} contexts
 */
export function
configureUIContextSubscriptions({}) {
  eventEmitter.on('tools.toolChanged', ({
tool }) => {
    updateUiOnToolChanged(tool);
  });



eventEmitter.on('selection.singleObjectSelect
ed', ({ term, isGroup }) => {
```

```
    const elemClassContainer =
document.getElementById(
      'single-selected-elem-attributes'
    );
    const groupElementsButton =
document.getElementById('group-elements-
tool');
    const ungroupElementsButton =
document.getElementById(
      'ungroup-elements-tool'
    );

    elemClassContainer.style.display =
'flex';

    // update class input value, according to
selected item
    const objectTermInput =
document.getElementById(
      'selected-elem-classname-input'
    );
    objectTermInput.value = term || '';

    groupElementsButton.style.display =
'none';
    if (isGroup) {
      ungroupElementsButton.style.display =
'flex';
    }
  });


eventEmitter.on('selection.multipleObjectsSel
ected', () => {
    const elemClassContainer =
document.getElementById(
      'single-selected-elem-attributes'
    );
    const groupElementsButton =
document.getElementById('group-elements-
tool');
    const ungroupElementsButton =
document.getElementById(
      'ungroup-elements-tool'
    );

    elemClassContainer.style.display =
'none';

    groupElementsButton.style.display =
'flex';
    ungroupElementsButton.style.display =
'none';
  });


eventEmitter.on('selection.noObjectsSelected'
, () => {
    const elemClassContainer =
document.getElementById(
      'single-selected-elem-attributes'
    );
    const groupElementsButton =
document.getElementById('group-elements-
tool');
    const ungroupElementsButton =
document.getElementById(
      'ungroup-elements-tool'
    );

    elemClassContainer.style.display =
'none';

    groupElementsButton.style.display =
'none';
    ungroupElementsButton.style.display =
'none';
  });

  eventEmitter.on(
    'ui.generateLSystemResultButtonClicked',
    ({ iterationsCount }) => {
```

```
      const resultContainer =
document.createElement('div');
      resultContainer.setAttribute('id', 'l-
system-result-container');
      const closeButton =
document.createElement('div');
      closeButton.setAttribute('class',
'close-button');
      closeButton.innerHTML = 'Close';
      closeButton.addEventListener('click',
() => {

appContainer.removeChild(resultContainer);
      });
      const resultCanvas =
document.createElement('canvas');
      resultCanvas.setAttribute('id', 'l-
system-result-canvas');

resultContainer.appendChild(closeButton);

resultContainer.appendChild(resultCanvas);
      const appContainer =
document.getElementById('app-container');

appContainer.appendChild(resultContainer);



eventEmitter.emit('ui.LSystemResultContainerG
enerated', {
        canvasElement: resultCanvas,
        iterationsCount,
      });
    }
  );
}
```

editor/constants/tools.js

```
export const TOOLS = {
  SELECTION: 'selection',
  LINE: 'line',
  ELLIPSE: 'ellipse',
  RECTANGLE: 'rectangle',
  IMAGE: 'image',
  BUILD_L_SYSTEM_RESULT:
'build_l_system_result',
};
```

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport"
content="width=device-width, initial-
scale=1.0" />
    <title>Fractal Editor</title>
  </head>
  <body>
    <div id="app-container" class="app-
container">
      <div class="action-tools"><div
class="app-name">Fractal Editor</div></div>
      <div class="drawing">
        <div id="tools-list" class="tools-
list">
          <div id="selection-tool-button"
class="button select"></div>
          <div id="rect-tool-button"
class="button rect"></div>
          <div id="image-tool-button"
class="button image"></div>
          <div id="lsystem-tool-button"
class="button lsystem">L</div>
        </div>

        <div id="svgcanvas-container"
class="svgcanvas-container">
          <canvas id="canvas"></canvas>
```

```
        </div>
      </div>

      <div class="selected-element-
attributes">
        <div id="single-selected-elem-
attributes">
          <div class="elem-class-
name">Class:</div>
          <input
            type="text"
            name="classname"
            id="selected-elem-classname-
input"
          />
        </div>
        <div id="group-elements-tool"
class="button group-elements"></div>
        <div id="ungroup-elements-tool"
class="button ungroup-elements"></div>
      </div>
    </div>
  </body>
</html>
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

АТ «УКРАЇНСЬКА ЗАЛІЗНИЦЯ»

ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ ІМЕНІ АКАДЕМІКА В. ЛАЗАРЯНА

УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФРАСТРУКТУРИ ТА ТЕХНОЛОГІЙ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ АВТОМОБІЛЬНО-ДОРОЖНІЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-ТЕЛЕКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ ТА КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ

ЗБІРНИК ТЕЗ ДОПОВІДЕЙ
81 Всеукраїнської науково-технічної конференції
молодих учених, магістрантів та студентів
«НАУКА І СТАЛИЙ РОЗВИТОК ТРАНСПОРТУ»
28 жовтня 2021 року

INFORMATION-TELECOMMUNICATION TECHNOLOGY TA COMPUTER MODELING

CONFERENCE PROCEEDINGS
81th all Ukrainian Scientific and Technical Conference
of young scientists, masters and students
"SCIENCE AND SUSTAINABLE DEVELOPMENT OF TRANSPORT"
October 28, 2021

ДНІПРО
2021

УДК 658.512.2:681.3.06

Інформаційно-телекомунікаційні технології та комп’ютерне моделювання [електронний ресурс]: збірник тез доповідей секції 81 Всеукраїнської науково-технічної конференції молодих учених, магістрантів та студентів «Наука і сталий розвиток транспорту» 28 жовтня 2021 р. – Дніпро: Дніпровський нац. ун-т залізн. трансп. ім. акад. В. Лазаряна, 2021. – 29 с. – URL: http://ndch.diit.edu.ua/upload/%D0%9A%D0%BE%D0%BD%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D0%B8/2021/81_All_UA_ST_Conference_of_YSMS_SSD_of_Transport/Infotelecom_and_Computer_Simulation_2021.pdf

У збірнику тез доповідей подано результати досліджень здобувачів вищої освіти і молодих учених, які присвячено питанням інформаційно-телекомунікаційних технологій та комп’ютерного моделювання, їх розвитку і поширенню сфер застосування. Тези доповідей подано в рамках 81 Всеукраїнської науково-технічної конференції молодих учених, магістрантів та студентів «Наука і сталий розвиток транспорту», яку проведено 28 жовтня 2021 року у Дніпровському національному університеті залізничного транспорту імені академіка В. Лазаряна.

Збірник тез доповідей призначено для здобувачів вищої освіти і молодих учених.

Текст тез доповідей учасників конференції подано в авторській редакції.

Офіційна наукова конференція здобувачів вищої освіти та молодих учених:
– Лист Державної наукової установи «Інститут модернізації змісту освіти» від 19.01.2021 р. № 22.1/10-83 «Про Перелік міжнародних, всеукраїнських науково-практичних конференцій здобувачів вищої освіти і молодих учених».

ЗМІСТ

процесів, в основу яких покладено нові інтелектуальні процедури упорядкування послідовностей елементів, замовлень. Як відомо, в математичних методах аналізу та планування також часто виникають завдання, які формально можуть бути зведені до оптимального конструктивного упорядкування елементів певних множин або недетермінованих послідовностей. Призначення таких процедур - отримання визначених структур упорядкування замовлень з урахуванням неоднорідності та складності операцій процесів формування. В роботі розглянуті нові питання щодо застосування інтелектуальних процедур упорядкування, які призначені для реалізації технологій процесів упорядкування мультипослідовностей замовлень, поданих для прикладу завдань розформування-формування (РФ) багатогрупових залізничних составів (БГС) на станціях, що являється одним з найбільш трудомістких етапів процесів переробки потоків вагонів на станціях.

В доповіді представлені результати досліджень та розробки, направлені на створення спеціалізованого математичного, програмного забезпечення, а також розрахунки та аналіз варіантів інтелектуальних процедур, які призначені для вибору оптимальних операторів на етапах процесів упорядкування недетермінованих послідовностей замовлень на основі моделей і методів штучних нейронних мереж. При цьому завдання вибору оптимального оператора перетворення кодів послідовностей замовлень визначається як завдання класифікації, що відбувається при неповній або збуреній початковій інформації. Для реалізації таких завдань класифікації застосовуються моделі та процедури асоціативної пам'яті Хеммінга.

Змістовно завдання розробки інтелектуальних процедур вибору оптимальних операторів перетворень полягає у , а також у програмній реалізації та порівняльному аналізі зазначених процедур при різних структурах шаблонів, які кодують недетерміновані послідовності замовлень на різних етапах процесів їх упорядкування. В доповіді приведено результати щодо визначення раціональної структури та параметрів шаблонів моделей мереж Хеммінга, При цьому також розраховуються оцінки чисельної ефективності та результативності їх застосування у порівнянні з повним перебором всіх можливих варіантів застосування операторів перетворення процесів їх упорядкування. Отримані дані моделювання свідчать про досить високу результативність застосування запропонованих інтелектуальних процедур вибору операторів для упорядкування послідовностей на основі моделей асоціативної пам'яті Хеммінга.

КОНСТРУКТІВНІ ПРОСТОРОВІ ПЕРЕТВОРЕННЯ ДВОВИМІРНИХ ФРАКТАЛІВ

Автор: Мосієнко В.С., студент групи ПЗ2021
Науковий керівник: д. т. н., професор Шинкаренко В.І.
Дніпровський національний університет залізничного
транспорту імені академіка В. Лазаряна

Самоподібність давно привернула увагу науковців і у зв'язку с цим з'явилася потреба в відображенні цього математичного означення графічно, для кращого розуміння того, що описує це поняття.

Метою магістерської роботи є розробка методу та засобів конструктивного формування двовимірних фракталів. За допомогою конструктивно-продукційного моделювання буде досягнута можливість побудови та відображення двовимірних фракталів у просторі, що розширює можливості для більш детального дослідження фракталів та експериментів з ними.

Приклад побудови трикутника Серпинського з використанням початкової версії засобу дано на рис 1. Збільшене зображення побудованого трикутника Серпинського дано на рис.2, який показано на рис. 1 у вигляді фрагменту для побудови цілої фігури.

Додатково буде виконано розширення можливостей конструювання, відповідно до можливих потреб користувачів, наприклад таких як зміна кутів нахилу, зміна кольору, збільшення/зменшення конкретних елементів алфавіту і таке інше.

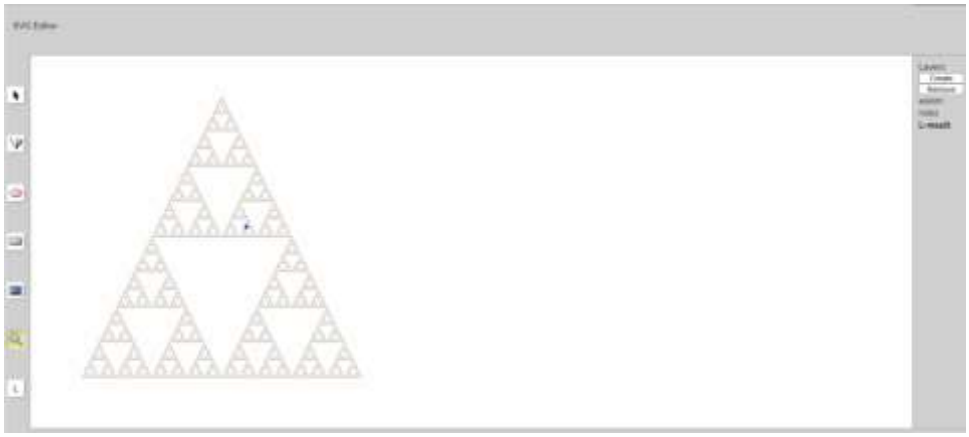

Рис 1. Приклад побудованого трикутника Серпинського з використанням початкової версії засобу

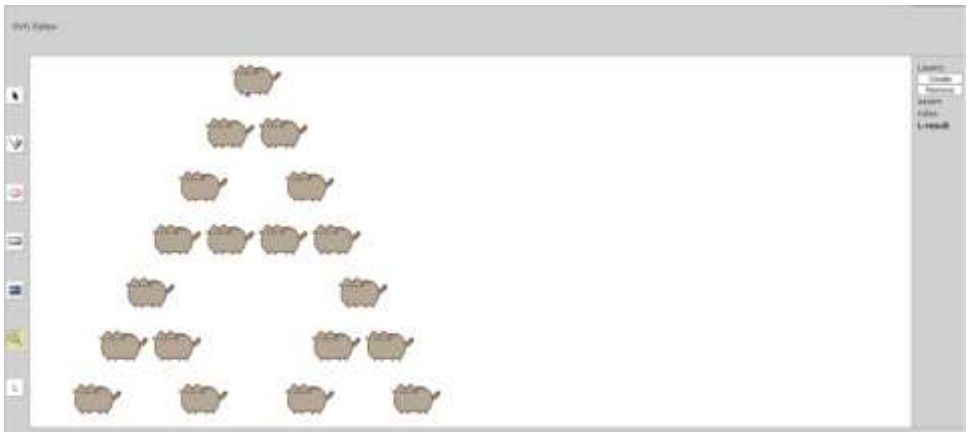

Рис 2. Збільшене зображення побудованого трикутника Серпинського на рис. 1.

ДОСЛІДЖЕННЯ СТРУКТУРНОЇ СХОЖОСТІ ОБ'ЄКТНО-ОРІЄНТОВАНИХ ПРОГРАМ

Автор: Ненахов К. Д., студент групи ПЗ2021
Науковий керівник: к.т.н, доцент кафедри КІТ Куроп'ятник О. С.
Дніпровський національний університет залізничного
транспорту імені академіка В. Лазаряна

Нині при швидкому розвитку інформаційних технологій інтелектуальна власність стає ціннішою, аніж раніше. Проблема пошуку та виявлення фрагментів програмного коду, що були запозичені в іншої людини, залишається однією з найбільш актуальних, складних та важливих проблем. Якщо дві програми мають істотну загальну частину (на рівні структури класів), то можна вважати, що в одній з них міститься плагіат; причому, плагіатор може змінити оригінальну програму вставкою додаткових операторів (синонімізація та виконання несуттєвих дій), перейменуванням змінних, зміною порядку виконання незалежних операторів, розбиттям деяких функцій на дві тощо.

Задачею майбутньої програмної системи є автоматизоване визначення використання у двох програмах спільних архітектурних рішень. Для вирішення цієї задачі найчастіше використовують структурні методи аналізу, що передбачають порівняння програмних кодів з урахуванням їх структури. Такі методи досліджують структуру програми не ізольовано, а як би в контексті, встановлюють взаємозв'язок різних характеристик, їх спільну поведі-