Міністерство освіти і науки України
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ПЕРВОМАЙСЬКИЙ ФАХОВИЙ КОЛЕДЖ»
Національного університету кораблебудування імені адмірала Макарова

Відділення «Інформаційних технологій»

Комп'ютерних технологій
(циклова комісія)

Пояснювальна записка

до дипломного проєкту
фаховий молодший бакалавр
(Освітньо-професійний ступень)

на тему «Створення комп'ютерної гри «SCAM»

Виконав: студент 4 курсу, групи 411-КІ-20
спеціальності
123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

«Комп'ютерні системи та мережі»
(освітня програма)

Михайлов О.В.
(прізвище та ініціали)
Керівник Островська І.О.
(прізвище та ініціали)
Рецензент Бесараб І.С.
(прізвище та ініціали)

Первомайськ - 2024 року

UNICHECK
by Turnitin

Ім'я користувача:
Олег В'ячеславович Калашніков

ID перевірки:
1016371611

Дата перевірки:
18.06.2024 12:37:06 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
21.06.2024 13:36:17 EEST

ID користувача:
100010544

Назва документа: Михайлов_О_В_411-КІ-20_2024_плагіат

Кількість сторінок: 53 Кількість слів: 8447 Кількість символів: 61953 Розмір файлу: 1.09 MB ID файлу: 10161788

39.6%
Схожість

Найбільша схожість: 7.72% з Інтернет-джерелом (https://qna.habr.com/q/949211)

36.6% Джерела з Інтернету | 733 ..... Сторінка 55

3.22% Джерела з Бібліотеки | 23 ..... Сторінка 58

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 32

Міністерство освіти і науки України
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ПЕРВОМАЙСЬКИЙ ФАХОВИЙ КОЛЕДЖ»
Національного університету кораблебудування імені адмірала Макарова

Відділення Інформаційних технологій
Циклова комісія Комп'ютерних технологій
Освітньо-професійний ступень фаховий молодший бакалавр
Галузь знань 12 «Інформаційні технології»
(шифр і назва)
Спеціальність 123 «Комп'ютерна інженерія»
(шифр і назва)
Освітня програма: «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Голова циклової комісії

_____ Островська І.О.
«16» травня 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ ЗДОБУВАЧУ ОСВІТИ

Михайлов Олександр Владиславович
(прізвище, ім'я, по батькові)

1. Тема проєкту *Створення комп'ютерної гри «SCAM»*

керівник проєкту Островська Ірина Олександрівна,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «12» квітня 2024 року № 46-у
2. Строк подання студентом проєкту 09.06.2024
3. Вихідні дані до проєкту текстові та графічні матеріали контенту

Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1 Дослідження та аналіз вимог до створення гри «SCAM»

1.1 Вимоги до створення комп'ютерних блокових ігор-головоломок

1.2 Методології та інструментарій створення комп'ютерних ігор

2 Створення гри «SCAM»

2.1 Алгоритм розв'язку задачі

2.2 Об'єктно-орієнтований аналіз

2.3 Діаграма станів

2.4 Об'єктно-орієнтоване проєктування

2.5 Об'єктно-орієнтоване програмування

3 Інструкція користувача

4 Охорона праці

5. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Островська І.О.	17.05.24	
2	Островська І.О.	17.05.24	
3	Островська І.О.	17.05.24	
4	Островська І.О.	17.05.24	

7. Дата видачі завдання 17.05.24

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Видача завдання	17.05.24	
2	Робота з літературою, збір інформації	19.05.24	
3	Оформлення загального розділу	22.05.24	
4	Оформлення спеціальних розділів	24.05.24	
5	Оформлення модулів програмного забезпечення	26.05.24	
6	Оформлення розділу з охорони праці	30.05.24	
7	Оформлення вступу, висновку	06.06.24	
8	Оформлення пояснювальної записки	09.06.24	

Здобувач освіти _____ Михайлов О.В.
(підпис) (прізвище та ініціали)

Керівник проєкту _____ Островська І.О.
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

«Комп'ютерна гра – взаємодія людини з комп'ютером або декількох людей між собою за допомогою комп'ютера для розваг, навчання чи тренування»[3].

У сучасному світі комп’ютерні ігри є однією з найбільших індустрій розваг. З кожним днем темпи їхнього розвитку зростають.

Мета дипломного проєкта – створення комп’ютерної 3D гри для платформи Windows на безкоштовній версії ігрового рушія Unity, використовуючи методи та засоби об’єктно-орієнтованого програмування.

Об’єктом дослідження цього проєкту є реалізація методів та засобів об'єктно-орієнтованого програмування мовою C# в розробці комп'ютерної гри за допомогою ігрового рушія Unity.

Предмет дослідження – розробка об‘єктно-орієнтованої моделі 3D комп'ютерної блокової гри-головоломки – ігрового додатку «Scam». Програма являє собою віконний продукт, написана мовою програмування високого рівня C# у середовищі розробки MicrosoftVisualStudio 2019.

У першому розділі мова йдеться про вимоги до розробки комп’ютерної гри. Наводиться опис існуючих аналогів блокових ігор, етапів створення ігор і методології їх розробки.

У розділі «Створення гри «Scam»» надано опис процесу розробки гри.

У розділі «Інструкція користувача» показані основні функції програми і дії, що вона виконує.

1 ДОСЛІДЖЕННЯ ТА АНАЛІЗ ВИМОГ ДО СТВОРЕННЯ ГРИ «SCAM»

Для реалізації об‘єктно-орієнтованої моделі гри потрібна ідея, а для реалізації ідеї потрібний план, а значить комплекс завдань та вимог.

Вимоги − це властивості, якими має володіти програмне забезпечення (далі – ПЗ) для адекватного визначення функцій, умов і обмежень виконання ПЗ, а також обсягів даних, технічного забезпечення та середовища функціонування.

Виявлення вимог – це процес вилучення інформації із різних джерел та проведення різних технічних зборів для формування окремих вимог до продукту та процесу розробки. Виконавець повинен узгодити вимоги з замовником.

Ідея гри полягає в емуляції боротьбі капітана з гуманоїдами на космічному кораблі.

Темою даного проєкту є створення 3D гри «SCAM».

Основне завдання – надати грі особливу атмосферу та реалістичний вигляд.

Гру потрібно створити за трасою послідовності подій. Під час реалізації проєкту можуть виникнути принципові труднощі, пов‘язані з недооцінкою поставленої задачі. У цих випадках треба критично переглянути постановку задачі.

Використання об'єктно-орієнтованого програмування спрощує нам завдання. Об'єктно-орієнтоване програмування − це підхід до розробки програмного забезпечення, заснований на об'єктах. Цей підхід дозволяє максимізувати принципи модульності і приховування інформації. Об'єктно-орієнтоване програмування базується на зв'язуванні або інкапсуляції структур даних і процедури, яка працює з даними в структурі, з модулем. Об'єктно-орієнтований принцип розробки дає багато переваг. Наприклад, кожен об'єкт інкапсулює його структуру даних з процедурою, використовуваної для роботи з екземплярами структури даних. Це дозволяє усунути в коді програми внутрішні залежності, які можуть швидко привести до того, що цей код буде важко

обслуговувати. Об'єкти можуть також успадковувати функції з породженого раніше об'єкта структури даних та інші характеристики, що дозволяє заощадити зусилля і забезпечити прозоре використання для багатьох цілей великих фрагментів коду. Об’єктно-орієнтоване програмування є способом організації програми. Цей підхід має кілька ключових концепцій - об’єкти і класи, інкапсуляція, наслідування та поліморфізм/ Основною ідеєю об’єктно-орієнтованого підходу є об’єднання даних і дій, виконуваних над цими даними, в єдине ціле, яке називається об’єктом. Т. б. головним компонентом об’єктно-орієнтованої програми є об’єкт. І замість того, щоб розглядати програму як набір послідовно виконуваних інструкцій, в ООП програма представляється у вигляді сукупності об’єктів. В програмуванні клас також може породити безліч підкласів. Ця можливість називається наслідуванням. Наслідування – це можливість, яка дозволяє одному класу наслідувати властивості іншого. В C# клас, що породжує всі інші класи, який наслідується, називається базовим класом. Класи, які наслідують базовий клас, називаються похідними. Базовий клас містить елементи, спільні для групи похідних класів. А похідні класи наслідують всі його властивості, одночасно володіючи власними властивостями. Т. б. кожен похідний клас представляє собою спеціалізовану версію базового класу. Роль наслідування в ООП – скоротити розмір коду та спростити зв’язки між елементами програми. В ООП концепція наслідування відкриває нові можливості програмування. А саме можливості повторного використання коду. Мова йде про те, що розроблений клас може бути використаний в інших програмах. Ця властивість називається можливістю повторного використання коду. Аналогічною властивістю в процедурному програмуванні володіють бібліотеки функцій, які можна включати в різноманітні програмні проєкти. Програміст може взяти існуючий клас, та, нічого не змінюючи, додати до нього свої елементи. Всі похідні класи унаслідують ці зміни, й в той же час кожен з похідних класів можна модифікувати окремо. Легкість повторного використання коду вже написаних програм є важливим достоїнством ООП.

Одним з достоїнств об’єктів є те, що вони дають користувачеві можливість створювати свої власні типи даних. Використання операцій та функцій різним чином в залежності від того, з якими типами даних вони працюють, називається поліморфізмом. [4]

Вибір мови програмування. Мова C# є в даний час найбільш поширена і перспективна мова програмування. Вона містить найбільш повний набір властивостей і можливостей, вироблених всією історією розвитку мов програмування. До істотних характерних властивостей C# слід віднести перш за все потужну підтримку об'єктно-орієнтованого підходу до розробки програм і механізму параметризації типів і алгоритмів. Широкий діапазон типів і розвинені можливості побудови призначених для користувача типів, дозволяють адекватно відобразити особливості предметної області; строгі правила поводження з константними типами сприяють надійності програм. Підвищенню надійності створюваних програм служить простий і гнучкий апарат управління винятковими ситуаціями. Розвинені схеми перетворення і приведення типів дозволяють забезпечити достатній компроміс між строгою типізацією і ефективністю виконання програм. Засоби явного управління надають зручний механізм структурування великих програм. C# є прямим спадкоємцем мови C і фактично включає його як підмножина. Тим самим, C# цілком містить добре зарекомендувала себе традиційну модель обчислень мови C, в тому числі, розвинений алгоритмічний базис, широкі можливості конструювання нових типів і гнучкі засоби роботи з пам'яттю, включаючи арифметику над покажчиками. Ця обставина забезпечує збереження в актуальному стані мільйони рядків програмного тексту, розробленого на C, і дає додаткові гарантії широкого використання C#. [4]

Для реалізації даної гри використовується середовище програмування Unity3d.

Unity – багатоплатформовий інструмент для розробки дво- та тривимірних додатків та ігор, що працює на операційних системах Windows та OS X.

Редактор Unity має простий інтерфейс, який легко налаштовувати, що складається з різних вікон, завдяки чому можна проводити налагодження гри прямо в редакторі. Рушій підтримує три сценарних мови: C#, JavaScript (модифікація), Boo (мова програмування, подібна до Python). Проєкт в Unity ділиться на сцени (рівні) − окремі файли, що містять свої ігрові світи зі своїм набором об'єктів, сценаріїв, і налаштувань. Сцени можуть містити в собі як, об'єкти (моделі), так і порожні ігрові об'єкти − тобто ті, які не мають моделі. Об'єкти, в свою чергу містять набори компонентів, з якими і взаємодіють скрипти. Також у них є назва (в Unity допускається наявність двох і більше об'єктів з однаковими назвами), може бути тег (мітка) і шар, на якому він повинен відображатися. Так, у будь-якого предмета на сцені обов'язково присутній компонент Transform − він зберігає в собі координати місця розташування, повороту і розмірів по всіх трьох осях. У об'єктів з видимою геометрією також за замовчуванням присутній компонент Mesh Renderer, що робить модель видимою. Також Unity підтримує фізику твердих тіл і тканини, фізику типу Ragdoll (ганчіркова лялька). У редакторі є система успадкування об'єктів; дочірні об'єкти будуть повторювати всі зміни позиції, повороту і масштабу батьківського об'єкта. Скрипти в редакторі прикріплюються до об'єктів у вигляді окремих компонентів. При імпорті текстури в рушій можна згенерувати alpha-канал, mip-рівні, normal-map, light-map, карту відображень, проте безпосередньо на модель текстуру прикріпити не можна − буде створено матеріал, з яким буде призначений шейдер, і потім матеріал прикріпиться до моделі. Редактор Unity підтримує написання і редагування шейдерів. Крім того він містить компонент для створення анімації, яку також можна створити попередньо в 3D-редакторі та імпортувати разом з моделлю, а потім розбити на файли. [4]

На сьогоднішній день індустрія комп’ютерних ігор у всьому світі є найбільшим сегментом світового ринку цифрового контенту, створюючи щорічно багатомільярдні доходи та залучаючи величезну аудиторію [1].

Комп’ютерна гра – це відеогра, яка призначена для роботи на персональному комп’ютері. Одним із популярних жанрів комп’ютерних ігор є roguelike-ігри та платформери. Roguelike-ігри – це жанр комп’ютерних ігор, характерними особливостями якого є процедурне створення ігрових рівнів, покроковий ігровий процес та перманентна смерть персонажа у випадку поразки [1].

При розробці сучасних комп’ютерних ігор найбільш продуктивною є технологія гральних рушіїв, яка покликана спростити процес розробки за рахунок уніфікації та систематизації внутрішньої структури гри. Гральний рушій є комплексом програмних компонентів, які відповідають за реалізацію основних функціональних можливостей гри: візуалізацію ігрової сцени, симуляцію фізичних законів реального світу у віртуальному, відтворення звуку, створення ілюзії інтелекту в поведінці ігрових персонажів, анімацію. Невід’ємними складовими гральних рушіїв є система скрипів, мережевий код, засоби керування пам’яттю, багатопотоковість та граф сцени [2].

Unity являє собою професійний гральний рушій, який використовується при створенні відеоігор для різноманітних платформ. Це інструмент, яким щодня користуються досвідчені розробники, а також один із найдоступніших інструментів для новачків [3].

Unity дозволяє процедурно створювати ігрові рівні як в 2D, так і в 3D вимірах. Зручний режим роботи зі спрайтами (двовимірне зображення, що застосовується в комп'ютерній графіці) в поєднанні зі спеціалізованими скриптами полегшують генерацію 2D грального рівня. Генерація 3D грального рівня реалізується шляхом маніпулювання ландшафтами (terrain) та інстанціонування наборів (prefabs), які включать 3D-об’єкти навколишнього ігрового середовища з відповідними характеристиками.

Алгоритмічна побудова 3D-ландшафту можлива двома методами. Перший метод включає в себе створення скрипта для використання вбудованого в гральний рушій редактора ландшафтів. Другий спосіб полягає в побудові сітки ландшафту (mesh). Даний метод дає ширший набір дій над ландшафтом, але є

складнішим ніж перший: створення сітки включає побудову первинного мешу, подальше його розбиття на трикутники та написання шейдерів (це програма для одного із ступенів графічного конвеєра, що використовується в тривимірній графіці для визначення остаточних параметрів об'єкта чи зображення) для створеного ландшафту.

Також важливим аспектом ігрових рівнів є «критичний шлях». Критичний шлях – це відрізок, вздовж якого гравець повинен рухатись, якщо він хоче завершити рівень. Слід зазначити, що до «критичного шляху» не включають допоміжні квести. Для того, щоб гравець рухався вздовж відповідного відрізку потрібно передбачити коректне розміщення бонусів, неігрових персонажів (союзних, нейтральних чи ворожих) тощо [4].

Складним аспектом у створенні будь-якої комп’ютерної гри, особливо roguelike-гри, є створення системи пошуку шляху та навігації в ігровому середовищі неігрових персонажів. За допомогою наявних в Unity функцій штучного інтелекту з передовою системою автоматизованого пошуку шляху полегшується процес налаштування навігації в ігровому середовищі [5].

Отже, гральний рушій Unity є універсальним інструментом для розробки комп’ютерних ігор, і тому для створення гри «SCAM» було обрано саме його.

Жанрова класифікація комп’ютерних ігор охоплює широкий спектр різних типів ігор, кожна з яких має свої унікальні особливості, механіки гри та цільову аудиторію. Основні жанри комп’ютерних ігор: екшн, пригоди, рольові ігри, стратегії, симулятори, спортивні ігри, гонки, пазли, музичні і ритмічні ігри, карткові ігри та настільні ігри, інтерактивне кіно…

Екшн (Action):

– шутери (Shooters): Включають піджанри як FPS (First-Person Shooter) та TPS (Third-Person Shooter). Приклад: Call of Duty, Counter-Strike;

– файтинг (Fighting): Бійки між персонажами з використанням різноманітних прийомів. Приклад: Mortal Kombat, Street Fighter;

– платформери (Platformers): Ігри, де гравець керує персонажем, який стрибає по платформах і долає перешкоди. Приклад: Super Mario, Celeste;

– біт-ем-ап (Beat 'em Up): Ігри, де гравець бореться проти численних ворогів, часто використовуючи рукопашний бій. Приклад: Streets of Rage, Double Dragon.

Пригоди (Adventure):

– Квест (Quest): Гравець досліджує світ, збирає предмети, розгадує головоломки. Приклад: Monkey Island, King’s Quest;

– Survival Horror: Ігри з акцентом на виживання у ворожому середовищі та страх. Приклад: Resident Evil, Silent Hill.

Рольові ігри (RPG - Role-Playing Games):

– традиційні (Traditional RPG): Включають розвинену сюжетну лінію, розвиток персонажа і часто покрокові битви. Приклад: Final Fantasy, The Witcher;

– MMORPG (Massively Multiplayer Online RPG): Ігри, де велика кількість гравців взаємодіють у великому онлайн-світі. Приклад: World of Warcraft, Guild Wars.

Стратегії (Strategy):

– RTS (Real-Time Strategy): Гравці будують бази, збирають ресурси та керують арміями в реальному часі. Приклад: StarCraft, Age of Empires;

– TBS (Turn-Based Strategy): Гравці роблять ходи по черзі, плануючи дії наперед. Приклад: Civilization, XCOM.

Симулятори (Simulation):

– симулятори життя (Life Simulation): Ігри, що моделюють аспекти повсякденного життя. Приклад: The Sims.

– симулятори будівництва та управління (Construction and Management Simulation): Гравці будують та керують різними об'єктами або містами. Приклад: SimCity, Cities: Skylines.

– транспортні симулятори (Vehicle Simulation): Моделюють керування транспортними засобами. Приклад: Microsoft Flight Simulator, Euro Truck Simulator.

Спортивні ігри (Sports).

Ігри, що моделюють різні види спорту. Приклад: FIFA, NBA 2K.

Гонки (Racing)

Ігри, де гравець змагається у швидкості, керуючи транспортними засобами. Приклад: Need for Speed, Forza Horizon.

Пазли (Puzzle).

Ігри, що вимагають від гравця розв'язання різноманітних головоломок. Приклад: Tetris, Portal.

Музичні і ритмічні ігри (Music and Rhythm).

Ігри, де гравець виконує дії у ритм музиці. Приклад: Guitar Hero, Dance Dance Revolution.

Карткові ігри та настільні ігри (Card and Board Games)

Цифрові версії традиційних настільних і карткових ігор. Приклад: Hearthstone, Gwent.

Інтерактивне кіно (Interactive Movie)

Ігри, що містять багато кінематографічних сцен і де гравець впливає на сюжет. Приклад: Heavy Rain, Detroit: Become Human.

Кожен жанр може містити різноманітні піджанри, що робить класифікацію дуже гнучкою і дозволяє враховувати нові типи ігор, що з'являються з розвитком технологій.

Комп'ютерні блокові ігри-головоломки це жанр ігор, де основним елементом геймплею є використання блоків або елементів, щоб вирішити різноманітні логічні або фізичні головоломки. Такі ігри часто включають елементи будівництва, розміщення блоків у відповідних позиціях, розрізання блоків чи інших дій з ними для досягнення певної мети або розв'язання проблеми. Ось деякі ідеї та характеристики таких ігор:

Логічні головоломки з блоками: Гравцям може потрібно складати блоки або об'єднувати їх у певному порядку, щоб розгадати складні логічні головоломки. Наприклад, рухати блоки так, щоб створити шлях для персонажа або м'ячика до цілі.

Фізичні головоломки: Це може включати розміщення блоків таким чином, щоб вони виконували фізичні дії, наприклад, створення механічних пристроїв або пулінг блоків для спрацювання різних механізмів.

Будівельні завдання: Гравці можуть отримувати завдання на побудову об'єктів або структур, використовуючи обмежену кількість блоків або матеріалів. Вони повинні використовувати свою креативність та логічне мислення, щоб здійснити побудову.

Експерименти з фізикою: Деякі блокові ігри можуть моделювати реалістичну фізику, де гравці можуть створювати різні структури або механізми, які повинні взаємодіяти зі світом, щоб вирішити завдання.

Розширені можливості блоків: Деякі ігри можуть містити спеціальні типи блоків з унікальними властивостями, які гравці можуть використовувати для досягнення своїх цілей.

Кооперативний або конкурентний мультиплеєр: Деякі блокові головоломки можуть пропонувати можливість спільної гри з іншими гравцями,

де вони мають співпрацювати або конкурувати у вирішенні головоломок або будівництві.

Загалом, комп'ютерні блокові ігри-головоломки дозволяють гравцям не лише розвивати логічне мислення та творчість, але й насолоджуватися процесом будівництва та експериментування з різними структурами і механізмами у віртуальному середовищі.

Ігри Tetris (рис. 1.1) – це культові та вічні онлайн-ігри-головоломки, які викликають у гравців орієнтацію в просторі, швидкість мислення та стратегічні здібності. Мета полягає в тому, щоб управляти падаючими геометричними фігурами, які називаються тетроміно, коли вони спускаються з верхньої частини екрана. Гравці повинні обертати та розташовувати ці тетроміно, щоб утворити повні горизонтальні лінії, які потім очищаються з ігрової зони. Відмінною рисою ігор Tetris є їх простота та доступність, що робить їх привабливими для гравців будь-якого віку та рівня кваліфікації. Механіку ігрового процесу легко зрозуміти, але вона поступово стає складнішою, оскільки гра прискорюється, а домовленості стають складнішими. Однією з ключових особливостей тетрісу є використання стратегії та планування. Гравці повинні передбачити форми, які вони отримають, і визначити найкраще розташування, щоб запобігти пропускам і створити можливості для очищення ліній. Очищення кількох ліній одночасно (відоме як «тетріс») заробляє більше балів і сприяє підвищенню результатів. Ігри Tetris часто включають різноманітні режими, щоб задовольнити різні вподобання. Від класичного режиму зі зростаючими рівнями складності до викликів на основі часу та змагальних опцій для кількох гравців, гравці можуть вибрати стиль, який відповідає їх стилю гри. Онлайн-ігри Tetris дозволяють гравцям насолоджуватися цією позачасовою класикою на різних пристроях, від комп’ютерів до смартфонів. Зручність гри в Тетріс у будь-який час і будь-де робить його популярним вибором як для швидких ігрових сесій, так і для тривалого ігрового процесу. Якщо ви шукаєте гру, яка перевірить ваші рефлекси та стратегічне мислення, ігри Tetris пропонують доступний і захоплюючий

онлайн-досвід. У Тетрісі реалізовано світ упорядкування падаючих блоків, очищення ліній і досягнення високих результатів. Незалежно від того, чи ви новачок, чи любитель тетрісу, ці ігри забезпечують нескінченні години розваг і викликів на різних ігрових платформах. [7]

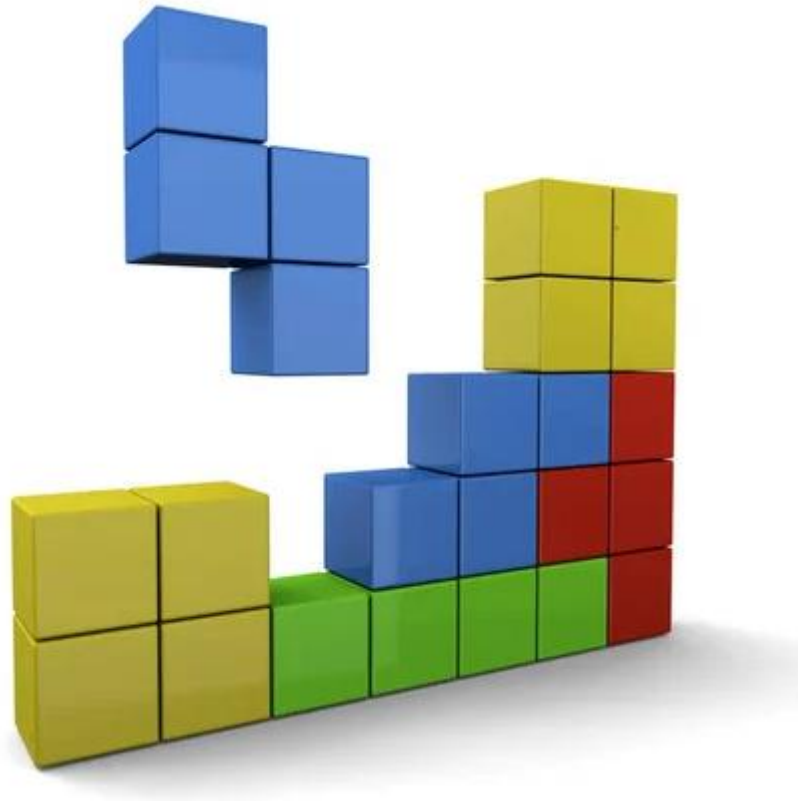

Рисунок 1.1 Тетріс

Minecraft – це видавничий тип відеографічної гри, розроблений шведським розробником Маркусом Перссоном, також відомим як Notch, та подальшими розробниками у компанії Mojang Studios. Гра вперше випущена у вересні 2011 року, і з тих пір стала однією з найпопулярніших і впливових відеоігор у світі.

Основна концепція Minecraft полягає в нескінченному світі, що складається з блоків, які представляють різні матеріали і ресурси, такі як земля, камінь, дерево, метал і так далі. Гравці в цій віртуальній реальності можуть взаємодіяти з цими блоками, збирати їх, створювати нові блоки і будувати різноманітні структури та об'єкти.

Основні аспекти концепції Minecraft (рис. 1.2) включають:

1. Будівництво і творчість: Гравці можуть будувати будь-які структури, які їм заманеться, використовуючи блоки різних матеріалів. Це може бути все, від простих хижин до складних міст чи навіть функціональних механізмів.

2. Дослідження і ресурси: У світі Minecraft гравці можуть досліджувати різні біоми, шукаючи ресурси, такі як руди для виробництва матеріалів і інструментів.

3. Виживання і боротьба: У більшості режимів гри є аспект виживання, де гравці повинні забезпечити собі їжу і пристосуватися до умов середовища. Також є можливість зустрічі із мобами — ворожими сутностями, які можуть атакувати гравця.

4. Мультиплеєр і співпраця: Minecraft підтримує гру в мультиплеєрному режимі, де гравці можуть спільно працювати над будівництвом, обмінюватися ресурсами і взаємодіяти у великих віртуальних світах.

5. Моди і спільнота: Гра має активну спільноту моддерів, які створюють різноманітні модифікації для гри, розширюючи її можливості і відкриваючи нові ігрові досвіди.

Концепція Minecraft виокремлюється своєю відкритістю і можливістю для гравців впливати на світ і створювати власні унікальні історії та пригоди у віртуальному просторі.

Гра Block Craft 3D є відомою за свою унікальну концепцію, яка поєднує елементи будівництва, творчості і соціальної взаємодії у віртуальному світі. Основна ідея гри полягає в тому, щоб гравці могли будувати різноманітні об'єкти, використовуючи блоки різних матеріалів, подібно до Minecraft. Однак Block Craft 3D має свої особливості: Концепція Minecraft виокремлюється своєю відкритістю і можливістю для гравців впливати на світ і створювати власні унікальні історії та пригоди у віртуальному просторі.

Рисунок 1.2 Minecraft

Гра Block Craft 3D є відомою за свою унікальну концепцію, яка поєднує елементи будівництва, творчості і соціальної взаємодії у віртуальному світі. Основна ідея гри полягає в тому, щоб гравці могли будувати різноманітні

об'єкти, використовуючи блоки різних матеріалів, подібно до Minecraft. Однак Block Craft 3D (рис. 1.3) має свої особливості:

1. Простий інтерфейс і управління: Гра спрощена у порівнянні з Minecraft, що робить її доступною для широкого кола користувачів, включаючи дітей.

2. Багатоцільовий геймплей: Гравці можуть будувати не тільки будинки, а й цілі міста, використовуючи різні блоки, які доступні в грі.

3. Спільнота і соціальні можливості: Гравці можуть ділитися своїми твореннями з іншими користувачами, відвідувати і оцінювати будівлі інших гравців. Це сприяє взаємодії і творчому обміну досвідом.

4. Мультимедійні можливості: Block Craft 3D також має функцію запису та демонстрації користувацьких відео, що дозволяє гравцям створювати контент для YouTube або інших платформ.

Головна ідея гри полягає в тому, щоб дозволити користувачам виражати свою творчість та будівельні навички у віртуальному середовищі, а також спілкуватися та співпрацювати з іншими гравцями.

Рисунок 1.3 Block Craft 3D

Block Craft 3D – це креативна будівельна гра, натхненна Minecraft, у яку ви можете грати онлайн і безкоштовно. Коли ви можете переставляти блоки, як завгодно, у величезному світі, повному різноманітних матеріалів, вашій творчості немає обмежень. Це саме те, що пропонує вам Block Craft 3D. Ця онлайн-гра «Створення та будівництво блоків» дає змогу збирати будь-які елементи, які тільки знайдете, і розміщувати їх у будь-якому місці, а це означає, що ви можете будувати будинки з каменю чи дерева, підземні печери з таємними ходами, гігантські скульптури чи навіть замок із красивий сад. Можна дати волю своєму внутрішньому архітектору та створити речі, про які до цього навіть не мріяли. Можна створити ціле місто і дати волю своїй уяві. Управління: стрілки / WASD = переміщення, миша = перегляд, клацання лівою кнопкою миші = збір, клацання правою кнопкою миші = місце, пробіл = стрибок, 2x пробіл = наведення, 1 - 9 = вибір слота. [7]

Stacking Boxes 2-4 Player — це весела гра з однією кнопкою для 4 гравців, у якій вам потрібно кидати коробки, щоб складати їх, доки не досягнете лінії. Ця захоплююча безкоштовна онлайн-гра пробудить у гравця змагальний дух. Завдання полягатиме в тому, щоб скидати коробки з надзвичайною точністю, щоб стопка коробок досягла пунктирної лінії та залишалася в стопці кілька секунд. Можна грати за невимушену жабу, круту корову, чарівного зайчика або стурбованого ведмедя панду та намагатися виграти кожну гру. Ви можете грати з 1, 2 або 3 іншими гравцями на тій самій клавіатурі або просто додати процесорних ботів. Кидайте свої коробки, щоб скласти їх, або навіть спробувати кинути коробку в своїх конкурентів поруч, щоб їх саботувати. Управління: W / J / Стрілка вгору / Миша [7]

1.1 Вимоги до створення комп'ютерних блокових ігор-головоломок

Створення комп'ютерних блокових ігор-головоломок вимагає деяких ключових елементів і підходів для досягнення успіху. Ось основні вимоги, які можна виділити:

1. Логічні головоломки і геймдизайн:

- інноваційні головоломки: Ігри повинні містити унікальні і цікаві головоломки, які стимулюють гравців до розв'язання логічних задач;
- прогресивна складність: Головоломки повинні збільшуватися за складністю по мірі проходження гри, щоб тримати гравців зацікавленими і змушувати їх вдосконалювати свої навички.

2. Фізика та взаємодія з об'єктами:

- реалістична фізика: Якщо в грі використовується фізика, вона повинна бути достатньо реалістичною, щоб гравці могли використовувати її для розв'язання головоломок і створення механізмів;.
- взаємодія з об'єктами: Гравці повинні мати можливість взаємодіяти з блоками чи об'єктами у грі, будувати, рухати або модифікувати їх для досягнення своїх цілей.

3. Креативність і будівництво:

- інструменти будівництва: Ігри повинні надавати гравцям інструменти для будівництва різних структур і механізмів;
- сприяння креативності: Важливо стимулювати гравців до креативності у створенні унікальних рішень і конструкцій.

4. Графіка та візуальний стиль:

- естетика блоків і світу: Візуальний стиль повинен бути привабливим і логічною частиною геймплею, що відображає тематику гри;
- чіткість і доступність: Графіка повинна бути достатньо чіткою, щоб гравці могли легко розрізняти різні блоки та об'єкти.

5. Оптимізація і продуктивність:

– ефективна робота з ресурсами: Важливо оптимізувати гру, щоб вона працювала ефективно на різних пристроях і не мала проблем з продуктивністю;

– масштабованість: Ігри мають бути здатні до масштабування, щоб підтримувати велику кількість блоків чи складних конструкцій без втрати продуктивності.

6. Звуковий дизайн і музика:

– атмосферний звук: Добре обрана музика та звуковий дизайн можуть значно покращити імерсію і геймплей гравців.

7. Моделювання спільноти:

– підтримка модів та спільнота: Підтримка модів і активна спільнота можуть додатково збагатити гру, надаючи гравцям нові можливості та контент.

Всі ці елементи разом формують основні вимоги до створення комп'ютерних блокових ігор-головоломок, які успішно залучають і зацікавлюють гравців своєю унікальністю та геймплейною механікою.

1.2 Методології та інструментарій створення комп’ютерних ігор

Створення комп'ютерних ігор – це складний і творчий процес, який вимагає використання спеціалізованих методологій та інструментарію. Є чотири основні аспекти методології та сім груп інструментів, які використовуються в процесі розробки комп'ютерних ігор.

Розглянемо методології розробки ігор.

1. Waterfall (Каскадна модель) (рис. 1.4):

– характеристики: Послідовний процес, де кожен етап розробки (аналіз, проєктування, реалізація, тестування) виконується послідовно.

– використання: Застосовується в ситуаціях, коли вимоги до гри стабільні і можуть бути чітко визначені на початку проєкту.

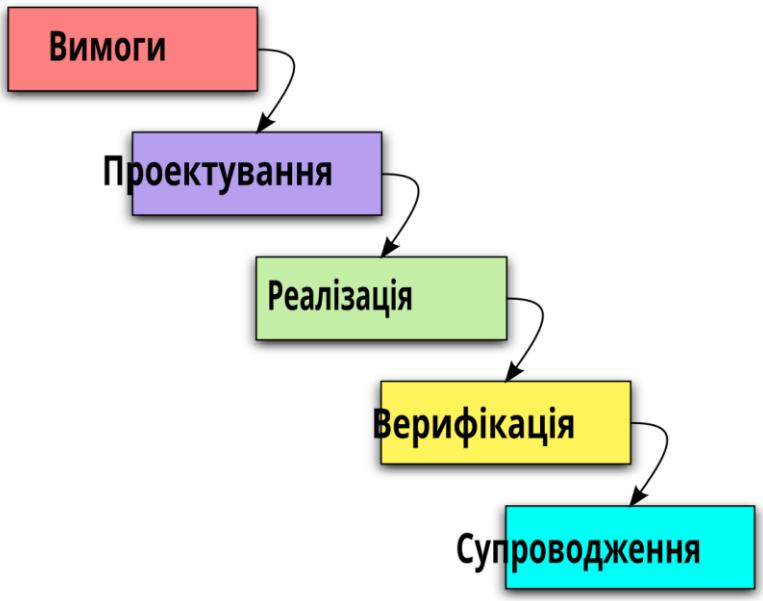

Рисунок 1.4 Методологія Waterfall

2. Agile (Гнучкі методології) (рис. 1.5):

– характеристики: Ітеративний процес розробки, де розробка поділяється на короткі цикли розробки (спринти);

– використання: Використовується в умовах, коли вимоги змінюються, і необхідно швидко адаптуватися до змін у вимогах чи умовах ринку.

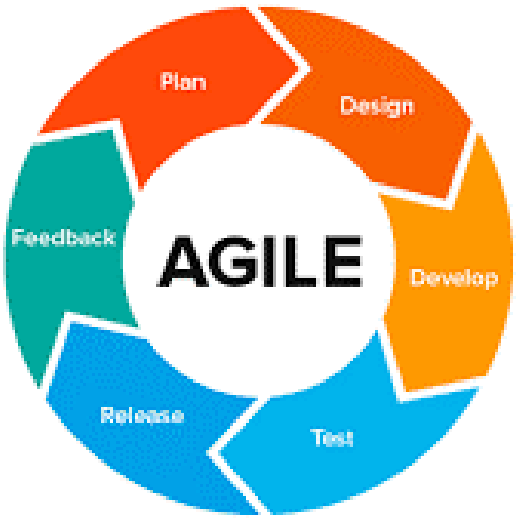

Рисунок 1.5 Методологія Agile

3. Scrum (рис. 1.6):

– характеристики: Форма гнучкої методології, де розробка поділяється на короткі, регулярні ітерації (спринти), кожен з яких закінчується готовим для випуску інкрементом продукту;

– використання: Часто використовується для команд розробників, які працюють над складними проєктами з високою ступенем змінюваності вимог.

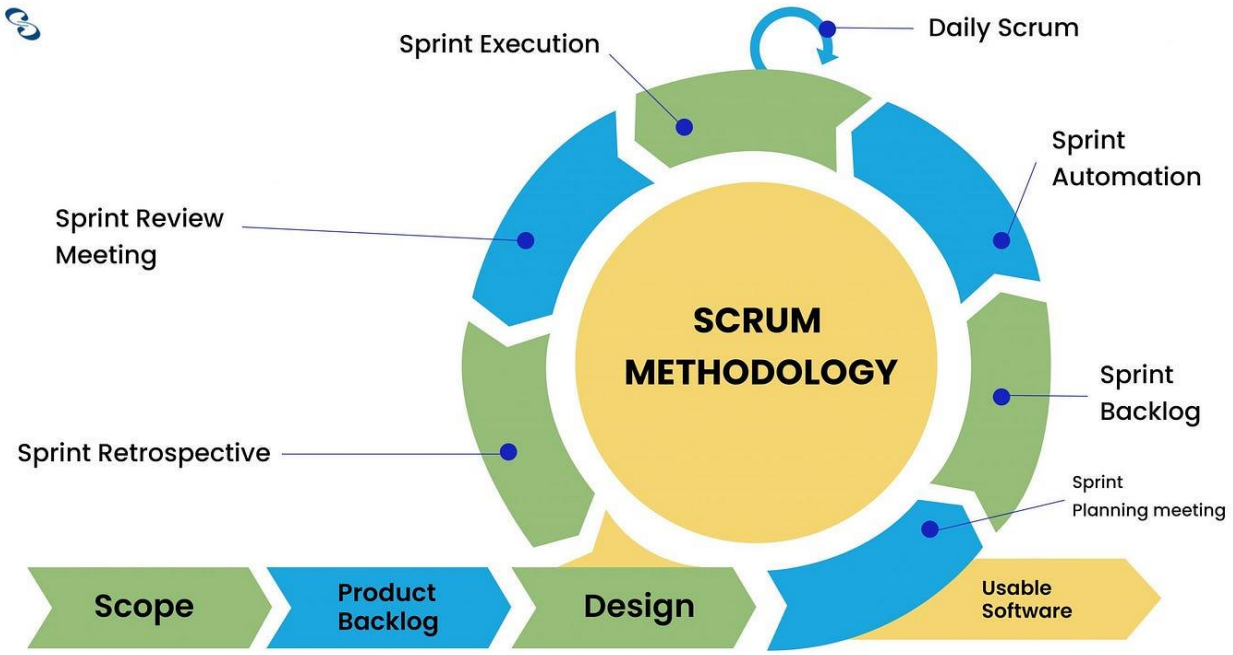

Рисунок 1.6 Методологія Scrum

4. Lean Game Development (рис. 1.7):

– характеристики: Методологія, яка наголошує на мінімізації витрат і максимізації цінності для гравців, шляхом розвитку продукту на основі зворотного зв'язку від користувачів;

– використання: Відповідає на сучасні вимоги швидкості і вимог до унікальності гри, вимагаючи постійної взаємодії зі споживачами.

Рисунок 1.7 Методологія Lean Game Development

Інструментарій для розробки ігор:

Інтегровані середовища розробки (IDE). Популярність: Використовуйте інструмент розробки games – рушії, такі як Unity, Unreal Engine, або Godot, які надають інтерфейс для створення графічних ігор та обробки різних аспектів гри, таких як фізика, анімація, звук і штучний інтелект.

Мови програмування і скриптування:

– C# для Unity: Популярна мова програмування для розробки ігор у Unity;

– C++ для Unreal Engine: Мова програмування для розробки ігор у Unreal Engine;

– GDScript для Godot: Спеціалізована мова програмування для розробки у Godot.

Графічні редактори:

– Adobe Photoshop або GIMP: Для створення текстур, спрайтів і інших візуальних елементів гри;

– Blender або Autodesk Maya: Для створення 3D-моделей і анімацій.

Звукові редактори і бібліотеки звуків:

– Audacity або Adobe Audition: Для обробки і створення звуків;

– FMOD або Wwise: Для інтеграції звукових ефектів і музики в гру.

Тестувальні інструменти: Unity Test Framework або Unreal Engine Testing Framework: Для автоматизації тестування гри і перевірки належності.

Керування версіями і співпраця. Git і GitHub або GitLab: Для керування версіями і спільної роботи над кодом і ресурсами гри.

Маркетинг і дистрибуція:

– Steam або itch.io: Платформи для дистрибуції готових ігор;

– Google Analytics або Unity Analytics: Для відстеження використання ігор і аналізу аудиторії.

Створення комп'ютерних ігор вимагає використання різноманітних методологій та інструментів, які допомагають розробникам досягати їхніх цілей і створювати захоплюючі і готові до випуску продукти. Кожен етап розробки вимагає уваги до деталей і співпраці всіх членів команди, щоб забезпечити успішність проєкту і задоволення користувачів.

Створення комп'ютерних ігор включає кілька ключових етапів (рис. 1.8) розробки, які допомагають структурувати процес і забезпечити якісний результат. Наведемо загальні етапи розробки ігор.

Концептуалізація і проєктування. На цьому етапі команда розробників визначає основну ідею і концепцію гри. Важливо визначити цільову аудиторію, жанр гри, основні механіки та цілі, які гра має досягнути. Основні кроки включають:

– створення концепції: Визначення основних характеристик гри, її механік, сюжету і візуального стилю;

– аналіз конкурентів: Вивчення і аналіз існуючих ігор в жанрі, визначення їхніх сильних і слабких сторін для уникнення повторень і для підвищення конкурентоспроможності;

– створення геймдизайн-документа (GDD): Документ, що описує всі аспекти гри, включаючи ігрову механіку, сценарій, візуальний стиль, звуковий дизайн і т.д.

Прототипування. На цьому етапі створюються прототипи гри для перевірки і валідації її основних механік і концепції. Прототипи можуть бути простими і не мають повноцінної графіки чи звукового супроводу, але дозволяють перевірити ідеї на практиці.

Прототипування геймплею: Створення простих прототипів для тестування ігрових механік і взаємодії гравця з ігровим світом.

Прототипування інтерфейсу: Розробка прототипів інтерфейсу користувача для оцінки зручності і ефективності.

Розробка. Цей етап включає створення великого обсягу контенту, програмування і інтеграцію різних аспектів гри.

Графічний дизайн і анімація: Створення текстур, моделей персонажів, об'єктів та анімації для візуального втілення гри.

Програмування ігрової логіки: Реалізація ігрових механік, штучного інтелекту, управління інтерфейсом, фізики і т.д.

Звуковий дизайн і музика: Створення звукових ефектів, діалогів, музики та їх інтеграція в гру.

Рисунок 1.8 Етапи створення комп'ютерних ігор

Тестування і відладка. Після завершення розробки гра проходить кілька етапів тестування для виявлення помилок і поліпшення якості продукту.

Тестування ігрових механік: Валідація правильності функціонування всіх ігрових елементів (рис. 1.9) і механік.

Тестування на злам і стабільність: Перевірка на проблеми з продуктивністю, стабільність інтерфейсу та інші технічні аспекти.

Випуск і підтримка. Після успішного завершення тестування гра готова до випуску.

Випуск гри: Розміщення ігри на платформах для загального доступу (Steam, App Store, Google Play тощо).

Підтримка і оновлення: Після випуску продукту підтримка гри включає у себе внесення оновлень, виправлення помилок і відгуків від гравців.

Кожен етап розробки вимагає зосередження на деталях, комунікацію в команді і управління ризиками для досягнення успіху в розробці комп'ютерних ігор.

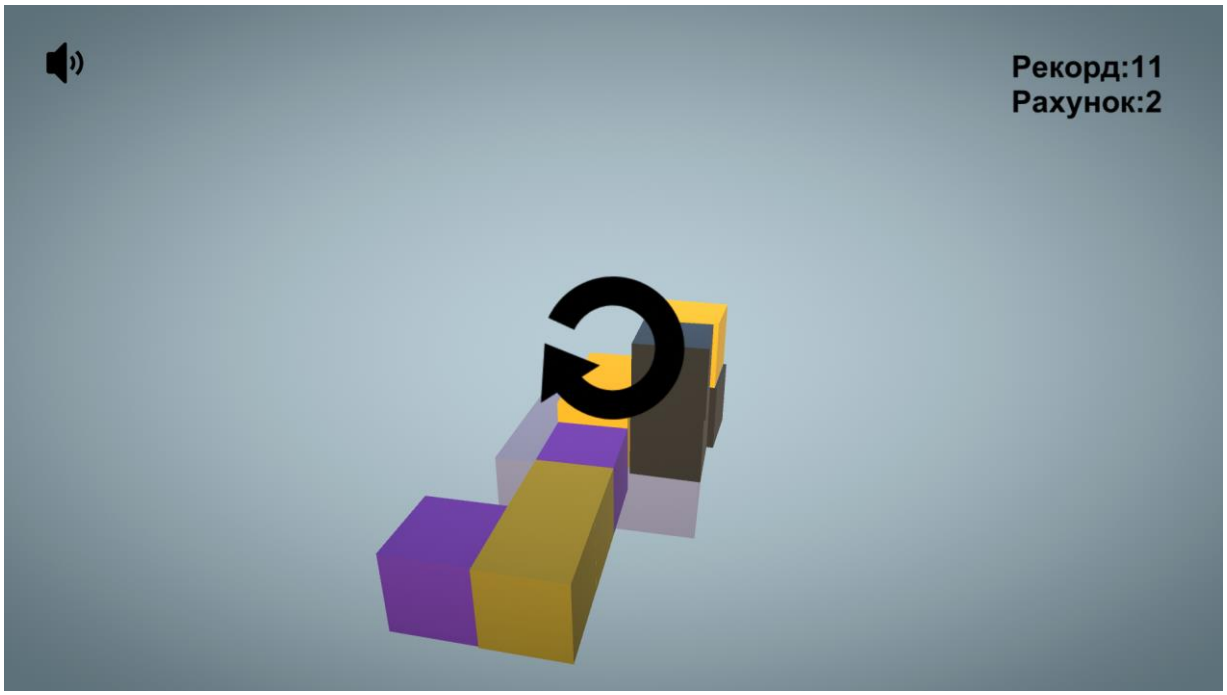

Рисунок 1.9 Тестування ігрових механік

2 СТВОРЕННЯ ГРИ «SCAM»

2.1 Алгоритм розв’язку задачі

Програма має зрозумілий інтерфейс. Управління здійснюється за допомогою натискання пальцем по сенсору або ж миші. В залежності від того де запущена сама гра.

1. Запуск гри.
2. Поява основної сцени гри, в якій знаходиться 2 кнопки: «Звук» (ввімкнути (вимкнути) звукові ефекти), «Магазин» (перегляд доступних спрайтів для ігрового процесу).
3. При натисканні на кнопку «Звук» ми будемо вимикати або вмикати звукові ефекти гри, що в ній присутні.
4. При натисканні на кнопку «Магазин» у нас відкриється нова сцена в якій ми можемо переглядати доступні нам спрайти для подальшого використання в грі.
5. При натисканні на ігровий екран у нас відразу почнеться гра.
6. Коли гравець набирає відповідну кількість очок йому відриваються нові спрайти, які супроводжуються зміною заднього фону.
7. Під час створення хмарочосу з кубиків можлива така ситуація, що одна із сторін буде важчою внаслідок чого хмарочос зруйнується. Від чого у нас з’явиться кнопка рестарту гри.

2.2 Об’єктно-орієнтований аналіз

Методологія об’єктно-орієнтованого програмування (ООА) була запропонована Йорденом для проєктування великих систем. Автор вважав, що дана методологія дозволяє більш адекватно зобразити предметну область в

системі і забезпечити більш надійний і перебудований проєкт за рахунок основних властивостей: інкапсуляція, успадкування, поліморфізм.

Об’єктно-орієнтований підхід виник в першу чергу у відповідь на зростаючу складність програмного забезпечення. Використання засобів ООП дає наступні основні переваги: зменшення складності програмного забезпечення, підвищення його надійності програмного забезпечення, забезпечення можливості модифікації окремих компонент програм без зміни решти компонент, забезпечення можливості повторного використання окремих компонентів програмного забезпечення.

Основною ідеєю об’єктно-орієнтованого підходу є об’єднання даних і дій, виконуваних над цими даними, в єдине ціле, яка називається об’єктом. Головним компонентом об’єктно-орієнтованої програми є об’єкт. І замість того, щоб розглядати програму як набір послідовно виконуваних інструкцій, в ООП програма представляється у вигляді сукупності об’єктів.

Мета даного курсового проєкту – створення власної гри в Unity.

Жанр гри «Scam» – ігра-аркада.

Граючи в ігри-аркади, гравцеві доводиться приймати миттєві рішення, покладатися на свою швидкість і реакцію. Як правило, всі аркади характеризуються великою системою бонусів. Нарахування бонусів відбувається як під час гри, так і після проходження того чи іншого рівня.

Аркади можуть бути різних напрямків. Наприклад, аркади стрілялки, або аркади гонки, також існують класичні аркади. До них відносяться такі популярні ігри як PacMan, Mario або IcyTower.

Суть гри полягає в тому, щоб набрати як найбільший рекорд та відкрити всі можливі спрайти для кубиків.

Для комфортного ведення гри потрібний яскравий та зручний інтерфейс, інтуїтивно зрозумілий користувачу.

При реалізації програми потрібно було використовувати об’єктно-орієнтований підхід, що дозволив уникнути проблем проєктування, характерних

для процедурного підходу. Важливим наслідком цього є можливість формування такої структури програми, коли додавання нових компонентів при її подальшому розвитку не буде впливати на існуючі компоненти, або таких вплив буде зведено до мінімуму.

2.3 Діаграма станів

Головне призначення діаграми станів (statechartdiagram) – описати можливі послідовності станів і переходів, які в сукупності характеризують поведінку системи. Діаграма станів представляє динамічні поведінку сутностей, на основі специфікації їхньої реакції на сприйняття деяких конкретних подій.

Діаграма станів даної гри (рис.2.1) має такий вигляд:

Рисунок 2.1 Діаграма станів

2.4 Об’єктно-орієнтоване проєктування

Методи та атрибути.

GameController – керування процесом гри:

– Start():–запускає Corotineдля функції ShowCubePlace;
– Update():–випадкова генерація точок для побудови хмарочоса;
– ShowCubePlace():– активатор перевірочного кода;

– SpawnPositions():– перевірка на можливість генерації точки;

– IsPositionEmpty(Vector3): –реалізовує генерацію випадкових точок на платформі;

– MoveCameraChangeBG(): – відповідає за переміщення камери в разі підняття хмарочосу по осі Y;

– AddPossibleCubes(int): –перевіряє зміну спрайта для кубів за набраний рекорд.

IsEnable–перевірка на відкриття спрайтів:

– Start():–перевіряє максимальний рекорд для відкриття нових спрайтів.

CameraShake–ефект тряски при руйнуванні хмарочоса камери:

– Start(): –запускає функцію трясіння камери;

– Update(): – перевіряє цілісність хмарочоса;

ExplodeCubes – ефект вибуху при програші:

OnCollisionEnter(Collision): –викликає ефект розпаду хмарочоса.

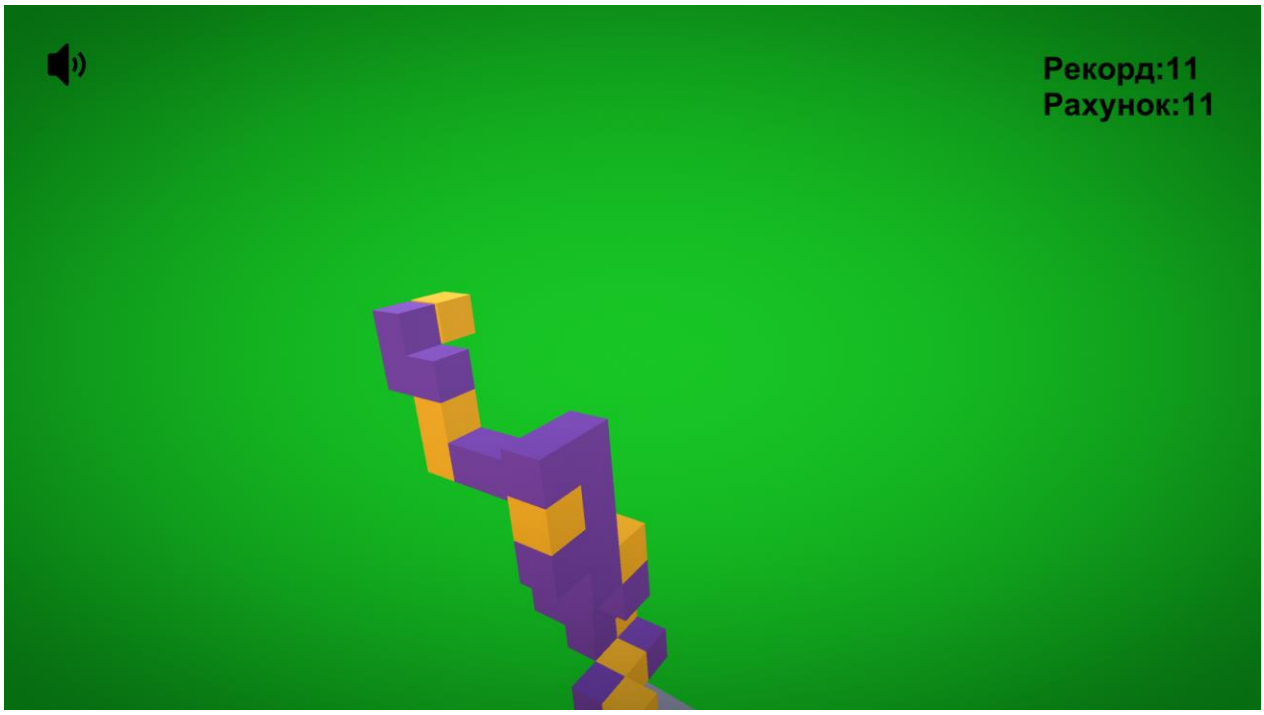

Рисунок 2.1 Момент руйнування хмарочосу

RotateCamera– обертання камери навколо платформи:

– Start():–при старті гри запускає переміщення камери навколо платформи;

– Update(): –регулює швидкість обертання камери.

Button – функціонування кнопок:

– Start(): – зберігає значення звуку;

– RestartGame(): – перезапуск ігрового процесу;

– ShopWork(): – перехід у меню магазину;

– ShopExit(): –повернення в стартове меню;

– MusicWork(): –відповідає за регулювання звуків гри.

2.5 Об’єктно-орієнтоване програмування

Першим скриптом, який я вирішив створити, був скрипт контролю процесом всієї гри. Читаючи різні джерела інформації, я знаходив різні приклади скриптів, на яких і був побудований даний код.

Скрипт«GameController»:

```
usingUnityEngine;
usingUnityEngine.UI;
using System;
usingSystem.Collections;
usingSystem.Collections.Generic;
usingUnityEngine.EventSystems;

public class GameController : MonoBehaviour
{
privateCubePosnowCube = new CubePos(0, 1, 0);
public float cubeChangePlaceSpeed = 0.5f;
```

```
public Transform cubeToPlace;
private float camMoveToYPosition, camMoveSpeed = 2f;

public Text scoreTxt;

publicGameObject[] cubesToCreate;

publicGameObjectcubeToCreate, allCubes;
publicGameObject[] canvasStartPage;
privateRigidbodyallCubesRB;

public Color[] BGcolors;
private Color toCameraColor;

private bool IsLose, firstCube;

private List<Vector3>allCubesPositions = new List<Vector3>
   {
new Vector3 (0, 0, 0),
new Vector3 (1, 0, 0),
new Vector3 (-1, 0, 0),
new Vector3 (0, 1, 0),
new Vector3 (0, 0, 1),
new Vector3 (0, 0, -1),
new Vector3 (-1, 0, -1),
new Vector3 (1, 0, 1),
new Vector3 (1, 0, -1),
new Vector3 (-1, 0, 1),
   };
```

```
privateintprevCountMaxHorizontal;
private Transform mainCam;
privateCoroutineshowCubePlace;

private List<GameObject>posibleCubesToCreate = new List<GameObject>();
private void Start()
   {
if (PlayerPrefs.GetInt("score") < 0)
posibleCubesToCreate.Add(cubesToCreate[0]);
else if (PlayerPrefs.GetInt("score") < 5)
AddPossibleCubes(2);
else if (PlayerPrefs.GetInt("score") < 10)
AddPossibleCubes(3);
else if (PlayerPrefs.GetInt("score") < 15)
AddPossibleCubes(4);
else if (PlayerPrefs.GetInt("score") < 20)
AddPossibleCubes(5);
else if (PlayerPrefs.GetInt("score") < 25)
AddPossibleCubes(6);
else
AddPossibleCubes(7);

toCameraColor = Camera.main.backgroundColor;
mainCam = Camera.main.transform;
camMoveToYPosition = 5.14f + nowCube.y - 1f;

allCubesRB = allCubes.GetComponent<Rigidbody>();
showCubePlace = StartCoroutine(ShowCubePlace());
   }
```

```
    private void Update()
       {
    if(Input.GetMouseButtonDown(0) &&cubeToPlace != null &&allCubes != null
&& !EventSystem.current.IsPointerOverGameObject())
          {
    if(!firstCube)
             {
    firstCube = true;
    foreach (GameObjectobj in canvasStartPage)
    Destroy(obj);
             }

    GameObjectcreateCube = null;
    if (posibleCubesToCreate.Count == 1)
    createCube = posibleCubesToCreate[0];
    else
    createCube = posibleCubesToCreate[UnityEngine.Random.Range(0,
posibleCubesToCreate.Count)];

    GameObjectnewCube = Instantiate(createCube, cubeToPlace.position,
Quaternion.identity) as GameObject;

    newCube.transform.SetParent(allCubes.transform);
    nowCube.setVector(cubeToPlace.position);
    allCubesPositions.Add(nowCube.getVector());

    if (PlayerPrefs.GetString("music") != "No")
    GetComponent<AudioSource>().Play();
```

```
        allCubesRB.isKinematic = true;
        allCubesRB.isKinematic = false;

        SpawnPositions();
        MoveCameraChangeBG();
            }
        if(!IsLose&&allCubesRB.velocity.magnitude> 0.1f)
            {
        Destroy(cubeToPlace.gameObject);
        IsLose = true;
        StopCoroutine(showCubePlace);
            }
        mainCam.localPosition = Vector3.MoveTowards(mainCam.localPosition,
        new Vector3(mainCam.localPosition.x, camMoveToYPosition,
mainCam.localPosition.z), camMoveSpeed * Time.deltaTime);

        if (Camera.main.backgroundColor != toCameraColor)
        Camera.main.backgroundColor = Color.Lerp(Camera.main.backgroundColor,
toCameraColor, Time.deltaTime / 1.5f);
           }
        IEnumeratorShowCubePlace()
           {
        while(true)
            {
        SpawnPositions();
        yield return new WaitForSeconds(cubeChangePlaceSpeed);
            }
           }
        private void SpawnPositions()
```

```
    {
        List<Vector3> positions = new List<Vector3>();
    if (IsPositionEmpty(new Vector3(nowCube.x + 1, nowCube.y, nowCube.z))
    &&nowCube.x + 1 != cubeToPlace.position.x)
    positions.Add(new Vector3(nowCube.x + 1, nowCube.y, nowCube.z));
    if (IsPositionEmpty(new Vector3(nowCube.x - 1, nowCube.y, nowCube.z))
    &&nowCube.x - 1 != cubeToPlace.position.x)
    positions.Add(new Vector3(nowCube.x - 1, nowCube.y, nowCube.z));
    if (IsPositionEmpty(new Vector3(nowCube.x, nowCube.y + 1, nowCube.z))
    &&nowCube.y + 1 != cubeToPlace.position.y)
    positions.Add(new Vector3(nowCube.x, nowCube.y + 1, nowCube.z));
    if (IsPositionEmpty(new Vector3(nowCube.x, nowCube.y - 1, nowCube.z))
    &&nowCube.y - 1 != cubeToPlace.position.y)
    positions.Add(new Vector3(nowCube.x, nowCube.y - 1, nowCube.z));
    if (IsPositionEmpty(new Vector3(nowCube.x, nowCube.y, nowCube.z + 1))
    &&nowCube.z + 1 != cubeToPlace.position.z)
    positions.Add(new Vector3(nowCube.x, nowCube.y, nowCube.z + 1));
    if (IsPositionEmpty(new Vector3(nowCube.x, nowCube.y, nowCube.z - 1))
    &&nowCube.z - 1 != cubeToPlace.position.z)
    positions.Add(new Vector3(nowCube.x, nowCube.y, nowCube.z - 1));
    if (positions.Count> 1)
    cubeToPlace.position = positions[UnityEngine.Random.Range(0,
positions.Count)];
    else if (positions.Count == 0)
    IsLose = true;
    else
    cubeToPlace.position = positions[0];
    }
    private bool IsPositionEmpty(Vector3 targetPos)
```

```
    {
if (targetPos.y == 0)
return false;
foreach(Vector3 pos in allCubesPositions)
        {
if (pos.x == targetPos.x&&pos.y == targetPos.y&&pos.z == targetPos.z)
return false;
        }
return true;
    }
private void MoveCameraChangeBG()
    {
intmaxX = 0, maxY = 0, maxZ = 0, maxHor;

foreach (Vector3 pos in allCubesPositions)
        {
if (Mathf.Abs(Convert.ToInt32(pos.x)) >maxX)
maxX = Convert.ToInt32(pos.x);

if (Convert.ToInt32(pos.y) >maxY)
maxY = Convert.ToInt32(pos.y);
if (Mathf.Abs(Convert.ToInt32(pos.z)) >maxZ)
maxZ = Convert.ToInt32(pos.z);
        }
if (PlayerPrefs.GetInt("score") <maxY)
PlayerPrefs.SetInt("score", maxY);

scoreTxt.text = "Рекорд:" + PlayerPrefs.GetInt("score") + "" + "  Рахунок:" +
maxY;
```

```
camMoveToYPosition = 5.14f + nowCube.y - 1f;

maxHor = maxX>maxZ ? maxX :maxZ;
if(maxHor % 3 == 0 &&prevCountMaxHorizontal != maxHor)
        {
mainCam.localPosition -= new Vector3(0, 0, 2.5f);
prevCountMaxHorizontal = maxHor;
        }
        // пример изменения цветов, можно добавить больше вариаций
if (maxY>= 25)
toCameraColor = BGcolors[4];
else if (maxY>= 20)
toCameraColor = BGcolors[3];
else if (maxY>= 15)
toCameraColor = BGcolors[2];
else if (maxY>= 10)
toCameraColor = BGcolors[1];
else if (maxY>= 5)
toCameraColor = BGcolors[0];
    }
private void AddPossibleCubes(int till)
    {
for (inti = 0; i< till; i++)
posibleCubesToCreate.Add(cubesToCreate[i]);
    }
}
structCubePos
{
```

```
publicint x, y, z;
publicCubePos(int x, int y, int z)
    {
this.x = x;
this.y = y;
this.z = z;
    }
public Vector3 getVector()
    {
return new Vector3(x, y, z);
    }
public void setVector(Vector3 pos)
    {
        x = Convert.ToInt32(pos.x);
        y = Convert.ToInt32(pos.y);
        z = Convert.ToInt32(pos.z);
    }
}
```

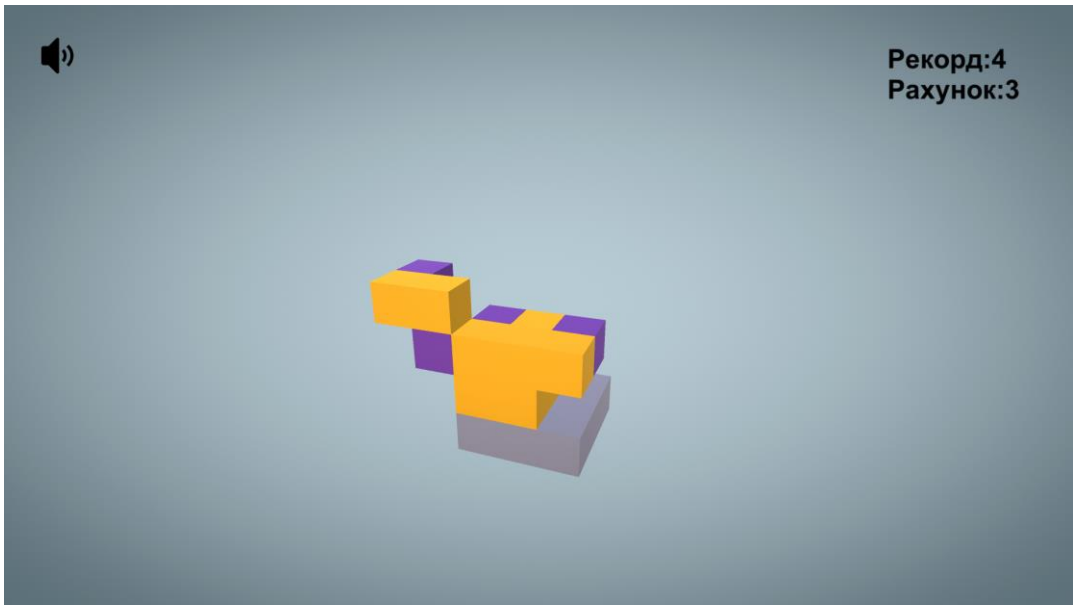

Рисунок 2.2 Ігровий процес

3 ІНСТРУКЦІЯ КОРИСТУВАЧА

Запустимо гру двічі клацнувши лівою кнопкою миші на значок гри (рис.3.1).

Рисунок 3.1 Значок гри

Чекаємо запуску гри, можемо бачити вікно запуску (рис.3.2).

Рисунок 3.2 Вікно запуску

Далі можемо бачити меню гри (рис.3.3).

Рисунок 3.3 Меню

По бажанню ми можемо вимкнути звукові ефекти, натиснувши на кнопку «Sound» (звук) (рис.3.4).

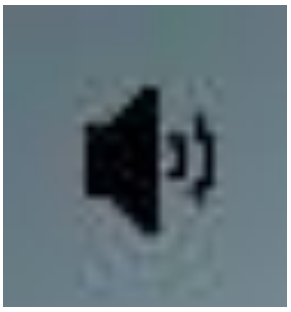

Рисунок 3.4 Кнопка «Sound»

Також ми маємо можливість відкрити внутрішньо ігровий магазин за допомогою кнопки «Shop» (магазин) (рис.3.5).

Рисунок 3.5 Кнопка «Shop»

Можемо спостерігати меню магазину (рис.3.6).

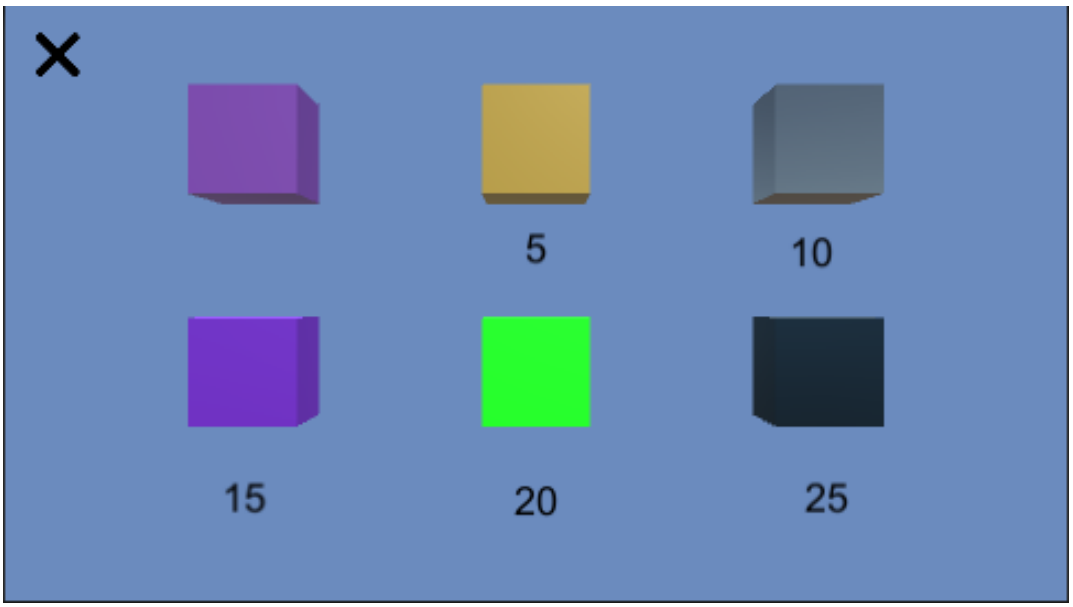

Рисунок 3.6 Магазин

Для повернення в головне меню гри, нам потрібно натиснути кнопку «Exit»(вихід)(рис.3.7).

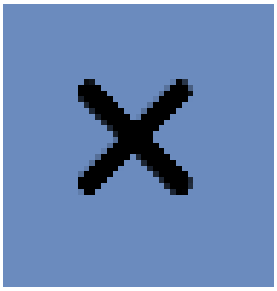

Рисунок 3.7 Кнопка «Exit»

Повернувшись в головне меню, можемо почати гру, натиснувши на екран. Після чого запуститься сама гра де ми починаємо ігровий процес (рис.3.8).

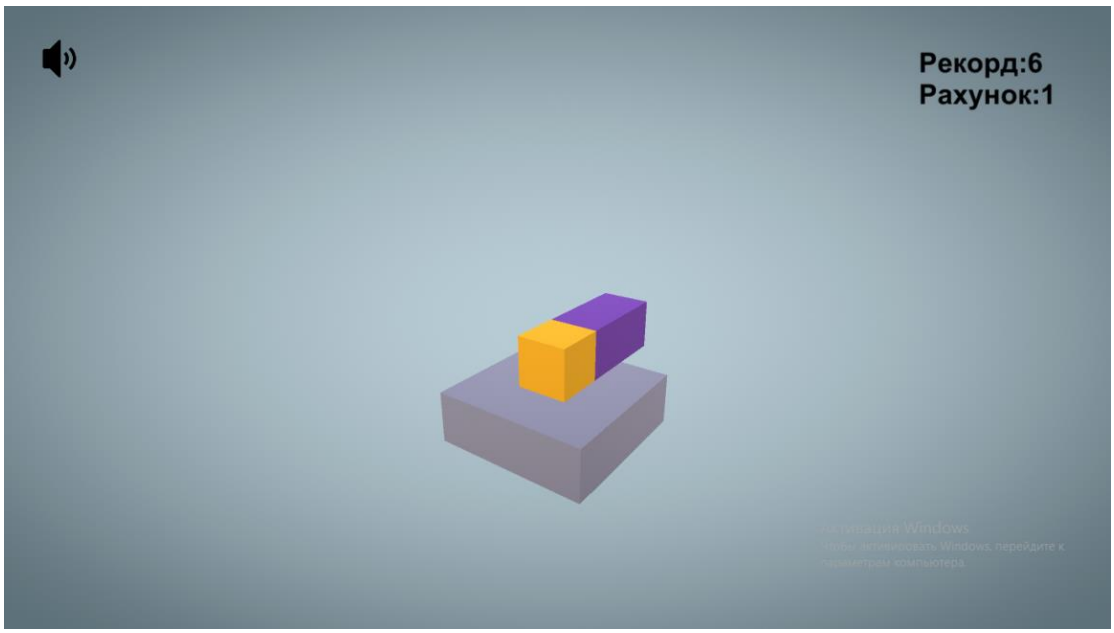

Рисунок 3.8 Ігровий процес

У випадку коли одна зі сторін хмарочоса буде переважати по вазі, хмарочос почне руйнуватись. Тобто падати на одну із переважаючих сторін (рис.3.9).

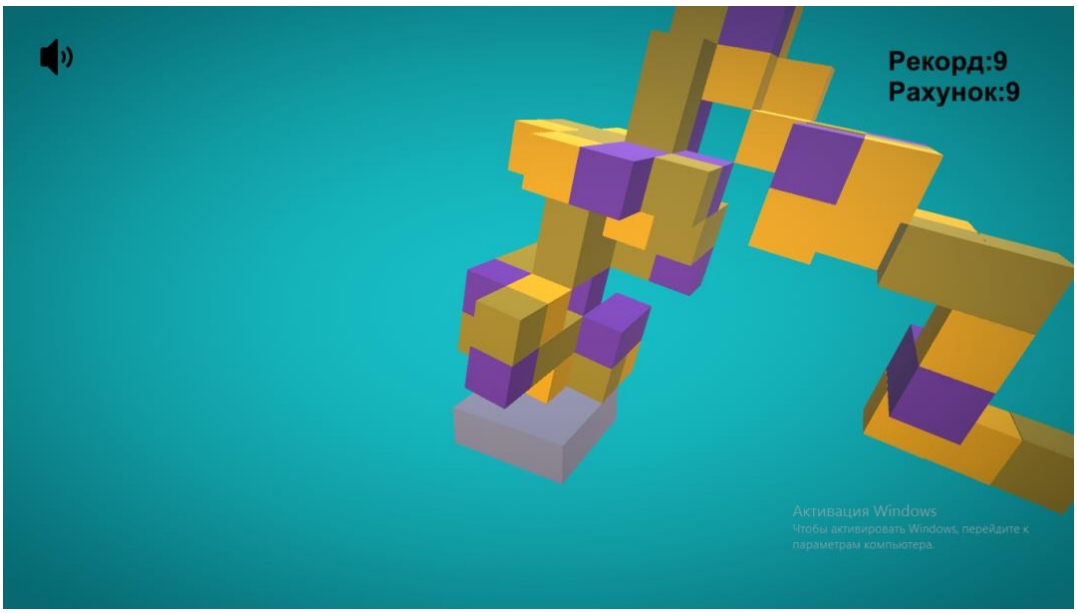

Рисунок 3.9 Руйнування хмарочосу

Коли хмарочос упаде, він розлетиться на маленькі кубики з яких був створений. Після чого з’явиться кнопка рестарту (рис.3.10).

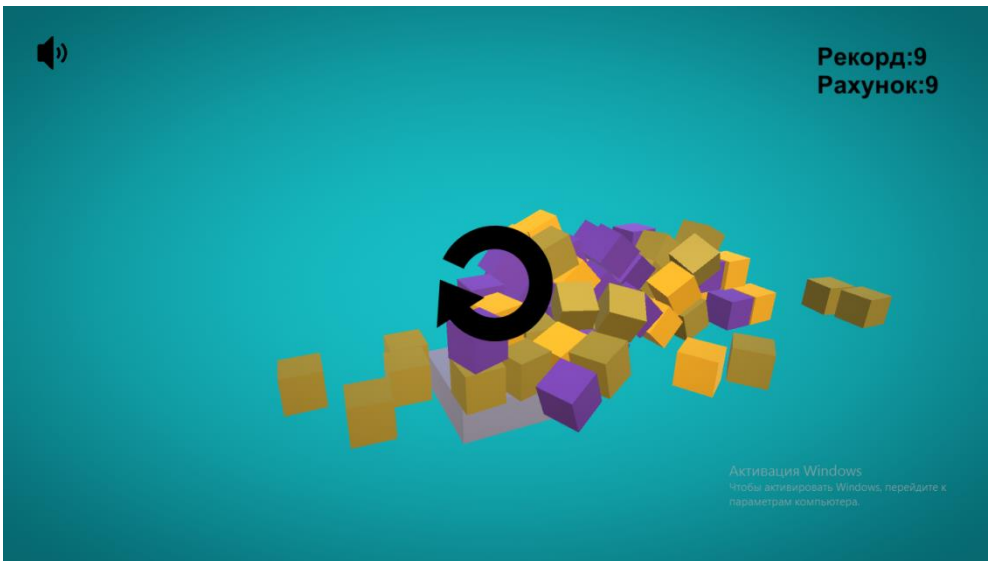

Рисунок 3.10 Кнопка рестарту

При наборі необхідної кількості очок задній фон гри буде змінюватись (рис.3.11).

Рисунок 3.11 Зміна заднього фону

4 ОХОРОНА ПРАЦІ

Організація робочого місця. Приміщення, в якому працює програміст, має загальну площу 8,37 м^2, висоту стелі 2,5 м. У приміщенні знаходиться 1 робоче місце з ПК. Робоче місце обладнане робочім столом площею 1,2 м^2, стільцем та персональним комп'ютером, що складається з монітора, системного блоку, клавіатури та миші. Слід відзначити, що площа одного робочого місця оператора ПК не повинна бути меншою за 6м^2, а об'єм не менший за 20м^3 , тобто площі та об'єму даного приміщення вистачає для розташування 1 робочого місця оператора ПК.

Аналіз умов праці показує, що у приміщенні лабораторії на програміста можуть негативно впливати наступні фізичні та психофізіологічні фактори:

– підвищена або знижена температура повітря робочої зони;

– підвищена або знижена вологість повітря;

– недостатня освітленість робочого місця;

– підвищений рівень шуму на робочому місці;

– підвищена іонізація повітря;

– підвищений рівень електромагнітних випромінювань;

– фізичні перевантаження (одноманітна поза викликає статичну втому).

Мікроклімат робочої зони програміста. Робота програміста за енерговитратами відноситься до категорії легких робіт Іа, Іб, тому повинні дотримуватися наступні вимоги (згідно ДСН 3.3.6.042-99):

– оптимальна температура повітря – 22°С (допустима – 20-24°С);

– оптимальна відносна вологість – 40-60% (допустима – не більш 75%);

– швидкість руху повітря не більш 0,1 м/с.

Виміряні за допомогою приладів (психрометр Августа) температура та вологість у лабораторії відповідають вказаним у таблиці для теплого періоду року.

Освітлення робочого місця. Нормованим параметром природного освітлення є коефіцієнт природного освітлення. КПО встановлюється в залежності від розряду виконуваних зорових робіт. Робота програміста відноситься до робіт середньої точності (IV розряд зорових робіт, мінімальний розмір об'єкту розрізнення складає 0,5-1,0мм), для яких при використанні бокового освітлення КПО=1,5%. Для штучного освітлення нормованим параметром виступає Емін – мінімальний рівень освітленості, та Кп – коефіцієнт пульсації світлового потоку, який не повинний бути більшим ніж 20%.

Мінімальна освітленість встановлюється в залежності від розряду виконуваних зорових робіт. Для IV розряду зорових робіт вона складає 300-500 лк.

Перевірка освітленості робочого місця програміста в лабораторії на кафедрі ЕОМ на відповідність розряду зорової роботи

Для штучного освітлення у приміщенні використовуються люмінесцентні лампи, які в порівнянні з лампами розжарювання мають ряд істотних переваг: за спектральним складом світла вони близькі до природного світла; мають підвищену світлову віддачу (у 2-5 разів вищу, ніж у ламп розжарювання); мають триваліший термін служби (до 10 тис. годин).

Розрахунок штучного освітлення проведемо для кімнати площею 8,37 м^2, ширина якої складає 2,7 м, довжина – 3,1 м, висота – 2,5 м за методом коефіцієнта використання світлового потоку.

Для визначення потрібної кількості світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою

$$F = \frac{ESKZ}{n}$$

F – світловий потік, що розраховується;

E – нормована мінімальна освітленість;

S – площа освітлюваного приміщення;

Z – відношення середньої освітленості до мінімальної;

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку К = 1,5);

n – коефіцієнт використання світлового потоку.

Таблиця 1 Оцінка параметрів робочого місця нормам

Фактор виробничого середовища	Санітарно-гігієнічні норми	Існуючі показники	Висновок
Температура	19-25ºС	20 ºС	В нормі
Відносна вологість	40-60%	-	-
Освітленість робочої поверхні	Не менше 300Лк	-	-
Відстань від очей до монітора	600-700мм	620	В нормі
Висота столу	680-720	730	Зависокий
Ширина столу	900	600	Замалий
Довжина столу	1600	1200	Замалий
Висота стільця	400-500	460	В нормі
Розміри сидіння стільця	380-400	386	В нормі
Площа на одного працівника	6м^2 (мінімально 4,5м^2)	8,37 м^2	В нормі

Виробничі випромінювання. Допустимі значення параметрів неіонізуючих електромагнітних випромінювань від монітору комп'ютера. Нормованим параметром невикористаного рентгенівського випромінювання виступає потужність експозиційної дози. На відстані 5 см від поверхні екрану монітору її рівень не повинен перевищувати 100 мкР/год. Максимальний рівень

рентгенівського випромінювання на робочому місці програміста зазвичай не перевищує 20 мкР/год.

Допустимі значення параметрів неіонізуючих електромагнітних випромінювань:

– напруженість електричної складової електромагнітного поля на відстані 50 см від поверхні відеомонітора: 10 В/м;

– напруженість магнітної складової електромагнітного поля на відстані 50 см від поверхні відеомонітора: 0,3 А/м;

– напруженість електростатичного поля не повинна перевищувати для дорослих користувачів: 20кВ/м.

На відстані 5-10 см від екрана і корпуса монітора рівні напруженості можуть досягати 140 В/м по електричній складовій, що значно перевищує допустимі значення.

Розрахунок необхідної потужності кондиціонеру для даного приміщення.

$S = a \times b = 2{,}7 \times 3{,}1 = 8{,}37$ м^2

$Q_З = S \times h \times q = 8{,}37 \times 2{,}5 \times 35 = 732{,}375$ [i]кВт

$Q_О = 0{,}3 \times p + 1 \times Q_{ок} = 0{,}3 \times 1053 + 0{,}3 = 316{,}2$ кВт

$Q_Р = n_ч \times q_ч + n_ж \times q_ж = 1 \times 0{,}1 = 0{,}1$ кВт

$Q_х = Q_З + Q_О + Q_Р = 732{,}375 + 316{,}2 + 0{,}1 = 1\,048{,}675$ Вт

Отже для достатнього кондиціонування приміщення підходить кондиціонер моделі MYSTERY MTH07CT-W3N2 потужністю 2,3кВт.

Розрахунок кількості лампочок для даного приміщення:

$Ф_л = E \times S \times n = 300 \times 8{,}37 \times 2{,}5 = 6277{,}5$ Лм

6277,5/900=6,975=< 7

Отже, приміщення повинно мати 7 лампочок по 10Вт кожна

Для пожежної безпеки даного офісного приміщення достатньо мати 1 вогнегасник ВВПА-500.

Людина може зазнати поразки електричним струмом як високої так і низької напруги і в деяких випадках ця поразка може бути смертельною. Небезпечні струми як високої так і низької напруги. Зокрема в навчальній лабораторії небезпечними є напруги, що перевищують 65В.

Всі прилади та експериментальні установки повинні бути побудовані таким чином, щоб запобігти можливості доторкання працюючих до деталей, по яких протікає електричний струм або які знаходяться під напругою. Необхідно слідкувати, щоб під час роботи не було відключено заземлення або обірвано провід заземлення. Перед початком роботи необхідно перевірити наявність заземлення робочої установки.

Заборонено:

– включати рубильники без дозволу спеціаліста;

– подавати напругу на робочий прилад, схему без попередньої перевірки та дозволу лаборанта;

– проводити переключення та зміну лічильників без виключення напруги;

– залишати без нагляду експериментальну установку під напругою;

– працювати на незаземленому обладнанні;

– запобігання аварійних ситуацій та ліквідація їх наслідків.

Аварійні ситуації можуть виникнути при слідуючих умовах:

– пошкодження контуру заземлення;

– пошкодження ізоляції проводів під напругою, особливо проводів, що знаходяться під напругою більше 300В;

– попадання людини під дію електричного струму;

– виникнення пожежі.

ВИСНОВОК

Підводячи підсумки роботи над проєктом, можна впевнено сказати, що ігри - одна з найбільш розвинених сфер у індустрії розваг з великим потенціалом розвитку.

Отже, виконання допомогло засвоїти теоретичні знання об'єктно-орієнтованого програмування, а також дало можливість набути практичних навичок та досвіду у створенні ігрових проєктів. В ході роботи продовжив вивчення інструментів та можливостей рушія Unity. При розборці об‘єктно-орієнтованої моделі комп'ютерної гри «SCAM» зіткнувся з технічними проблемами, які потрібно було вирішувати, та питаннями особистого планування часу та дій з розробки всіх аспектів гри.

Безкоштовна версія Unity не має всієї повноти функціоналу, ймовірно тому при збереженні неодноразово пошкоджувався проєкт, доводилося все починати з початку.

Чітке формулювання вимог дали змогу розробити об‘єктно-орієнтовану модель комп'ютерної гри «SCAM». Є деякі погрішності, їх треба допрацювати та виправити існуючі недоліки.

Аналіз проведений під час роботи, показав, що актуальність ігрових додатків буде з часом тільки зростати. Так, як людство ще з доісторичних часів полюбляє грати, а сучасна людина є частиною інформаційного суспільства, ще і має можливість отримати доступ до високоякісних ігрових платформ та ігор з високим ступенем деталізації.

Отже, у ході виконання дипломного проєкту було створено комп’ютерну гру «Scam» в ігровому движку Unity мовою програмування C#.

Для комфортного використання гри було спроєктоване зручне керування ігровим процесом за допомогою миші та зрозумілий інтерфейс.

В майбутньому у своїй грі я хочу покращити графіку, додати спец-ефекти та можливість користуватись грою не лише на комп’ютері, але й на смартфоні.

Під час виконання дипломного проєкту були одержані навички написання кодів мовою C# та роботи в Unity.

При розробці програми всі поставлені мною задачі були виконані.

Розробка виявилась дуже захоплюючим процесом. Під час роботи над своєю грою мені вдалось ознайомитись та активно попрацювати з набором чудових технологій, а також закріпити теоретичні знання з об’єктно-орієнтованого програмування. Також дипломний проєкт надав змогу розв’язати реальну задачу, реалізувати алгоритм розробки комп’ютерної гри, закріпити практичні навички розробки в програмному середовищі, створити програмне забезпечення з використанням об’єктно-орієнтованого підходу, навчитися правильно розподіляти робочий час, необхідний для розробки як самої гри, так і супровідної документації.

Ціль роботи полягала у фаховому застосуванні принципів об’єктно-орієнтованого програмування і була виконана.

ЛІТЕРАТУРА

1) Unity у дії. Мультиплатформна технологія на C#. – К.: Вітер, 2018. – 608 с.

2) Паласіос, Хорхе Unity 5.x. Програмування штучного інтелекту в іграх/Хорхе Паласіос. – К.: ДМК Прес, 2016. – 849 с.

3) Р. В. Пилипчук, Р. Г. Кацалап. Комп’ютерна гра // Енциклопедія Сучасної України : енциклопедія [електронна версія] / ред.: І. М. Дзюба, А. І. Жуковський, М. Г. Железняк та ін.; НАН України, НТШ. Київ: Інститут енциклопедичних досліджень НАН України, 2014. Т. 14. URL: https://esu.com.ua/article-4393

4) Розробка ігор на Unity 2018 за 24 години / Майк Гейг; [переклад з англійської М. А. Райтмана]. - К: Моно, 2020. - 464 с. - (Світовий комп'ютерний бестселер. Геймдізайн).

5) Фурсова Н.А. Особливості розробки мережевої комп'ютерної гри в жанрі Multiplayer First-Person Shooter / Н.А. Фурсова, О.Є. Козак // Наука і виробництво: ДВНЗ «ПДТУ». – Маріуполь – 2019.

6) Sue Blackman “Beginning 3DGame Development with Unity” 2011;

7) Silvergames – Режим доступу: URL: https://www.silvergames.com/uk/t/tetris

8) Вікіпедія. Вільна енциклопедія – Режим доступу: URL: https://ua.wikipedia.org

9) CraftPix – Збірник різноманітних ассетів – 3D, 2D, аудіо, UI.
URL: https://craftpix.net

10) Itch.io – бібліотека ігор та ассетів: 3D-моделі, об'єкти, 2D-спрайти, піксель-арт та текстури. – Режим доступу: URL: https://itch.io

11) Sketchfab – Режим доступу: URL: https://sketchfab.com/tags