ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «РОЗРОБКА ГРИ «ЗМІЙКА» ЖАНРУ АРКАДА МОВОЮ PYTHON»

Виконав: студент 5 курсу, групи ППЗ–51
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Носач Д.Р.
(прізвище та ініціали)

Керівник Аверічев І.М.
(прізвище та ініціали)

Рецензент Зінченко О.В.
(прізвище та ініціали)

Нормоконтроль Трінтіна Н.А
(прізвище та ініціали)

Київ –2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність - 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
Негоденко О.В.

“_____” _____2023 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

НОСАЧА ДМИТРА РУСЛАНОВИЧА

(прізвище, ім‘я, по батькові)

1. Тема роботи: «РОЗРОБКА ГРИ «ЗМІЙКА» ЖАНРУ АРКАДА МОВОЮ PYTHON»

Керівник роботи: Аверічев І.М., к.е.н., доцент кафедри ІПЗ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “24” лютого 2023 року протокол №26

2. Строк подання студентом роботи «01» червня 2023 року

3. Вхідні дані до роботи:
   3.1. Середовище розробки visual studio 2022
   3.2. Алгоритм дії гри
   3.3. Розроблена гра

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
    4.1. Аналіз та порівняння існуючих прототипів
    4.2. Дослідження програмних засобів для розробки гри
    4.3. Програмна реалізація гри
    4.4. Приклади використання та тестування гри
    4.5. Висновки

5. Перелік графічного матеріалу.
    5.1. Титульний слайд
    5.2. Мета, об’єкт та предмет дослідження
    5.3. Терміни
    5.4. Аналіз існуючих рішень
    5.5. Технічне завдання
    5.6. Програмні засоби реалізації
    5.7. Схема роботи гри
    5.8. Екрани гри
    5.9. Висновки

6. Дата видачі завдання: 25 «лютого» 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	11.04-15.04	Виконано
2	Дослідження існуючих інструментів для автоматизації тестування ПЗ	16.04-18.04	Виконано
3	Проектування архітектури системи	19.04 – 24.04	Виконано
4	Висновки, оформлення роботи	26.04 – 08.05	Виконано
5	Розробка демонстраційних матеріалів	09.05-15.05	Виконано

6	Попередній захист роботи	16.05-01.06	Виконано
7	Подання роботи в деканат	01.06	Виконано

Студент Носач Д.Р.

(підпис) (прізвище та ініціали)

Керівник роботи Аверічев І.М.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Ключові слова: PYTHON, ООП, VISUAL STUDIO, ГРА ЗМІЙКА

Мета дипломного проекту покращення навичок роботи з Python та реалізація ігрового додатку.

Об’єкт дослідження - розробка ігрових додатків та особливості мови Python

Предмет дослідження - ігровий додаток створений за допомогою Python

Python — це інтерпретована об’єктно-орієнтована мова програмування високого рівня загального призначення. Подібно до PERL, Python є мовою програмування, популярною серед досвідчених програмістів C++ і Java.

Працюючи на Python, користувачі можуть інтерпретувати оператори в кількох операційних системах, включаючи системи на основі UNIX, Mac OS, MS-DOS , OS/2 і різні версії Microsoft Windows 10 і Windows 11 .

Цей дипломний проект присвячений додатковому дослідженню мови Python, парадигм ООП та розробці програмного забезпечення (Гра “Змійка”)

Завдання. На основі дослідженого матеріалу розробити ігровий додаток мовою Python

ЗМІСТ

ВСТУП

Python — це інтерпретована об’єктно-орієнтована мова програмування високого рівня загального призначення. Подібно до PERL, Python є мовою програмування, популярною серед досвідчених програмістів C++ і Java.

Працюючи на Python, користувачі можуть інтерпретувати оператори в кількох операційних системах, включаючи системи на основі UNIX, Mac OS, MS-DOS, OS/2 і різні версії Microsoft Windows 10 і Windows 11.

Python пропонує динамічні типи даних, готові класи та інтерфейси для багатьох системних викликів і бібліотек. Користувачі також можуть розширити його за допомогою іншої мови програмування, наприклад C або C++. Її високорівневі структури даних, динамічне зв’язування та динамічна типізація роблять її однією з основних мов програмування для швидкої розробки програм.

Python також часто використовується як клей або мова сценаріїв, яка плавно з’єднує існуючі компоненти. Користувачі можуть використовувати його для створення сценаріїв у технології Microsoft Active Server Page.

Основні випадки використання Python включають наступне:

1. ML
2. веб-розробка на стороні сервера
3. розробка програмного забезпечення
4. системний сценарій

Кожен, хто користується Facebook, Google, Instagram, Reddit, Spotify або YouTube, стикався з кодом Python.

РОЗДІЛ 1. ОСОБЛИВОСТІ РОЗРОБКИ ІГРОВОГО ЗАСТОСУНКУ

1.1 Особливості розробки ігрових додатків

Концепція гри: Процес розробки починається з переконливої ігрової концепції, яка визначає жанр, механіку, історію та цільову аудиторію гри.

Ігрова механіка: Розробка основної механіки ігрового процесу, включаючи елементи керування, цілі, завдання та системи проходження, має вирішальне значення для захопливого ігрового досвіду.

Історія та персонажі: Створення захопливого сюжету та персонажів, що запам'ятовуються, додає грі глибини та емоційності [1].

Вибір платформи:

Вибір цільової платформи для гри, наприклад, ПК, консолі, мобільних пристроїв або віртуальної реальності (VR), впливає на підхід до розробки, технологічний стек і користувацький досвід.

Ігровий рушій та інструменти:

Вибір відповідного ігрового рушія, такого як Unity, Unreal Engine або Godot, на основі вимог проекту та досвіду команди, забезпечує основу для розробки.

Використання спеціалізованих інструментів, таких як програмне забезпечення для 3D-моделювання (наприклад, Blender, Maya), інструменти графічного дизайну (наприклад, Photoshop, Illustrator) та програмне забезпечення для редагування звуку (наприклад, Audacity), допомагає створювати ресурси для гри.

Візуальний дизайн: Створення візуально привабливої графіки, зокрема 2D- чи 3D-активів, оточення, анімації та спецефектів, покращує естетику гри.

Звуковий дизайн: Включення високоякісних звукових ефектів, фонової музики, закадрового голосу та інших аудіоелементів посилює занурення та залучення гравців.

Інтерфейс користувача та користувацький досвід (UI/UX):

Розробка інтуїтивно зрозумілого та візуально привабливого користувацького інтерфейсу сприяє легкій навігації, доступності та безперебійній взаємодії.

Забезпечення плавного ігрового процесу, чуйності та оптимізації на різних пристроях і розмірах екранів покращує користувацький досвід.

Програмування та розробка:

Реалізація ігрової логіки, механіки та правил за допомогою таких мов програмування, як C++, C#, JavaScript або Python.

Співпраця з системами контролю версій (наприклад, Git) для ефективної командної роботи, управління кодом та відстеження помилок.

Регулярне тестування та налагодження гри для виявлення та виправлення проблем, забезпечення стабільності, продуктивності та сумісності [3].

Багатокористувацький режим та мережева робота:

Реалізація багатокористувацької функціональності вимагає інтеграції мережевих функцій, серверної архітектури, механізмів узгодження та синхронізації.

Забезпечення безпечної та ефективної передачі даних, управління затримками та запобігання шахрайству є критично важливими питаннями.

Монетизація та покупки в додатку (IAP):

Розробка стратегій монетизації, таких як внутрішні покупки, реклама, підписки або преміум-контент, для отримання доходу.

Баланс між монетизацією та користувацьким досвідом, а також уникнення надмірної або нав'язливої монетизації є важливим для задоволення гравців.

Проведення комплексного тестування для виявлення та виправлення помилок, збоїв, проблем з продуктивністю та забезпечення відповідності гри проектним специфікаціям [5].

Ігрове тестування, збір відгуків користувачів та ітерації над дизайном гри для покращення механіки геймплею та загального користувацького досвіду.

Підготовка гри до розповсюдження на різних платформах, таких як магазини додатків (Google Play, Apple App Store), Steam або консольні маркетплейси.

Дотримання специфічних для платформи інструкцій, правил і процесів сертифікації для забезпечення успішного розгортання.

Підтримка та оновлення після запуску:

Надання постійної підтримки, виправлення помилок та оновлення, щоб відповідати на відгуки користувачів, впроваджувати нові функції та підтримувати залученість гравців.

Моніторинг та аналіз метрик гравців, відгуків користувачів та взаємодії зі спільнотою для розробки майбутніх оновлень та покращень.

1.2 Огляд аналогів гри змійка

Гра "Змійка" є класичною та культовою грою, яка користується популярністю серед гравців протягом десятиліть.

Slither.io - це багатокористувацька онлайн-гра, яка поєднує в собі геймплей Snake з елементами концепції Agar.io. Гравці керують змієподібною істотою і прагнуть стати найбільшою на сервері, споживаючи гранули та перемагаючи інших гравців.

Snake.io - це ще одна багатокористувацька гра, де гравці керують зміями та змагаються з іншими на спільній арені. Мета полягає в тому, щоб виростити свою змію, поїдаючи гранули та усуваючи інших змій, уникаючи зіткнень.

Worms Zone - це казуальна аркадна гра з персонажами, схожими на черв'яків. Гравці мандрують барвистим середовищем, збирають їжу, щоб довше рости, і вступають у битви з іншими черв'яками. Гра пропонує різноманітні бонуси та черв'яків, яких можна налаштовувати.

Curve Fever - це багатокористувацька гра, де гравці керують лінією, яка росте під час руху. Мета - вижити і перехитрити суперників, уникаючи зіткнень з іншими лініями, намагаючись заманити їх у пастку своєю лінією.

Snake Pass - це 3D-головоломка-платформер, яка пропонує унікальний погляд на концепцію Snake. Гравці керують змійкою на ім'я Нудл і повинні орієнтуватися

в складному середовищі, вирішувати головоломки та збирати предмети, щоб просуватися в грі.[12]

Nimble Quest поєднує механіку Snake з елементами рольової гри. Гравці керують партією героїв у піксельно-артовому світі, рухаючись у шерензі та перемагаючи ворогів. Мета - вижити, підвищити рівень і відкрити нових персонажів.

Slink.io - це багатокористувацька гра, де гравці керують різнокольоровими змійками у 3D-середовищі. Мета полягає в тому, щоб виростити свою змію, споживаючи кулі, уникаючи зіткнень з іншими зміями. Вона пропонує різні режими гри та бонуси.

Snakebird - це гра-головоломка, яка зберігає основну механіку Snake, але вводить складний дизайн рівнів та виклики. Гравці керують змієподібними птахами і повинні оминати перешкоди, розв'язувати головоломки та збирати фрукти, щоб просуватися вперед.

Snake '97 - це ностальгічна данина оригінальній грі Snake, яку можна було знайти на перших мобільних телефонах. Вона відтворює класичну монохромну піксельну графіку та управління, надаючи гравцям ностальгічний досвід на сучасних пристроях.

Snake Rewind - це сучасна версія класичної гри Snake, створена розробником Nokia Snake. Вона додає нові можливості, такі як бонуси, кілька рівнів та різні режими гри, зберігаючи при цьому основний ігровий процес.

Це лише кілька прикладів численних аналогів та варіацій гри Snake, доступних на різних платформах. Кожна гра пропонує свої унікальні повороти та доповнення до оригінальної ігрової формули, надаючи гравцям різноманітний досвід, але зберігаючи при цьому суть класичної концепції Snake.

1.3 Огляд мов програмування

Коли мова заходить про мови програмування, існує безліч варіантів, кожен з яких має свої сильні та слабкі сторони, а також сфери застосування

Переваги: Python відома своєю простотою, читабельністю та чистим синтаксисом, що робить її легкою у вивченні та написанні коду. Вона має велику стандартну бібліотеку та велику спільноту, що пропонує широку підтримку та ресурси. Python є універсальною і широко використовується в різних сферах, включаючи веб-розробку, аналіз даних, машинне навчання та написання сценаріїв.

Недоліки: Python може бути повільнішою за швидкістю виконання порівняно з мовами нижчого рівня. Вона також не підходить для мобільних або ресурсоємних додатків, де потрібен низькорівневий контроль.

Переваги: JavaScript - це фактична мова для веб-розробки, яка працює у браузерах і дозволяє створювати інтерактивні та динамічні веб-сторінки. Вона має велику екосистему бібліотек і фреймворків, таких як React і Angular, що полегшує ефективну розробку інтерфейсу та бекенду. JavaScript постійно розвивається, а його універсальність поширюється на розробку мобільних додатків (з такими фреймворками, як React Native) та розробку на стороні сервера (з Node.js).

Недоліки: JavaScript може мати проблеми з сумісністю з браузерами, а його динамічна природа може призвести до помилок під час виконання. Як однопотокова мова, вона може мати проблеми з високопродуктивними обчисленнями та обробкою великих обсягів даних.

Переваги: Java - це широко розповсюджена і незалежна від платформи мова, в якій основна увага приділяється надійності та безпеці. Вона має велику стандартну бібліотеку, чудову документацію та велику спільноту розробників. Java використовується для додатків корпоративного рівня, розробки додатків для Android, програмування на стороні сервера тощо. Вона пропонує такі функції, як автоматичне керування пам'яттю (збір сміття) та потужну підтримку багатопотоковості [12].

Недоліки: Java може мати круту криву навчання, а її синтаксис може бути багатослівним порівняно з іншими мовами. Керування пам'яттю, яке здійснює JVM, може призвести до певного зниження продуктивності. Крім того, популярність Java-аплетів та розробки Java-додатків на основі браузера

Переваги: C++ - це потужна та ефективна мова, що забезпечує низькорівневий контроль над системними ресурсами та апаратним забезпеченням. Вона широко використовується в розробці ігор, вбудованих систем, високопродуктивних обчислень та інших критичних до продуктивності додатків. C++ пропонує такі можливості, як сильна перевірка типів, шаблони та прямі маніпуляції з пам'яттю [12].

Недоліки: C++ може бути складною та важчою для вивчення порівняно з мовами вищого рівня. Вона вимагає ретельного керування пам'яттю та ручного виділення/звільнення пам'яті, що може призвести до витоків пам'яті та помилок сегментації, якщо поводитись неправильно. Природа C++, що залежить від часу компіляції, може призвести до довших циклів розробки.

Переваги: C# - це універсальна мова, яка переважно використовується для розробки під Windows, включаючи десктопні додатки, розробку ігор з Unity та бекенд-розробку з .NET Core. Вона пропонує баланс між продуктивністю та простотою використання, має сучасний синтаксис та надійні інструменти розробки. C# має сильний акцент на об'єктно-орієнтованому програмуванні, управлінні пам'яттю через збір сміття та велику стандартну бібліотеку.

Недоліки: C# має меншу підтримку крос-платформної розробки порівняно з іншими мовами, такими як Python або JavaScript. Її екосистема та бібліотеки більше орієнтовані на Windows, хоча такі зусилля, як .NET Core, розширили сферу її застосування. Крім того, C# не настільки поширена у певних сферах, таких як веб-розробка, порівняно з JavaScript або Python.

Це лише кілька прикладів мов програмування, і кожна з них має свої переваги та недоліки залежно від цільового використання. При виборі мови програмування для проекту важливо враховувати такі фактори, як вимоги до продуктивності, час розробки, підтримка спільноти, а також конкретний домен або платформу.

1.4 Особливості мови Python

Для виконання завдання враховуючи переваги та недоліки кожних з мов було обрано використовувати мову Python.

Python — це об’єктно-орієнтована, динамічна, високорівнева, інтерпретована мова програмування з відкритим кодом. Це дуже легко освоїти і поширене як серед початківців, так і серед професіоналів. Python має кілька функцій і сильну підтримку спільноти, завдяки чому він має величезну колекцію вбудованих модулів і бібліотек для різноманітних програм [12].

Python — це високорівнева мова програмування загального призначення, розроблена Гвідо Ван Россумом у лютому 1991 року. Python розроблено таким чином, що в ньому більше уваги приділяється читабельності коду завдяки використанню значних відступів. Це динамічна та одна з найуніверсальніших мов програмування, які ми маємо. Python дуже простий у вивченні і використовується для початку програмування та знаходить своє застосування в широкій сфері. Використовуючи Python, ми можемо не лише розробляти комп’ютерне програмне забезпечення, але й веб-додатки та програми для Android, вбудовані системні програми та багато іншого.

Python має кілька функцій і сильну підтримку спільноти, завдяки чому він має величезну колекцію вбудованих модулів і бібліотек для різноманітних програм. Не обмежуючись цим, Python також підтримує об’єктно-орієнтоване програмування, а також підхід до процедурно-орієнтованого програмування. Python 3.10 — це остання версія, яка була випущена з багатьма функціями та оптимізаціями. В останні роки Python незмінно вважався однією з найпопулярніших мов програмування.

Таким чином, існує багато функцій python, але ті, які роблять його таким популярним і орієнтованим на застосування в різних областях, обговорюються нижче [12].

Python є однією з найбільш зручних мов програмування. Можна легко вивчити основи Python, ознайомитися з його синтаксисом і мати можливість писати

базові програми за кілька днів. Однак вивчення складних концепцій і освоєння Python може зайняти у вас деякий час. Порівняно з іншими мовами, такими як C, C++, Java тощо, Python є найпростішою мовою для вивчення та освоєння.

Синтаксис Python дуже простий. Як правило, це слова з англійської мови. Під час написання коду зазвичай відчувається, що давати інструкції дитині. Окрім цього, на відміну від інших мов програмування (як-от C, C++, Java тощо, немає необхідності відкривати чи закривати дужки для визначення області. У Python ми використовуємо відступи (пробіли або табуляції) для обсяг, який робить код чистим і вражаючим.

Код Python не компілюється відразу, перетворюється на файл .exe, а потім виконується. Python є інтерпретованою мовою, що означає, що її код виконується рядок за рядком, а не весь одразу, як в інших мовах програмування. Це построкове виконання також полегшує налагодження коду.

Python — це безкоштовна мова програмування з відкритим вихідним кодом, що означає, що її можна використовувати безкоштовно в будь-якій операційній системі та без проблем з авторським правом. Кожен може завантажити Python з його офіційного веб-сайту разом із його бібліотеками та документацією. Ви можете завантажити, але це також дозволяє створювати власні модулі чи бібліотеки та поширювати їх.

Об’єктно-орієнтоване програмування — це парадигма програмування, яка базується на поняттях класів та об’єктів. Класи служать схемою для об’єктів, яка містить дані та методи, які діють на ці дані. Концепція об’єктно-орієнтованого програмування зосереджена на створенні багаторазового коду з хорошим рівнем абстракції.

Одним із найважливіших аспектів Python є об’єктно-орієнтоване програмування. Python підтримує конструкції об’єктно-орієнтованого програмування, такі як класи, інкапсуляція даних, успадкування, поліморфізм тощо. У Python легко створювати та використовувати класи, об’єкти та можемо реалізувати ООП-конструкції. Завдяки такому підходу можна створювати ефективні та потужні програми на Python [12].

Python є кросплатформною мовою. Багато разів, завантажуючи програмне забезпечення з якогось веб-сайту, ви могли помічати список версій цього програмного забезпечення, сумісного з різними операційними системами. У Python це не так, як тільки ви напишете код python на одній машині чи операційній системі, його можна буде запустити будь-де. Наприклад, якщо ми створили програму Python на Mac, ми можемо запустити той самий код у Linux, Windows чи будь-якій іншій операційній системі без жодних змін. Це пояснюється тим, що код Python спочатку перетворюється на проміжну форму, відому як байт-код, а потім виконується.

Python має можливості, які можна розширити та стати більш універсальною мовою програмування. Python виявилася універсальною мовою, оскільки вона охоплює велику сферу програм для розробки програмного забезпечення завдяки своїй адаптованості до різних функціональних можливостей. Ми можемо скомпілювати код такими мовами, як C/C++, а потім використовувати його в нашому коді Python, який можна скомпілювати та запустити де завгодно. Це дозволяє виконувати код, написаний на інших мовах програмування. Це надає Python нові можливості та функціональність завдяки інтеграції коду інших мов програмування.

Python — це мова програмування високого рівня, що означає, що користувачі можуть легко писати та розуміти чи інтерпретувати код. Мова програмування високого рівня дозволяє програмісту писати коди, які менш залежать від конкретного типу машини. Python — це мова програмування з дуже сильною абстракцією від низькорівневих конструкцій системи чи машини. Під час написання коду розробнику не потрібно турбуватися про архітектуру, керування пам’яттю або базовий тип машини [12].

Сьогодні майже кожна програма, яку нам потрібно розробити, безумовно, потребує бази даних, і тут з’являється API бази даних Python (DB-API), який забезпечує інтерфейс майже до більшості основних комерційних баз даних. Деякі з баз даних, які підтримуються стандартним Python, це MySQL, PostgreSQL, Microsoft SQL, Oracle, Informix тощо. Для використання потрібно імпортувати

інтерфейс для конкретної бази даних. Використовуючи Python, ви можете працювати як з реляційними, так і з нереляційними базами даних.

Коли ми використовуємо наші комп'ютери чи смартфони, як ми взаємодіємо з ними, або що ми бачимо? Очевидно, що на екрані є різні значки програм, і коли ми відкриваємо деякі програми, ми бачимо гарне візуальне представлення, яке полегшує нам взаємодію та використання цієї програми. Це те, що ми називаємо GUI або графічним інтерфейсом користувача. Це один із ключових аспектів мови програмування Python. У Python є багато бібліотек графічного інтерфейсу користувача, таких як Tkinter, PyQT5, PSide2 тощо. Графічний інтерфейс користувача надає користувачеві можливість легше взаємодіяти з програмою та системою, а бібліотеки графічного інтерфейсу користувача python мають функції, які роблять розробку програмного забезпечення з інтенсивним використанням графіки легкою та швидкою.

Python складається з великої кількості бібліотек, які є крос-платформними та надають багатий набір модулів і функцій. Ці бібліотеки сумісні з різними операційними системами, такими як UNIX, Mac, Windows тощо. Завдяки великій кількості бібліотек нам не потрібно писати код для кожної окремої речі, а імпортувати та використовувати необхідні функції. Наприклад, якщо вам потрібно отримати доступ до деяких веб-сайтів і ви хочете отримати з них дані, вам не потрібно писати функції для запиту, відповіді та інших речей з нуля. Для цього доступні різні бібліотеки, якими ви можете скористатися.

Ми також можемо інсталювати інші пакунки, які не є частиною стандартної бібліотеки, якщо нам потрібні додаткові функції.

Python є динамічно типізованою мовою. Динамічно типізований означає, що, на відміну від інших мов програмування, у Python нам не потрібно явно оголошувати тип даних (наприклад, int, float, double, char тощо). Тип даних змінної визначається під час виконання. Окрім цього, одну змінну можна використовувати для зберігання різних типів даних у різних екземплярах програми. Ця функція python економить багато часу та допомагає нам уникнути пасток, які могли б виникнути, якби вимагалося явно зазначити тип даних [13].

Крім розглянутих вище функцій, Python надає кілька вдосконалених механізмів програмування, таких як розуміння, генератори, декоратори тощо, що робить його більш особливим. Крім того, керування пам’яттю забезпечується внутрішнім шляхом використання приватної купи в Python. Через таке керування внутрішньою пам’яттю в Python немає поняття покажчиків.

Python - це популярна та універсальна мова програмування, яка добре підходить для розробки таких ігор, як Snake.

Чистий та інтуїтивно зрозумілий синтаксис Python полегшує написання та розуміння коду. Ця простота корисна як для початківців, так і для досвідчених розробників, дозволяючи швидше розробляти та легше підтримувати кодову базу гри.

Python має багату екосистему бібліотек та фреймворків, які можуть пришвидшити розробку ігор. Для розробки ігор Snake такі бібліотеки, як Pygame, надають необхідні функції для обробки графіки, користувацького вводу, виявлення зіткнень та звуку.

Python є кросплатформенною мовою, що означає, що та сама кодова база може працювати на різних операційних системах без значних модифікацій. Це полегшує розробку ігор Snake, які можна розгортати і використовувати на різних платформах, включаючи Windows, macOS і Linux [15].

Високорівнева природа Python та великі бібліотеки дозволяють швидко створювати прототипи ігрових функцій. Динамічна типізація та інтерпретована природа мови дозволяють проводити швидкі ітерації та експерименти, пришвидшуючи процес розробки.

Python може похвалитися великою та активною спільнотою розробників, які сприяють її розвитку та підтримці. Це означає, що розробники, які працюють над грою на Snake, можуть знайти безліч ресурсів, навчальних посібників та форумів, де вони можуть отримати поради, вирішити проблеми та поділитися своїми напрацюваннями зі спільнотою.

Python може легко взаємодіяти з іншими мовами, такими як C/C++, за допомогою вбудованих механізмів розширення. Ця функція дозволяє розробникам

оптимізувати критичні до продуктивності компоненти гри, реалізуючи їх мовами нижчого рівня, водночас користуючись перевагами високорівневих абстракцій Python для інших частин гри.

Підтримка ООП у Python дозволяє розробникам організовувати код у модульні та багаторазові компоненти. У грі Snake це може бути корисним для керування поведінкою змійки, обробки ігрових об'єктів та реалізації ігрових станів.

Python має функцію автоматичного керування пам'яттю через збір сміття, звільняючи розробників від необхідності вручну керувати розподілом та вивільненням пам'яті. Це спрощує процес розробки, зменшує ризик витоку пам'яті та підвищує стабільність гри Snake.

Широкі бібліотеки Python для обробки даних та штучного інтелекту (ШІ) можуть бути використані для покращення гри Snake. Наприклад, бібліотеки машинного навчання, такі як TensorFlow або PyTorch, можна використовувати для розробки ШІ-агентів, які розумно грають у гру або генерують адаптивні рівні.

Python пропонує ряд інструментів для тестування та налагодження, які полегшують розробку високоякісних ігор. Фреймворки на кшталт pytest та інструменти на кшталт pdb дозволяють розробникам писати та виконувати тести, виявляти та виправляти проблеми, а також забезпечувати надійність гри Snake.

Python надає інструменти та фреймворки, такі як PyInstaller та cx_Freeze, які дозволяють пакувати та розповсюджувати ігри на Python як окремі виконувані файли. Це дозволяє зручно ділитися грою Snake з іншими користувачами, не вимагаючи від них встановлення Python або додаткових залежностей.

Хоча Python не є оптимальним вибором для ігрових рушіїв, що вимагають високої продуктивності, або ігор з високою графікою, він чудово забезпечує баланс між швидкістю розробки, читабельністю та простотою використання. Її великі бібліотеки, крос-платформна сумісність та активна спільнота роблять її чудовим вибором для швидкої та ефективної розробки гри Snake.

1.5 Принципи ООП в Python

Об'єктно-орієнтоване програмування (ООП) - це парадигма програмування, яка фокусується на організації коду навколо об'єктів, які є екземплярами класів.

Python, як об'єктно-орієнтована мова, підтримує принципи ООП. Інкапсуляція відноситься до об'єднання даних і методів в єдину одиницю, відому як клас. У Python класи інкапсулюють атрибути (дані) та методи (функції), які оперують цими даними. Принцип інкапсуляції забезпечує цілісність, абстрактність та модульність даних [15].

Абстрагування передбачає представлення складних об'єктів реального світу у вигляді спрощених моделей у коді. Абстракція у Python досягається за допомогою класів та інтерфейсів, які приховують непотрібні деталі та виставляють на загальний огляд лише найважливіші особливості. Це дозволяє розробникам зосередитися на основних характеристиках об'єкта та його поведінці, не заглиблюючись у деталі реалізації.

Спадкування - це механізм, за допомогою якого клас може успадковувати властивості (атрибути та методи) від іншого класу. У Python клас може успадковувати від одного або декількох батьківських класів, утворюючи ієрархію класів. Спадкування полегшує повторне використання коду, сприяє модульності та дозволяє створювати спеціалізовані класи (похідні класи) на основі існуючих (базових або батьківських класів).

Поліморфізм означає здатність об'єктів набувати різних форм або демонструвати різну поведінку в залежності від контексту. Поліморфізм у Python досягається за допомогою перевизначення та перевантаження методів. Перевизначення методу дозволяє підкласу надавати іншу реалізацію методу, успадкованого від суперкласу. Перевантаження методів дозволяє класу визначати декілька методів з однаковими іменами, але різними параметрами.

Модульність - це принцип проектування, який заохочує поділ коду на незалежні модулі, які можна використовувати повторно. Python підтримує модульність шляхом створення класів, які діють як самодостатні одиниці. Кожен

клас інкапсулює певну функціональність, що полегшує обслуговування, тестування та організацію коду.

Асоціація являє собою зв'язок між класами, де об'єкти одного класу пов'язані з об'єктами іншого класу. У Python асоціація встановлюється шляхом створення посилань або атрибутів між класами. Це дозволяє об'єктам співпрацювати, взаємодіяти та обмінюватися інформацією один з одним.

Композиція - це форма об'єднання, коли один клас містить об'єкти інших класів як частину своєї структури. У Python композиція досягається шляхом створення атрибутів на рівні класу, які посилаються на інші об'єкти. Композиція дозволяє створювати складні об'єкти, комбінуючи простіші, і сприяє повторному використанню коду та гнучкості [15].

Залежність - це зв'язок, при якому зміна в одному класі може вплинути на інший клас. У Python залежність встановлюється, коли один клас залежить від іншого класу через параметри методів або значення, що повертаються. Залежності між класами повинні бути зведені до мінімуму, а слабкі зв'язки заохочуються для кращої супроводжуваності та гнучкості.

Дотримуючись цих принципів, розробники можуть писати більш організований, підтримуваний і масштабований код на Python. ООП у Python сприяє повторному використанню коду, модульному дизайну та абстрагуванню, що призводить до ефективної розробки та полегшує обслуговування програмних систем.

Існує багато інших методологій, які використовуються для розробки програмного забезпечення, але найпопулярнішим є об’єктно-орієнтоване програмування.

Принципи об’єктно-орієнтованого програмування є найпопулярнішими серед інших, оскільки вони стосуються реальних об’єктів. Кожна операція, яка буде функціональною, розглядається з точки зору класів та об’єктів. Це забезпечує кращий стиль програмування, оскільки вам не потрібно писати код, який потрібно запускати у будь-який час. Замість цього ви можете створити класи, що визначають функціональність, і викликати цю функцію, створивши її об’єкт [15].

Окрім цих, він також забезпечує такі принципи, як спадкування, які підвищують можливість повторного використання коду та допомагають дуже легко застосовувати оновлення. А абстракція, інкапсуляція допомагає підвищити безпеку даних.

Мови, які підтримують об’єктно-орієнтоване програмування – C++, Java, JavaScript, C#, PHP тощо.

А мови, які є повністю об’єктно-орієнтованим програмуванням або чистим об’єктно-орієнтованим – це – python, Ruby, Scala тощо.

Об’єктно-орієнтовані принципи в основному включають 4 стовпи, які разом роблять ООП дуже потужною концепцією.

1. Абстракція
2. Інкапсуляція
3. Спадщина
4. Поліморфізм

ООП базується на реальних інженерних речах. Подібно до підходів, які застосовуються в іншій інженерії – інженери-електронники створюють певний пристрій, інженери-автомобілі створюють якийсь транспортний засіб тощо. Таким чином, підходи до проектування слідували їхнім, те ж саме відтворено в розробці програмного забезпечення з концепцією цих 4 принципів об’єктно-орієнтованого програмування.

Якщо ми проаналізуємо концепцію програмування, то виявимо, що є два елементи програмування – дані та операції над даними [функції].

Отже, якщо ми розробляємо програмне забезпечення, воно призначене лише для виконання завдань із даними. І якщо будь-яка функція виконується над даними, то можуть існувати шанси, що в майбутньому ту саму операцію можна буде виконати або тим самим, або якимось модифікованим способом. Тож це дуже легко робиться за допомогою концепції принципів ООП. Розглянемо принципи:

Абстракція

Абстракцію можна визначити як приховування внутрішньої реалізації та показ лише необхідних функцій або набору послуг, які пропонуються. Це найважливіша частина об'єктно-орієнтованого програмування.

Специфікатори доступу — це ключові слова, які вказують, звідки можна отримати доступ до (атрибутів і методів) класу. Наприклад, приватне може бути доступне лише в межах класу, загальнодоступне може бути доступне з будь-якого місця тощо.

Подібним чином інші мови програмування дотримуються власної реалізації для досягнення абстракції.

```
клас телебачення{
   void changeChannel(){
      //Реалізація
   }
   void adjustVolume(){
      //Реалізація
   }
   void otherFeatues(){
      //Реалізація
   }
}
```

У цьому прикладі псевдокоду стверджується, що клас телебачення надає такі функції, як перемикання каналів, регулювання гучності тощо, тож нам як користувачу потрібно використовувати лише ці функції. Нам не потрібно знати внутрішню реалізацію, наприклад, як працює функція для зміни каналів, як коди керують регулюванням гучності тощо. Нам надано метод для виконання цього завдання, тому просто використовуйте це. Якщо ми розглянемо приклад більше

специфічні для програмування, то ми можемо сказати, що функції, включені в бібліотеку Math, такі як – pow(), min(), max() тощо, існують, і ми використовуємо їх лише для отримання результату. Це не той випадок, коли ми перевіряємо його виконання. Отже, це теж абстракція.

Особливості абстракції.

Безпека. За допомогою абстракції сторонні особи не знають внутрішньої реалізації, тому класи стають більш безпечними, тому неавторизований користувач не матиме доступу до даних.

Легке вдосконалення. Припустімо, що в майбутньому ми захочемо змінити логіку реалізації в класі, тому нам не доведеться змінювати всю зовнішню логіку, яка є для користувача. Просто нам потрібно внести зміни в методи, і це не вплине на функціональність.

Покращує простоту – це допомагає використовувати функції без знання реалізації, тому це покращує легкість для користувачів у використанні.

Ремонтопридатність покращиться – не впливаючи на користувача, він може вносити будь-які типи внутрішніх змін. Так ремонтопридатність покращиться.

Інкапсуляцію можна визначити як зв’язування даних і атрибутів або методів і членів даних в єдиний блок. У класах ми маємо дані та атрибути, які виконують операції з цими даними. Відповідно принципу інкапсуляції ООП ці дані можна об’єднати в єдиний блок. Інкапсуляція підвищує безпеку даних, оскільки все, що стосується одного завдання, має бути згруповано, а доступ до даних надається відповідно до потреб.

І цього можна досягти за допомогою концепції приховування даних .

Інкапсуляція = приховування даних + абстракція.

Приховування даних – означає приховування даних класу та обмеження доступу до зовнішнього світу. Приклад – використання ключових слів специфікатора доступу, як-от private, що обмежує дані лише доступними та модифікованими в одному класі. Зовнішні користувачі не можуть отримати доступ до даних.

Особливості інкапсуляції.

1. Він забезпечує додатковий захист завдяки концепції приховування даних.

2. Він групує дані та операції з ними в єдиний блок.

3. Він надає додатковий рівень безпеки до даних і дозволяє отримати доступ до даних лише авторизованим особам.

Спадщина.

Спадкування — це спосіб придбання ознак існуючого класу в новий клас. Припустимо, що існує клас, який вважається батьківським класом і має деякі методи, пов’язані з ним. І ми оголошуємо новий клас, який розглядається як дочірній клас, який має власні методи. Отже, коли дочірній клас успадковується від батьківського класу, тоді всі методи, пов’язані з батьківським класом, будуть доступні в дочірньому класі разом із власними методами дочірнього класу.

У спадкуванні беруть участь два ключові слова – базовий і похідний клас.

Базовий клас – його також називають батьківським класом. Визначено основний клас, який має основні властивості та методи.

Похідний клас – це розширення базового класу, який також називають дочірнім класом. Він має властивості та методи, які є в базовому класі з власними функціями в ньому.

Спадкування має кілька типів, які залежать від реалізації мов програмування. Деякі з поширених типів успадкування:

Одинарне успадкування – коли існує лише один похідний клас.

Множинне успадкування – коли дочірній клас успадковує більше ніж один базовий клас.

Багаторівневе успадкування – коли є рівень успадкування. З класу A в клас B в клас C.

Ієрархічне успадкування – коли більше ніж один похідний клас успадковує один базовий клас.

Поліморфізм є найважливішою концепцією принципу об'єктно-орієнтованого програмування. Воно має значення «poly» – багато, «morph» – форми. Отже, поліморфізм означає багато форм.

В об'єктно-орієнтованому програмуванні будь-який об'єкт або метод має більше ніж одне ім'я, пов'язане з ним. Це не що інше, як поліморфізм.

Для об’єктів – коли будь-який об’єкт може містити посилання на свій батьківський об’єкт, це поліморфізм.

OOPs визначає 2 типи поліморфізму: поліморфізм часу компіляції та поліморфізм виконання.

Поліморфізм часу компіляції. Також називається статичним зв’язуванням.

Поліморфізм виконання. Також називається динамічним зв’язуванням.

Поліморфізм часу компіляції досягається за допомогою перевантаження методів.

Перевантаження методу – коли оголошено більше одного методу з однаковою назвою, але з різною кількістю параметрів або різним типом даних параметра, це називається перевантаженням методу.

Принцип об’єктно-орієнтованого програмування — це найпоширеніша парадигма програмування, яка допомагає розробляти масштабні програми в реальному світі за допомогою модулів, які дозволяють багатьом розробникам працювати разом, щоб створити повну систему. Принципи об’єктно-орієнтованого програмування також покращують функції безпеки за допомогою таких потужних концепцій, як абстракція та інкапсуляція.

Більшість популярних мов програмування повністю засновані на принципах ООП, як Python, Ruby тощо.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ЗАСТОСУНКУ

2.1 Огляд засобів для написання коду

Інтегровані середовища розробки (IDE) для написання коду на Python та розробки гри Snake, Visual Studio Code (VS Code) та подібні інструменти є популярним вибором.

Visual Studio Code - це легке та універсальне середовище розробки, яке підтримує розробку на Python. Вона пропонує широкий спектр функцій, включаючи навігацію кодом, підсвічування синтаксису, налагодження та інтеграцію з Git. Переваги VS Code для розробки ігор на Python включають в себе широку екосистему розширень, таких як розширення Python, яке забезпечує лінтування, IntelliSense та багатий досвід налагодження. VS Code також підтримує інтеграцію з системами контролю версій і пропонує інтерфейс, що налаштовується. Однак налаштування середовища розробки та конфігурування певних функцій може потребувати додаткових кроків.

PyCharm - це спеціалізоване середовище розробки на Python, розроблене компанією JetBrains. Вона пропонує повний набір функцій для розробки на Python, включаючи завершення коду, аналіз коду, інструменти для налагодження та підтримку модульного тестування. Серед переваг PyCharm - інтелектуальні підказки коду, потужні можливості рефакторингу та безшовна інтеграція з популярними фреймворками, такими як Django та Flask. Це забезпечує безперебійну розробку, особливо для великих проектів. Однак, широкі можливості PyCharm та його інтерфейс можуть бути непосильними для початківців або проектів з більш простими вимогами. Професійна версія PyCharm може вимагати платної ліцензії для певних розширених функцій.

Spyder - це IDE, спеціально розроблена для наукових обчислень та аналізу даних за допомогою Python. Вона включає такі функції, як інтерактивна консоль, дослідник змінних та інструменти налагодження. Серед переваг Spyder для розробки ігор - інтеграція наукових бібліотек, таких як NumPy та SciPy, які можуть

бути корисними для математичних операцій в ігровій механіці. Зручний інтерфейс Spyder та його зосередженість на аналізі даних роблять його придатним для розробників, які потребують балансу між розробкою ігор та обробкою даних. Однак для початківців Spyder може мати крутішу криву навчання, а його інтерфейс може бути не таким гнучким у налаштуванні, як інші IDE.

Jupyter Notebooks надає інтерактивне середовище для написання та виконання коду на Python. Вони поєднують код, документацію та візуалізації в єдиному інтерфейсі, схожому на документ. Переваги Jupyter Notebooks для розробки ігор включають можливість створювати та ділитися інтерактивними прототипами ігор, візуалізувати ігрові дані та надавати детальні пояснення. Jupyter Notebooks особливо корисні для дослідницького кодування, експериментів та освітніх цілей. Однак, блокноти можуть не підходити для великих проектів або довготривалої підтримки коду, оскільки їм може бракувати деяких функцій традиційних IDE, наприклад, розширених можливостей налагодження.

Ці IDE надають різні функції та робочі процеси для покращення розробки ігор на Python. Вибір залежить від особистих уподобань, вимог проекту та знайомства з інструментами. Варто враховувати такі фактори, як розмір проекту, потреби у співпраці, вимоги до продуктивності, доступні розширення та загальний досвід розробки, який пропонує кожна IDE.

2.2 Характеристика visual studio

Зважуючи переваги та недоліки було обрано використовувати IDE visual studio 2022.

Visual Studio 2022 - це потужне інтегроване середовище розробки (IDE), розроблене компанією Microsoft. Воно надає повний набір інструментів і функцій для підтримки розробки програмного забезпечення на різних платформах і мовах програмування. Нижче я надам детальний опис Visual Studio 2022 в рамках заданого ліміту в 5000 символів.

Visual Studio 2022 дозволяє розробникам створювати додатки для різних платформ, включаючи Windows, macOS, iOS, Android та Інтернет. Вона надає шаблони проектів, інструменти налагодження та емулятори/симулятори для спрощення крос-платформної розробки, що дозволяє розробникам охопити ширшу аудиторію зі своїми додатками.

Visual Studio 2022 підтримує широкий спектр мов програмування, включаючи C#, C++, JavaScript, TypeScript, Python та інші. Вона пропонує широкі можливості редагування коду, IntelliSense, рефакторинг коду та інструменти налагодження для конкретних мов, що робить її придатною для різноманітних сценаріїв розробки та дозволяє розробникам ефективно працювати з мовами, яким вони надають перевагу.

Visual Studio 2022 надає потужні засоби налагодження, які допомагають виявляти та виправляти проблеми в додатках. Підтримуються різні методи налагодження, такі як покрокове виконання коду, точки зупинки, вікна спостереження та профілювання продуктивності. IDE пропонує комплексний досвід налагодження, гарантуючи, що розробники можуть легко діагностувати та усувати помилки у своєму коді.

Visual Studio 2022 легко інтегрується з популярними системами контролю версій, такими як Git, що дозволяє розробникам ефективно керувати вихідним кодом. Вона надає такі функції, як візуалізація гілок, вирішення конфліктів та історія комітів, що полегшує співпрацю з членами команди та відстеження змін у сховищах коду.

Visual Studio 2022 пропонує ряд функцій для підвищення продуктивності редагування коду. Це інтелектуальне завершення коду (IntelliSense), навігація по коду, інструменти рефакторингу та настроюване форматування коду. IDE також підтримує аналіз коду в реальному часі та надає пропозиції щодо покращення якості коду та дотримання стандартів кодування.

Visual Studio 2022 надає надійну систему управління проектами, яка допомагає організовувати та керувати великими програмними проектами. Вона пропонує шаблони проектів, файли рішень, управління залежностями проекту та

параметри конфігурації проекту. IDE дозволяє розробникам легко налаштовувати параметри проекту, керувати посиланнями та конфігураціями збірок.

Visual Studio 2022 включає в себе набір інструментів для тестування, які полегшують розробку високоякісного програмного забезпечення. Він підтримує фреймворки для модульного тестування, такі як MSTest, NUnit і xUnit, а також надає такі функції, як дослідник тестів, аналіз покриття коду і налагодження тестів. Ці інструменти дозволяють розробникам писати, запускати та аналізувати тести безпосередньо в IDE.

Visual Studio 2022 легко інтегрується з різними хмарними сервісами, такими як Azure, що дозволяє розробникам створювати хмарні додатки та сервіси. Вона пропонує інструменти для розгортання, управління та моніторингу хмарних ресурсів і підтримує такі технології, як Azure Functions, Azure App Service і Azure DevOps.

Visual Studio 2022 надає функції для полегшення співпраці та командної розробки. Вона підтримує спільний доступ в реальному часі, що дозволяє розробникам співпрацювати в режимі реального часу над однією і тією ж кодовою базою. Вона також інтегрується з Azure DevOps для керування робочими елементами, контролем вихідного коду та конвеєрами безперервної інтеграції/безперервного розгортання (CI/CD).

Visual Studio 2022 включає функції з підтримкою штучного інтелекту, такі як IntelliCode з підтримкою ШІ, який надає контекстно-орієнтовані рекомендації щодо коду на основі шаблонів і практик, отриманих з мільйонів репозиторіїв коду. Ці можливості ШІ допомагають розробникам писати код ефективніше та зменшити кількість типових помилок [17].

2.3 Реалізація програмного застосунку

Функціональні вимоги:

Рух змійки: Змійка повинна мати можливість рухатися в різних напрямках (вгору, вниз, вліво, вправо) на основі вводу користувача або заздалегідь визначених елементів керування.

Генерація їжі: Гра повинна випадковим чином генерувати їжу на ігровому полі для змійки, яку вона може з'їсти.

Зростання змії: Коли змія їсть їжу, вона повинна рости в довжину, додаючи новий сегмент до свого тіла.

Виявлення зіткнень: Гра повинна виявляти зіткнення змійки з межами ігрового поля або з власним тілом, що призводить до завершення гри.

Система підрахунку очок: Гра повинна відстежувати очки гравця на основі кількості з'їдених змійкою продуктів харчування.

Високі бали: Гра повинна вести облік найвищих результатів, досягнутих різними гравцями.

Прогресія рівнів: Гра повинна включати кілька рівнів зростаючої складності, де кожен рівень вводить нові виклики або перешкоди.

Посилення: Гра може містити бонуси, які надають гравцеві тимчасові переваги, наприклад, підвищену швидкість або непереможність.

Кінець гри: Коли змія зіштовхується з межею або власним тілом, гра повинна відображати екран завершення гри з фінальним рахунком гравця.

Не функціональні вимоги:

Продуктивність: Гра повинна бути чуйною та мати плавні та плавні рухи, забезпечуючи безперебійний ігровий процес.

Інтерфейс користувача: Гра повинна мати інтуїтивно зрозумілий та зручний інтерфейс, з чіткими інструкціями та легко ідентифікованими ігровими елементами.

Графіка та звук: Гра повинна мати візуально привабливу графіку та додаткові звукові ефекти, що покращують загальний ігровий досвід.

Сумісність: Гра має бути сумісною з різними платформами та пристроями, такими як настільні комп'ютери, мобільні телефони та планшети.

Оперативність: Гра повинна швидко реагувати на дії користувача, забезпечуючи швидкий рух змійки в потрібному напрямку.

Надійність: Гра повинна працювати стабільно, без збоїв або неочікуваних помилок під час ігрового процесу.

Доступність: Гра має бути доступною для широкого кола користувачів, у тому числі для людей з обмеженими можливостями, завдяки таким функціям, як регулювання розміру шрифту чи кольорової гами.

Локалізація: Гра повинна підтримувати кілька мов, щоб гравці з різних регіонів могли розуміти гру і насолоджуватися нею.

Безпека: Гра повинна безпечно обробляти дані користувачів, захищаючи особисту інформацію та запобігаючи несанкціонованому доступу.

Масштабованість: Гра повинна бути масштабованою, щоб пристосуватися до потенційних оновлень, додавання нових функцій або збільшення трафіку користувачів.

Ці вимоги надають вичерпну схему для розробки ігрового додатку Snake, забезпечуючи як бажану функціональність, так і якість користувацького досвіду.

Після дослідження переходимо до побудови програмного застосунку у середовищі visual studio який наведено на Рис 2.1 або іншому IDE як приклад PyCharm.

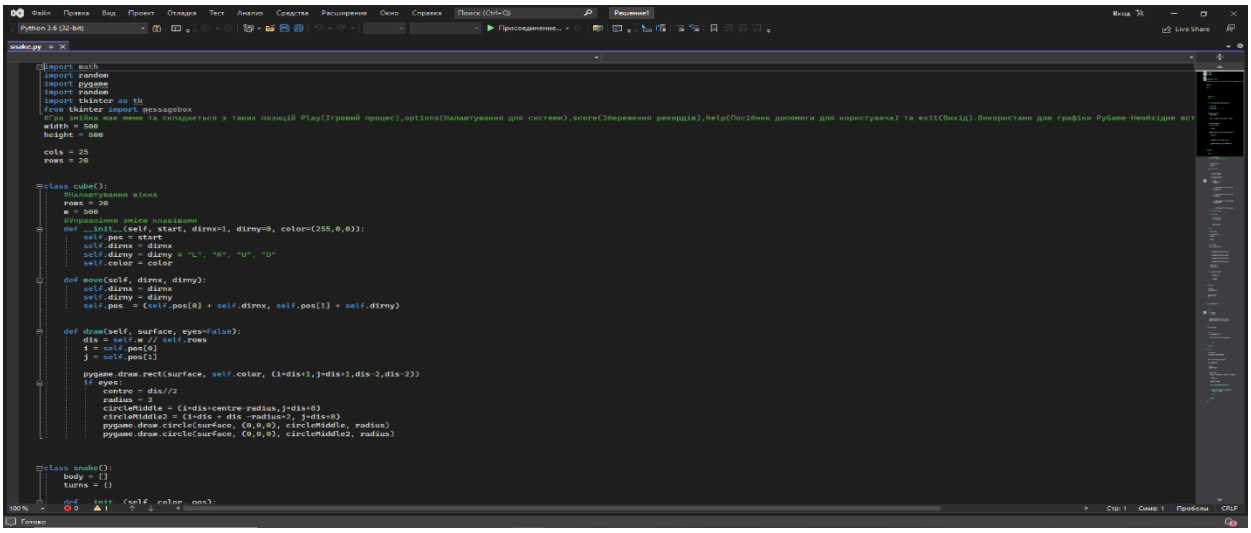


Рисунок 2.1.Написаний код в IDE visual studio 2022

Також беремо для опису один з класів застосунку:

```
class cube():
    #Налаштування вікна
    rows = 20
    w = 500
    #Управління змією клавішами
    def __init__(self, start, dirnx=1, dirny=0, color=(255,0,0)):
        self.pos = start
        self.dirnx = dirnx
        self.dirny = dirny # "L", "R", "U", "D"
        self.color = color

    def move(self, dirnx, dirny):
        self.dirnx = dirnx
        self.dirny = dirny
        self.pos  = (self.pos[0] + self.dirnx, self.pos[1] + self.dirny)

    def draw(self, surface, eyes=False):
```

```
        dis = self.w // self.rows
        i = self.pos[0]
        j = self.pos[1]

        pygame.draw.rect(surface, self.color, (i*dis+1,j*dis+1,dis-2,dis-2))
        if eyes:
            centre = dis//2
            radius = 3
            circleMiddle = (i*dis+centre-radius,j*dis+8)
            circleMiddle2 = (i*dis + dis -radius*2, j*dis+8)
            pygame.draw.circle(surface, (0,0,0), circleMiddle, radius)
            pygame.draw.circle(surface, (0,0,0), circleMiddle2, radius)
```

За допомогою цього класу в грі відбуваються такі дії:

1. Налаштування вікна
2. Управління ігровим персонажем

2.4 Діаграми UML

UML, що розшифровується як Unified Modeling Language, — це спосіб візуального представлення архітектури, дизайну та реалізації складних програмних систем. Коли ви пишете код, у програмі є тисячі рядків, і важко відстежувати зв’язки та ієрархії в системі програмного забезпечення. Діаграми UML поділяють цю програмну систему на компоненти та підкомпоненти.

UML — це стандартизована мова моделювання, яку можна використовувати в різних мовах програмування та процесах розробки, тому більшість розробників програмного забезпечення зрозуміють її та зможуть застосувати у своїй роботі.

Хоча багато інженерів бояться діаграм, вони корисні в гнучкому середовищі розробки: вони забезпечують продуктивність і цілеспрямованість розробки.

Замість того, щоб думати про них як про «приємне мати», ставтеся до своїх діаграм UML як до основних аспектів документації. Діаграми UML можуть допомогти командам інженерів:

1. Швидко залучайте нових членів команди або розробників, які змінюють команди.
2. Навігація вихідним кодом.
3. Плануйте нові функції до того, як розпочати програмування.
4. Спілкуйтеся з технічною та нетехнічною аудиторією легше.

Однак діаграми, які не розвиваються разом з проектом, марні, тому необхідно мати діаграми, що постійно розвиваються. Lucidchart, хмарне рішення для діаграм, полегшує цей процес. Lucidchart може генерувати діаграми послідовності UML з текстової розмітки, що робить діаграми автоматичними та еластичними.

Зображення блоків у алгоритмі, їх розміри, товщина ліній, кут нахилу ліній тощо, регламентуються Державним стандартом "Схеми алгоритмів, програм, даних і систем", а саме: 19.701-90 (ISO 5807-85).

Блоки у блок-схемі з'єднуються лініями потоків. У кожен блок може входити не менше однієї лінії, з блоку ж (окрім логічного) може виходити лише одна лінія потоку. З логічного блоку завжди виходять дві лінії потоку: одна у випадку виконання умови, інша - при її невиконанні. Бажано, щоб лінії потоку не перетинались.

Алгоритм може бути детальним, або спрощеним (деякі зрозумілі блоки можуть не записуватись, інакше алгоритм збільшується в розмірі).

Блок-схема - це спосіб представлення алгоритму в графічній формі, у вигляді геометричних фігур, сполучених між собою лініями (стрілками). Форма блока визначає тип дії, а текст всередині блоку дає детальне пояснення конкретної дії. Стрілки на лініях, що сполучають блоки схеми, вказують послідовність виконання команд, передбачених алгоритмом. Блок-схеми, за рахунок наочності спрощують створення ефективних алгоритмів, розуміння роботи вже створених, а як наслідок і їх оптимізацію. Існуючі стандарти на типи блоків дозволяють легко адаптувати

алгоритми, створені у вигляді блок-схем до будь-яких існуючих на сьогоднішній день мов програмування.

На Рис 2.2 наведена блок-схема алгоритму, яка зображає послідовність блоків, з'єднаних між собою стрілками, які вказують послідовність виконання і зв'язок між блоками. Всередині блоків записується їх короткий зміст.

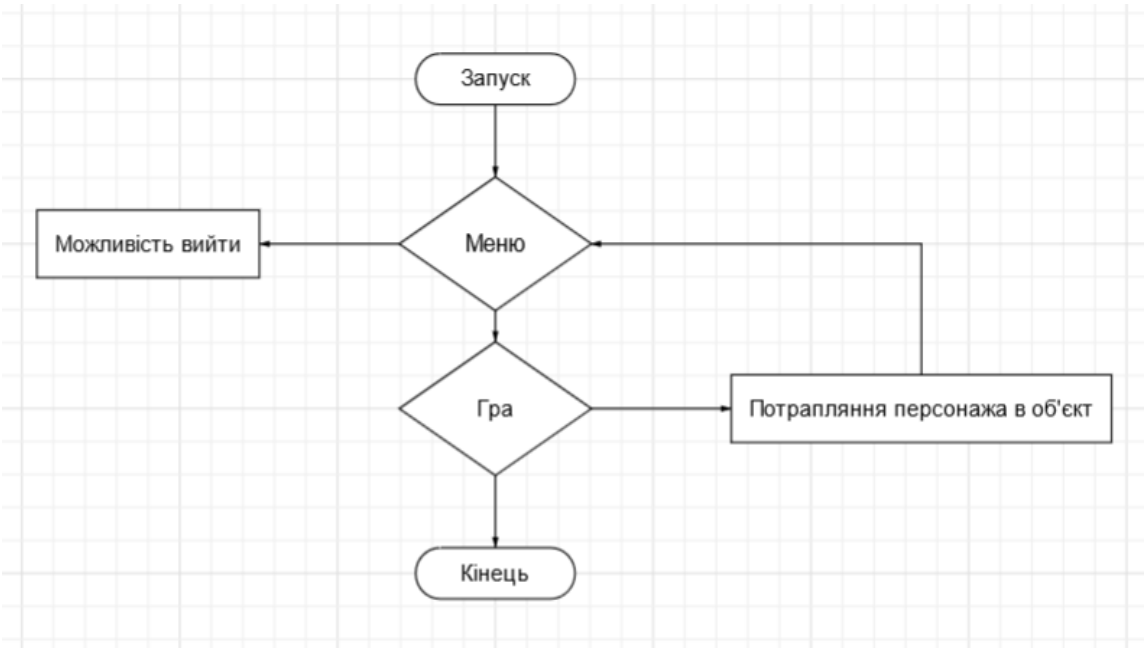


Рисунок 2.2. Блок схема для гри “Змійка”

Основні види блок-схем:

- лінійні (нерозгалужені);

- розгалужені;

- циклічні;

- з підпрограмами;

- змішані.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених межею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та

відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть діаграми прецедентів полягає в тому, що проєктована система подається у вигляді множини сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (англ. use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система під час діалогу з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

- асоціації (англ. association relationship)
- включення (англ. include relationship)
- розширення (англ. extend relationship)
- узагальнення (англ. generalization relationship)

При цьому загальні властивості варіантів використання можна подати трьома різними способами, а саме — за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації — одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується під час побудови всіх графічних моделей систем у формі канонічних діаграм.

Включення (англ. include) у мові UML — це різновид відношення залежності між базовим варіантом використання і його окремим випадком. При цьому відношенням залежності (англ. dependency) є таке відношення між двома елементами моделі, за якого зміна одного елемента (незалежного) спричиняє зміну іншого елемента (залежного).

Відношення розширення (англ. extend) визначає взаємозв'язок базового варіанту використання з іншим варіантом використання, функціональна поведінка якого залучається базовим не завжди, а тільки за виконання додаткових умов.

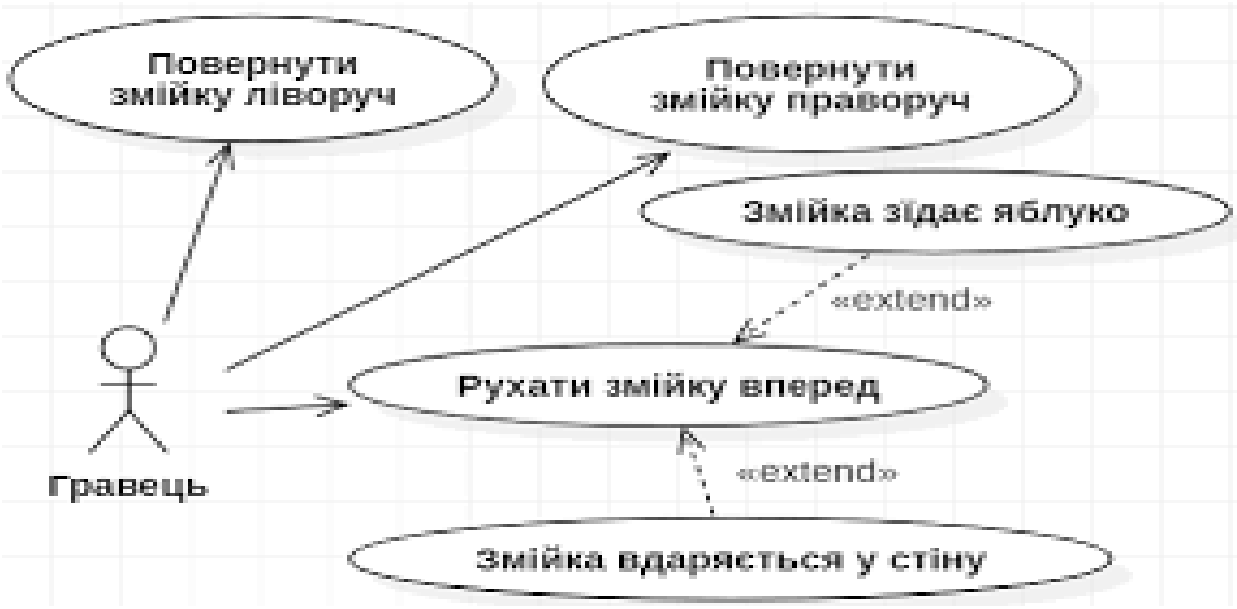


Рисунок 2.3 Діаграма прецедентів для гри “Змійка”

Діагра́ма кла́сів (англ. class diagram) — статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів наведена на Рис 2.4 та слугує для представлення статичної структури моделі системи в термінології класів об'єктноорієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини

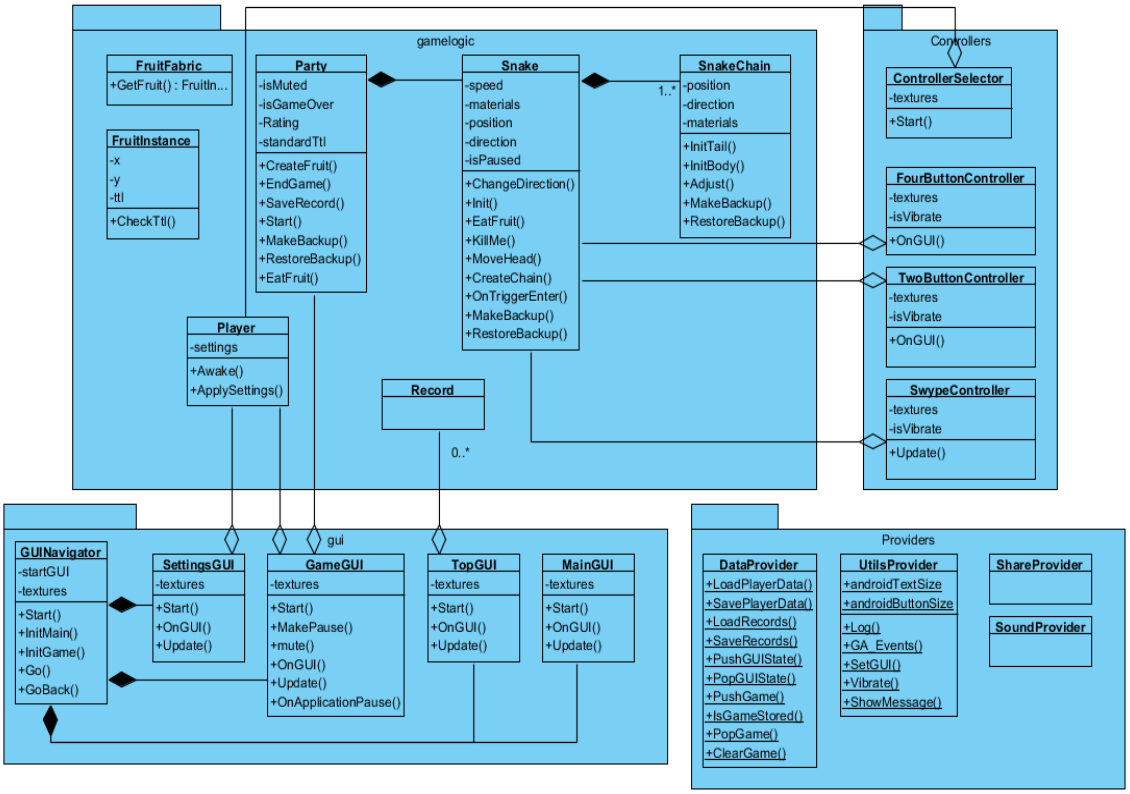


Рисунок 2.4. Діаграма класів гри “Змійка”

Асоціація показує, що об'єкти однієї сутності (класу) пов'язані з об'єктами іншої сутності.

Якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарною. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів; вони називаються n-арними. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносин. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність

посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це «обличчя», яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Ви можете явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді. Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: «нуль або одиниця» (0..1), «багато» (0 .. *), «одиниця або більше» (1.. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає «будь-яке число об'єктів, крім 2 і 5».

Агрегація

Приклад агрегації: професор може бути пов'язаний з одним або більше класів, в той час кожен клас пов'язаний лише з одним професором

Агрегація — проста асоціація між двома класами, де один з класів має вищий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і її зображують як просту асоціацію з незафарбованим ромбом з боку «цілого».

За потреби можна вказувати кратність агрегації.

Агрегація не накладає жорстких умов на термін існування об'єктів. Наприклад, «частини» можуть існувати тоді, коли «ціле» зникає.

Графічно агрегація представлена порожнім ромбом на блоці «цілого», і лінією, яка проведена від цього ромба до класу, що міститься в ньому («частин»).

Композиція

Приклад відношення композиції між Університетом та Підрозділом, яка чітко вказує, що об'єкти Підрозділу не існують, якщо не існує Університет, якому вони належать

Композиція — більш строгий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та екземплярів класів що містяться в ньому. Якщо контейнер буде знищений, то весь його вміст буде також знищено.

Графічно представляється як і агрегація, але з зафарбованим ромбиком.

Відмінності між композицією і агрегацією

Відносини між класом Виш і класами Студент і Факультет злегка відрізняються один від одного, хоча обидва є відносинами агрегування. У виші може бути будь-яка кількість студентів (включаючи нуль), і кожен студент може навчатися в одному або декількох вишах; виш може складатися з одного або декількох факультетів, але кожен факультет належить одному і лише одному вишу. Відношення між класами Виш і Факультет називають композицією, оскільки при знищенні моделі Виш моделі факультетів, що належать цьому ВНЗ, також будуть знищені. Студент і Виш пов'язані агрегацією тому, що Студента не можна видалити при знищенні Вишу.

Дані діаграми спрощують розробку та допомогають розробникам створювати ігрові застосунки

РОЗДІЛ 3. ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1.Тестування додатку

Після розробки програмного забезпечення переходимо до його тестування.

Користувач запускає додаток та переходить у меню.

Рисунок 3.1-Головне меню

Після вибору кнопки Play відбувається ігровий процес.

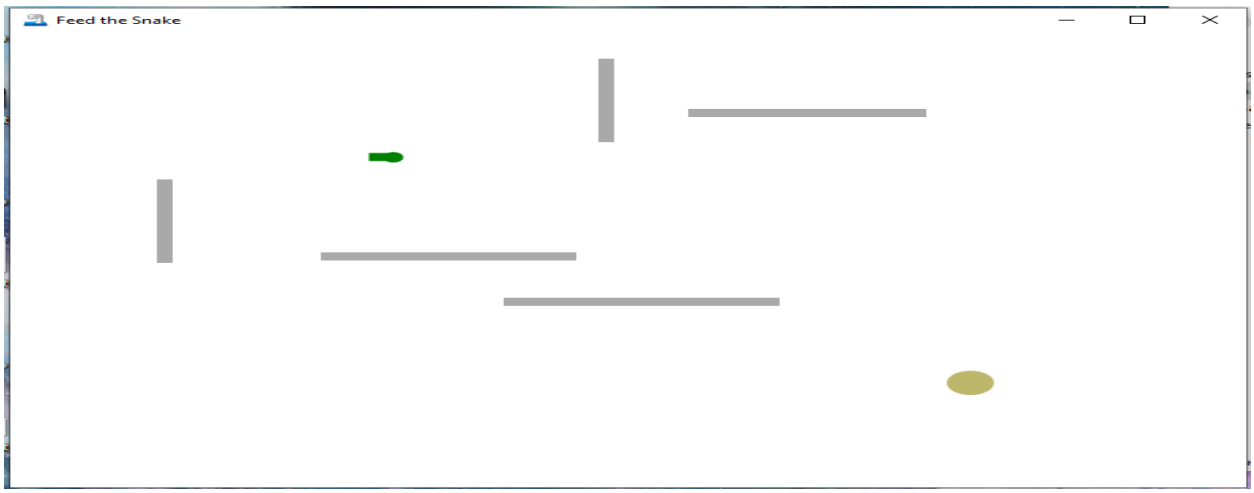


Рисунок 3.2-Гра

Перевіряємо функцію виводу результату та збереження рекорду.

Рисунок 3.3-Вивід результату

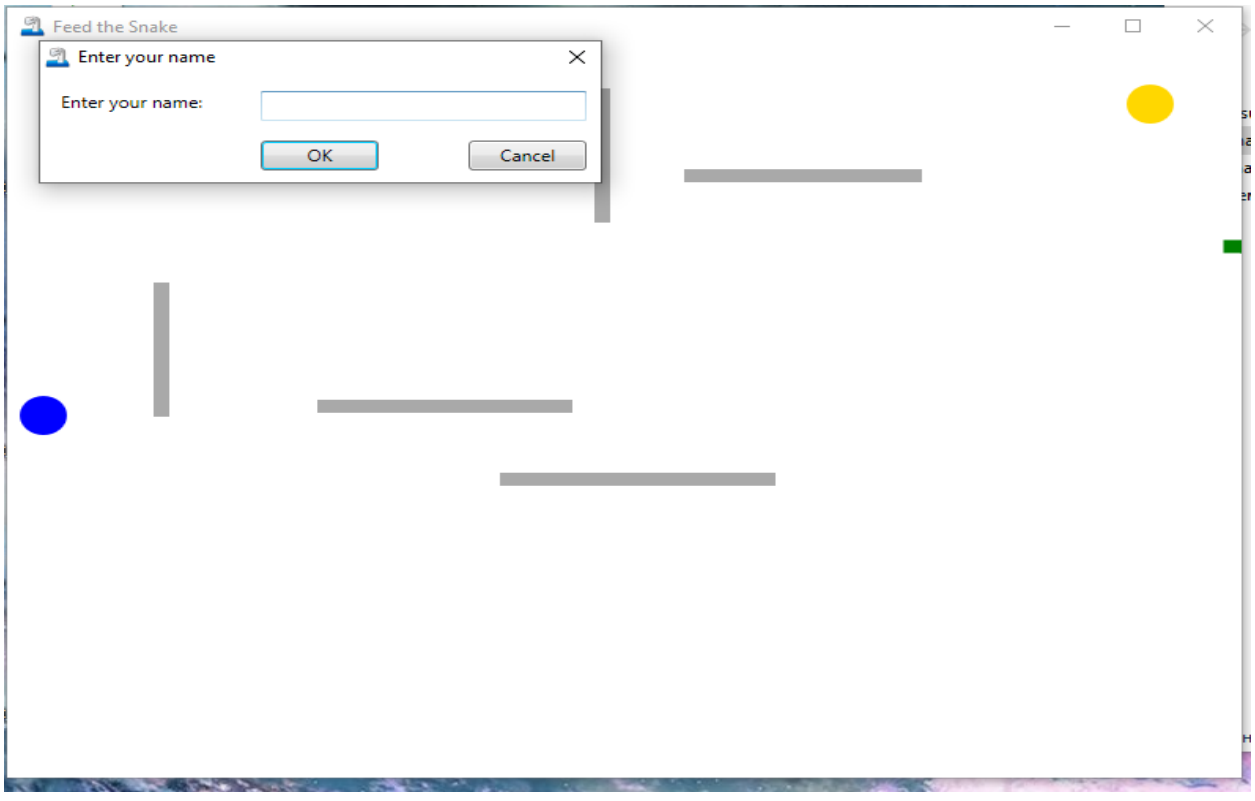


Рисунок 3.4-Збереження рекорду

3.2. Інструкція користувача

Спочатку відбувається запуск додатку та користувач переходить у головне меню. Через головне меню користувач має доступ до таких функцій:

- Гра
- Опції налаштування
- Результат
- Інструкція
- Вихід

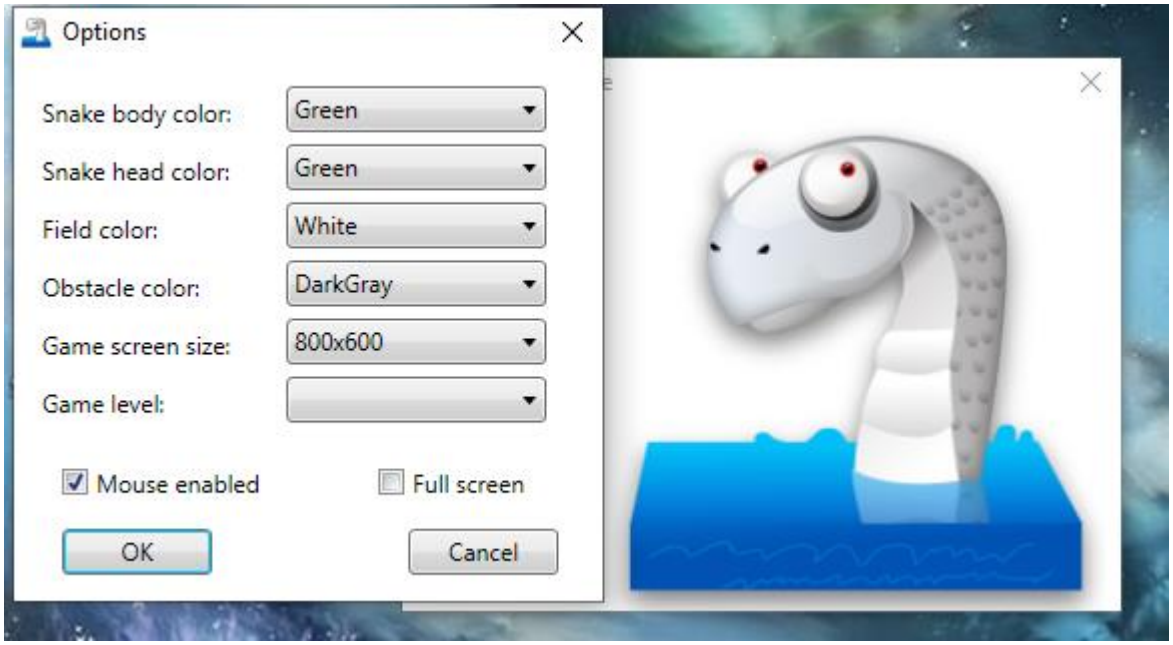


Рисунок 3.5-Налаштування гри

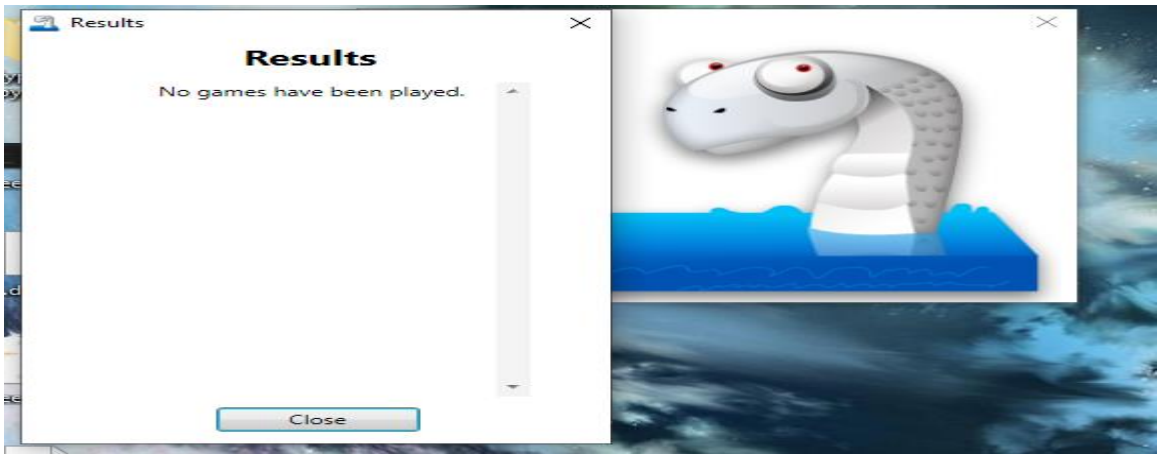


Рисунок 3.6-Результат гравця

Основна діяльність відбувається після натиску кнопки Play де користувач має змогу управляти змією, накопичувати очки та бити рекорди інших гравців.

ВИСНОВКИ

1. Проведено дослідження розробки ігор засобами об’єктно-орієнтовних мов програмування

2. Розвинуто навички взаємодії з мовою Python

3. Розроблено та протестовано ігровий додаток “Змійка”

Результатом проекту стала функціональна та цікава гра, в яку можна грати користувачам будь-якого рівня кваліфікації. Дослідження дало цінну інформацію про процес розробки ігор за допомогою Python і Pygame, а також допомогло доповнити наявні знання про розробку ігор. Гра також може мати практичне застосування в освіті та розвагах і може бути вдосконалена та модифікована для створення нових ігор або програм.

Перспективи розвитку роботи:

1. Адаптація гри під мобільні додатки

2. Додавання більш технологічної графіки

3. Створення системи монетизації

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1.https://www.oracle.com/java/

2.https://www.lemondeinformatique.fr/actualites/lire-java-open-source-c-est-fait-et-c-est-en-gpl-21350.html

3.Top Programming Languages 2020 (англ.), IEEE Spectrum.

4.TIOBE Index | TIOBE — The Software Quality Company. www.tiobe.com.

5.Buyya. Object-oriented Programming with Java: Essentials and Applications. — Tata McGraw-Hill Education, 2009. — 678 с. — ISBN 9780070669086.

6.Java 6 -server speed ÷ C++ GNU g++ speed | Computer Language Benchmarks Game

7.Metz, Cade. Google pits C++ against Java, Scala, and Go (англ.). The Register (3 июня 2011

8.Robert Tolksdorf. Programming languages for the Java Virtual Machine JVM (англ.).

9.SUN SHIPS JDK 1.1 -- JAVABEANS INCLUDED Ada 83 LRM, Sec 12.1: Generic Declarations. archive.adaic.com.

10.JavaFX FAQ. www.oracle.com.

11.Smith, Donald. The Future of JavaFX and Other Java Client Roadmap Updates.

12.OpenJDK: Project Coin. openjdk.java.net.

13.Index corruption and crashes in Apache Lucene Core / Apache Solr with Java 7 Акторное расширение языка Java в среде MPS

14.What's New in JDK 8. www.oracle.com.

15.Method References (англ.). The Java™ Tutorials. docs.oracle.com.

16.JDK 9 release delayed another four months.

17.Oracle Announces Java SE 9 and Java EE 8. Press Release (англ.). Project Jigsaw (англ.). openjdk.java.net.

ДОДАТКИ

ДОДАТОК А

```
import math

import random

import pygame

import random

import tkinter as tk

from tkinter import messagebox
```

#Гра змійка має меню та складається з таких позицій Play (Ігровий процес), options (Налаштування для системи), score (Збереження рекордів), help (Посібник допомоги для користувача) та exit (Вихід). Використано для графіки PyGame-Необхідне встановлення модуля

```
width = 500

height = 500

cols = 25

rows = 20

class cube():

    #Налаштування вікна

    rows = 20

    w = 500

    #Управління змією клавішами

    def __init__(self, start, dirnx=1, dirny=0, color=(255,0,0)):

        self.pos = start
```

```
        self.dirnx = dirnx
        self.dirny = dirny # "L", "R", "U", "D"
        self.color = color
    def move(self, dirnx, dirny):
        self.dirnx = dirnx
        self.dirny = dirny
        self.pos  = (self.pos[0] + self.dirnx, self.pos[1] + self.dirny)
    def draw(self, surface, eyes=False):
        dis = self.w // self.rows
        i = self.pos[0]
        j = self.pos[1]
        pygame.draw.rect(surface, self.color, (i*dis+1,j*dis+1,dis-2,dis-2))
        if eyes:
            centre = dis//2
            radius = 3
            circleMiddle = (i*dis+centre-radius,j*dis+8)
            circleMiddle2 = (i*dis + dis -radius*2, j*dis+8)
            pygame.draw.circle(surface, (0,0,0), circleMiddle, radius)
            pygame.draw.circle(surface, (0,0,0), circleMiddle2, radius)
```

```
class snake():
    body = []
    turns = {}

    def __init__(self, color, pos):
        #pos is given as coordinates on the grid ex (1,5)
        self.color = color
        self.head = cube(pos)
        self.body.append(self.head)
        self.dirnx = 0
        self.dirny = 1
    #Калькулятор набраних очків
    def move(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
            keys = pygame.key.get_pressed()
            for key in keys:
                if keys[pygame.K_LEFT]:
                    self.dirnx = -1
                    self.dirny = 0
                    self.turns[self.head.pos[:]] = [self.dirnx,self.dirny]
```

```
            elif keys[pygame.K_RIGHT]:
                self.dirnx = 1
                self.dirny = 0
                self.turns[self.head.pos[:]] = [self.dirnx,self.dirny]
            elif keys[pygame.K_UP]:
                self.dirny = -1
                self.dirnx = 0
                self.turns[self.head.pos[:]] = [self.dirnx,self.dirny]
            elif keys[pygame.K_DOWN]:
                self.dirny = 1
                self.dirnx = 0
                self.turns[self.head.pos[:]] = [self.dirnx,self.dirny]
        for i, c in enumerate(self.body):
            p = c.pos[:]
            if p in self.turns:
                turn = self.turns[p]
                c.move(turn[0], turn[1])
                if i == len(self.body)-1:
                    self.turns.pop(p)
            else:
                c.move(c.dirnx,c.dirny)
```

```
  #Результат
def reset(self,pos):
    self.head = cube(pos)
    self.body = []
    self.body.append(self.head)
    self.turns = {}
    self.dirnx = 0
    self.dirny = 1
def addCube(self):
    tail = self.body[-1]
    dx, dy = tail.dirnx, tail.dirny
    if dx == 1 and dy == 0:
        self.body.append(cube((tail.pos[0]-1,tail.pos[1])))
    elif dx == -1 and dy == 0:
        self.body.append(cube((tail.pos[0]+1,tail.pos[1])))
    elif dx == 0 and dy == 1:
        self.body.append(cube((tail.pos[0],tail.pos[1]-1)))
    elif dx == 0 and dy == -1:
        self.body.append(cube((tail.pos[0],tail.pos[1]+1)))
    self.body[-1].dirnx = dx
    self.body[-1].dirny = dy
```

```
    def draw(self, surface):
        for i,c in enumerate(self.body):
            if i == 0:
                c.draw(surface, True)
            else:
                c.draw(surface)
def redrawWindow():
    global win
    win.fill((0,0,0))
    drawGrid(width, rows, win)
    s.draw(win)
    snack.draw(win)
    pygame.display.update()
    pass
def drawGrid(w, rows, surface):
    sizeBtwn = w // rows

    x = 0
    y = 0
    for l in range(rows):
        x = x + sizeBtwn
        y = y +sizeBtwn
```

```
        pygame.draw.line(surface, (255,255,255), (x, 0),(x,w))
        pygame.draw.line(surface, (255,255,255), (0, y),(w,y))
def randomSnack(rows, item):
    positions = item.body
    while True:
        x = random.randrange(1,rows-1)
        y = random.randrange(1,rows-1)
        if len(list(filter(lambda z:z.pos == (x,y), positions))) > 0:
            continue
        else:
            break

    return (x,y)

#Функції в меню
def main():
    global s, snack, win
    win = pygame.display.set_mode((width,height))
    s = snake((255,0,0), (10,10))
    s.addCube()
    snack = cube(randomSnack(rows,s), color=(0,255,0))
```

```
        flag = True
        clock = pygame.time.Clock()
        while flag:
            pygame.time.delay(50)
            clock.tick(10)
            s.move()
            headPos = s.head.pos
            if headPos[0] >= 20 or headPos[0] < 0 or headPos[1] >= 20 or headPos[1] <
0:
                print("Score:", len(s.body))
                s.reset((10, 10))
            if s.body[0].pos == snack.pos:
                s.addCube()
                snack = cube(randomSnack(rows,s), color=(0,255,0))
            for x in range(len(s.body)):
                if s.body[x].pos in list(map(lambda z:z.pos,s.body[x+1:])):
                    print("Score:", len(s.body))
                    s.reset((10,10))
                    break
            redrawWindow()
    main()
```

ДОДАТОК Б

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка гри «Змійка» жанру аркада мовою Python

Виконав студент 5 курсу
Групи ППЗ-51
Носач Дмитро Русланович
Керівник роботи
К.е.н, доц, доцент кафедри ІПЗ Аверічев Ігор Миколайович
Київ – 2023

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи.** Покращення навичок роботи з Python та реалізація ігрового додатку
- **Об'єкт дослідження.** Розробка ігрових додатків та особливості мови Python
- **Предмет дослідження.** Ігровий додаток створений за допомогою Python

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Дослідити тему пов`язану з мовою програмування ООП
2. Відточити навички роботи з Python
3. Розробити ігровий додаток «Змійка» в жанрі аркада

3

АНАЛІЗ АНАЛОГІВ

	Можливість нагодувати змійку	Додаткові перешкоди на полі	Вибір кольору змійки та поля	Таблиця лідерів
	+	+	-	+
	+	-	+	+
	+	+	+	+

4

ВИМОГИ ДО ДОДАТКУ

Функціональні вимоги:

1. Рух змійки: Змійка повинна мати можливість рухатися в різних напрямках (вгору, вниз, вліво, вправо) на основі вводу користувача або заздалегідь визначених елементів керування.
2. Генерація їжі: Гра повинна випадковим чином генерувати їжу на ігровому полі для змійки, яку вона може з'їсти.
3. Зростання змії: Коли змія їсть їжу, вона повинна рости в довжину, додаючи новий сегмент до свого тіла.
4. Виявлення зіткнень: Гра повинна виявляти зіткнення змійки з межами ігрового поля або з власним тілом, що призводить до завершення гри.
5. Система підрахунку очок: Гра повинна відстежувати очки гравця на основі кількості з'їдених змійкою продуктів харчування.

5

ВИМОГИ ДО ДОДАТКУ

Не функціональні вимоги:

1. Продуктивність: Гра повинна бути чуйною та мати плавні та плавні рухи, забезпечуючи безперебійний ігровий процес.
2. Інтерфейс користувача: Гра повинна мати інтуїтивно зрозумілий та зручний інтерфейс, з чіткими інструкціями та легко ідентифікованими ігровими елементами.
3. Графіка та звук: Гра повинна мати візуально привабливу графіку та додаткові звукові ефекти, що покращують загальний ігровий досвід.
4. Сумісність: Гра має бути сумісною з різними платформами та пристроями, такими як настільні комп'ютери, мобільні телефони та планшети.

6

ПРОГРАМНІ ЗАСОБИ ТА ІНСТРУМЕНТИ РЕАЛІЗАЦІЇ

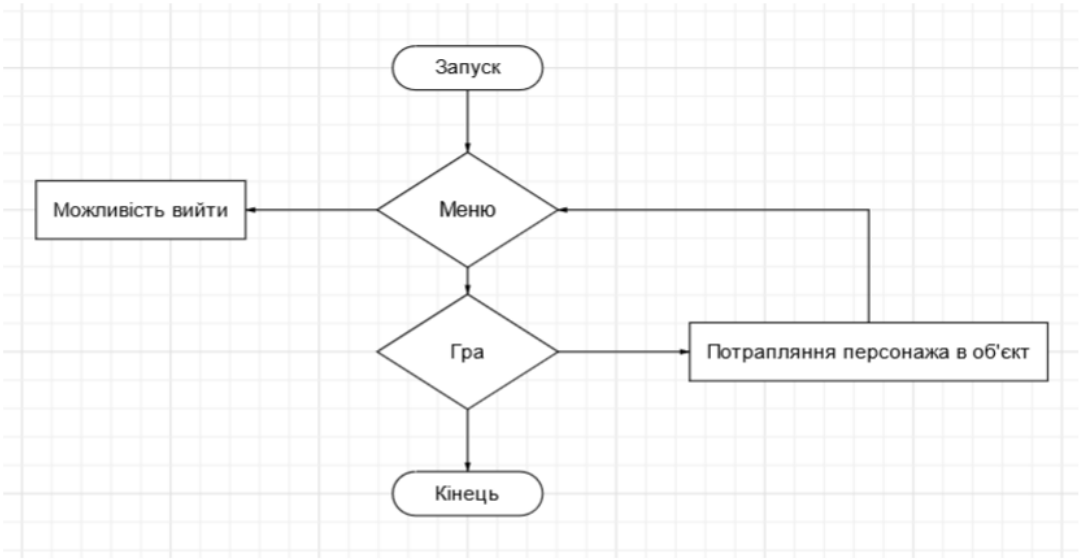
python

Visual Studio

pygame

7

БЛОК-СХЕМА

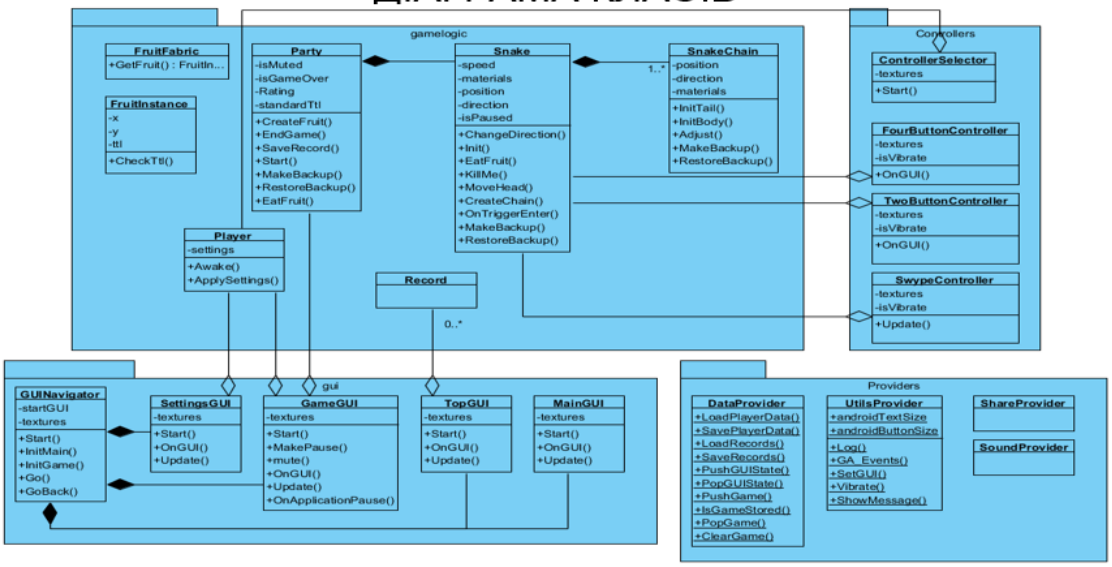


8

ДІАГРАМА ПРЕЦЕДЕНТІВ

9

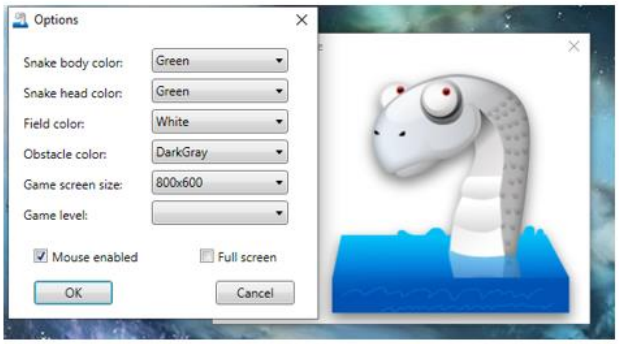
ДІАГРАМА КЛАСІВ



10

ЕКРАННІ ФОРМИ

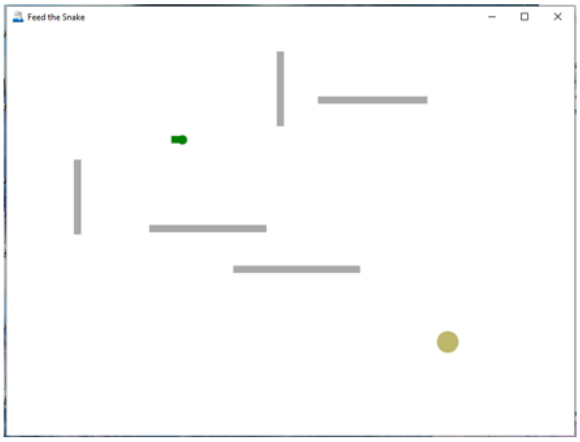
Головне меню гри



Налаштування

11

ЕКРАННІ ФОРМИ



Гра

12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Аверічев І.М., Носач Д.Р. Розробка гри «Змійка» жанру аркада мовою Python // Всеукраїнська науково-технічна конференція “Застосування програмного забезпечення в інфокомунікаційних технологіях” - Київ: ДУТ, 2023 - 128-129 с.;
2. Аверічев І.М., Носач Д.Р. Процес розробки ігор за допомогою Python та Pygame // Всеукраїнська науково-технічна конференція “Застосування програмного забезпечення в інфокомунікаційних технологіях” - Київ: ДУТ, 2023 - 130-131 с с..

13

ВИСНОВКИ

1. Проведено дослідження розробки ігор засобами об’єктно-орієнтовних мов програмування
2. Розвинуто навички взаємодії з мовою Python
3. Розроблено та протестовано ігровий додаток “Змійка”

14

ДЯКУЮ ЗА УВАГУ!