МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ РОЗШИРЕНИХ МОЖЛИВОСТЕЙ ГРИ “JUNKOSMOS” У СЕРЕДОВИЩІ GAMEMAKER»**

Виконав: студент 4 курсу, групи 6.1211-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Т.Д. Бікулов
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, PhD Столярова А.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп’ютерних наук, доцент, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Бікулову Тимуру Даміровичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Проєктування та реалізація розширених можливостей гри “Junkosmos” у середовищі GameMaker

керівник роботи Столярова Анастасія Валеріївна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 02.06.2025 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз та обґрунтування вибору середовища розробки проєкту.
2. Проєктування розширених можливостей гри “Junkosmos”.
3. Програмна реалізація розширених можливостей гри “Junkosmos”.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 26.12.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	13.01.2025	
2.	Збір вихідних даних.	24.01.2025	
3.	Обробка методичних та теоретичних	20.02.2025	
	джерел.		
4.	Розробка першого та другого розділу.	11.04.2025	
5.	Розробка третього розділу.	19.05.2025	
6.	Оформлення та нормоконтроль	26.05.2025	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	16.06.2025	

Студент _____ Т.Д. Бікулов
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.В. Столярова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Проєктування та реалізація розширених можливостей гри “Junkosmos” у середовищі GameMaker»: 57 с., 35 рис., 4 табл., 19 джерел, 2 додатка.

ВІДЕОГРА, ЗАСТОСУНОК, ІГРОВА МЕХАНІКА, ІГРОВИЙ РУШІЙ, ПРОЄКТУВАННЯ, СЕРЕДОВИЩЕ РОЗРОБКИ, СЦЕНАРІЙ, GML.

Об’єкт дослідження – процес розробки комп’ютерних ігор.

Предмет дослідження – технології оновлення комп’ютерної гри “Junkosmos” засобами GameMaker.

Мета роботи: розробка та інтеграція розширених функціональних можливостей у існуючу гру “Junkosmos” з використанням середовища розробки GameMaker, що дозволить покращити ігровий досвід та розширити її потенціал.

Методи дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання, проєктування, конструювання та тестування програмного забезпечення.

У рамках кваліфікаційної роботи було розглянуто загальні характеристики особливостей середовища GameMaker, а саме: реалізований аналіз та обгрунтування вибору середовища розробки проєкту; досліджено основні характеристики ігрового рушія GameMaker; розглянуто особливості ігрового рушія GameMaker; виконаний аналіз Game Maker Language; досліджено алгоритм проєктування оновлення гри та надана характеристика функціоналу гри; виконана програмна реалізація розширених можливостей гри “Junkosmos”; проведено тестування функціональних вимог.

В результаті отримано оновлений ігровий продукт, у якому були розширені геймплейні можливості. Оновлення були розроблені засобами рушія GameMaker. В процесі оновлення було додано нові функції, та візуальні ефекти.

SUMMARY

Bachelor's qualifying paper "Design and Implementation of Extended Features for the Game "Junkosmos" in the GameMaker Environment": 57 pages, 35 figures, 4 tables, 19 references, 2 supplements.

VIDEO GAME, APPLICATION, GAME MECHANICS, GAME ENGINE, DESIGN, DEVELOPMENT ENVIRONMENT, SCRIPT, GML.

The object of the research is the process of computer game development.

The subject of the research is the technology of updating the computer game "Junkosmos" using GameMaker.

The aim of this work is to develop and integrate extended functionality into the existing game "Junkosmos" using the GameMaker development environment, in order to improve the gaming experience and expand the game's potential.

The research methods include requirements gathering and analysis, software modeling, design, implementation, and testing techniques.

As part of this qualification work, the general characteristics of the GameMaker environment were studied, namely: an analysis and justification of the choice of development environment were carried out; the key features of the GameMaker game engine were investigated; the specifics of the engine were examined; the GameMaker Language (GML) was analyzed; the algorithm for updating the game was studied and the game functionality described; the extended features of "Junkosmos" were implemented; and functional testing was performed.

As a result, an updated version of the game was produced, featuring expanded gameplay capabilities. The updates were developed using the GameMaker engine. New functionality and visual effects were added during the update process.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 7
1 Аналіз та теоретичне обґрунтування вибору середовища розробки програмного продукту 9
1.1 Огляд і порівняння середовищ розробки ігор 9
1.1.1 Аналіз середовища Unity Game Engine..... 10
1.1.2 Аналіз середовища Unreal Engine 13
1.1.3 Аналіз середовища GameMaker..... 15
1.2 Характеристика ігрового рушія GameMaker..... 18
1.3 Особливості ігрового рушія GameMaker..... 22
1.4 Аналіз Game Maker Language 24
2 Проєктування розширених можливостей гри “Junkosmos”..... 28
2.1 Концептуальний рівень проєктування..... 29
2.2 Логічний рівень проєктування..... 31
2.3 Фізичний рівень проєктування 34
3 Програмна реалізація розширених можливостей гри “Junkosmos”..... 36
3.1 Характеристика функціоналу гри..... 36
3.2 Написання коду 36
3.3 Створення графічних елементів 42
3.4 Тестування функціональних вимог після оновлень 46
Висновки 52
Перелік посилань..... 54
Додаток А Система управління персонажем 55
Додаток Б Скрипт бою з босом..... 56

ВСТУП

Сфера розробки та оновлення відеоігор динамічно еволюціонує, висуваючи дедалі вищі вимоги до рівня реалізації проєктів. Завдяки активному впровадженню новітніх цифрових технологій розробники отримують розширені можливості для створення конкурентоспроможного продукту, здатного зацікавити широку аудиторію. Процес внесення змін і доповнень до ігрового середовища сприяє глибшому аналізу ігрової логіки, взаємодії з користувачем, способів комунікації всередині гри, візуальних і функціональних елементів.

Актуальність даного проєкту пояснюється стрімким зростанням індустрії створення відеоігор, її впливом на сучасну цифрову культуру та важливістю для соціальних процесів у глобальному контексті. Розробка відеоігор і їх модифікацій посідає провідні позиції серед напрямів IT-сфери, що підтверджує актуальність обраного вектора дослідження. Ініціатива, орієнтована на україномовних користувачів, сприяє активному розвитку вітчизняного ринку цифрових розваг та допомагає просувати українські розробки за його межами.

Game Maker – зрозумілий та простий ігровий рушій, який підтримує розробку проєктів під широкий спектр операційних систем і платформ, серед яких Windows, Mac OS X, Ubuntu, Android, iOS, Tizen, Windows Phone, Windows 8, PlayStation 3, PS4, PS Vita, Xbox One та HTML5. Порівняно з іншими рушіями, створення ігор за допомогою GameMaker відбувається значно швидше. Процес інсталяції та початкового налаштування не потребує глибоких технічних знань, що робить його зручним для новачків. Крім того, перенесення проєкту на інші платформи виконується автоматично, без необхідності змінювати програмний код.

Мета роботи: розробка та інтеграція розширених функціональних можливостей у існуючу гру “Junkosmos” з використанням середовища

розробки GameMaker, що дозволить покращити ігровий досвід та розширити її потенціал.

Для досягнення поставленої мети необхідно виконати низку послідовних **завдань**:

- визначити жанрову спрямованість, тип гри та цільову платформу;
- створити концепцію змін основних компонентів ігрового процесу;
- обрати відповідний інструмент розробки та ознайомитися з його можливостями;
- підготувати графічні та анімаційні ресурси, що потрібні для оновлення;
- реалізувати зміни безпосередньо в проєкті гри;
- провести тестування, виявити й усунути технічні недоліки.

Об’єкт дослідження – процес розробки комп’ютерних ігор.

Предмет дослідження – технології оновлення комп’ютерної гри “Junkosmos” засобами GameMaker.

Методи дослідження – у дослідженні застосовуються формалізовані методи збору та аналізу вимог до програмного забезпечення, методи моделювання, а також підходи до проєктування, програмної реалізації й перевірки функціональності програмних рішень.

Кваліфікаційна робота структурно складається зі вступу, трьох розділів, висновків, переліку посилань та двох додатків.

В першому розділі розглянуто загальні характеристики особливостей середовища GameMaker реалізований аналіз та обґрунтування вибору функціонального середовища розробки проєкту.

В другому розділі розкриті питання щодо проєктування оновлення гри “Junkosmos”, а саме: досліджено алгоритм проєктування розширених можливостей гри та надана характеристика функціоналу гри.

В третьому розділі виконана програмна реалізація зазначених вище функцій. Також в результаті тестування гри було здійснено виправлення та усунення виявлених багів та технічних недоліків.

1 АНАЛІЗ ТА ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ ВИБОРУ СЕРЕДОВИЩА ПРОГРАМНОГО ПРОДУКТУ

1.1 Огляд і порівняння середовищ розробки ігор

На сучасному етапі розвитку цифрових технологій створення власних доповнень і модифікацій до комп’ютерних ігор відіграє вагому роль у сфері інформаційних технологій. Для успішного втілення подібного програмного проєкту необхідно підібрати таке програмне середовище, яке забезпечує розробника достатнім набором функціональних можливостей для реалізації ігрових рішень різної складності.

З урахуванням того, що розширення функціоналу ігор потребує використання відповідного програмного інструментарію, а також зважаючи на велику кількість доступних платформ, важливо здійснити обґрунтований вибір середовища, що відповідатиме вимогам до ефективності та якості майбутнього продукту. Серед найвідоміших програмних систем для створення ігрового контенту, які активно використовуються в професійному середовищі, можна виділити такі як Unity**,** Unreal Engine та GameMaker Studio. Вони різняться за технічними можливостями, підтримкою графічних бібліотек, типами ігор, які можна реалізувати, а також платформами, для яких призначено кінцевий продукт [5].

Задля визначення оптимального інструментарію для реалізації розширеного функціоналу гри “Junkosmos” доцільним є проведення огляду трьох провідних програмних платформ. У межах цього аналізу буде розглянуто ключові технічні особливості кожного середовища, а також їх сильні та слабкі сторони.

Отримані результати порівняльного аналізу дозволять здійснити обґрунтований вибір найбільш придатного програмного засобу для подальшої розробки.

1.1.1 Аналіз середовища Unity Game Engine

Серед широкого спектра інструментів для створення і модернізації комп'ютерних ігор особливе місце посідає рушій Unity, який здобув популярність завдяки своїй гнучкості та функціональності. Це середовище включає потужну інтегровану платформу розробки з вбудованим графічним інтерфейсом, що дозволяє працювати як із дво-, так і з тривимірними сценами. Unity забезпечує інструменти для створення моделей, налаштування фізичних властивостей об'єктів, розробки анімаційних ефектів і реалізації складної логіки взаємодії у віртуальному середовищі (див. рис. 1.1).

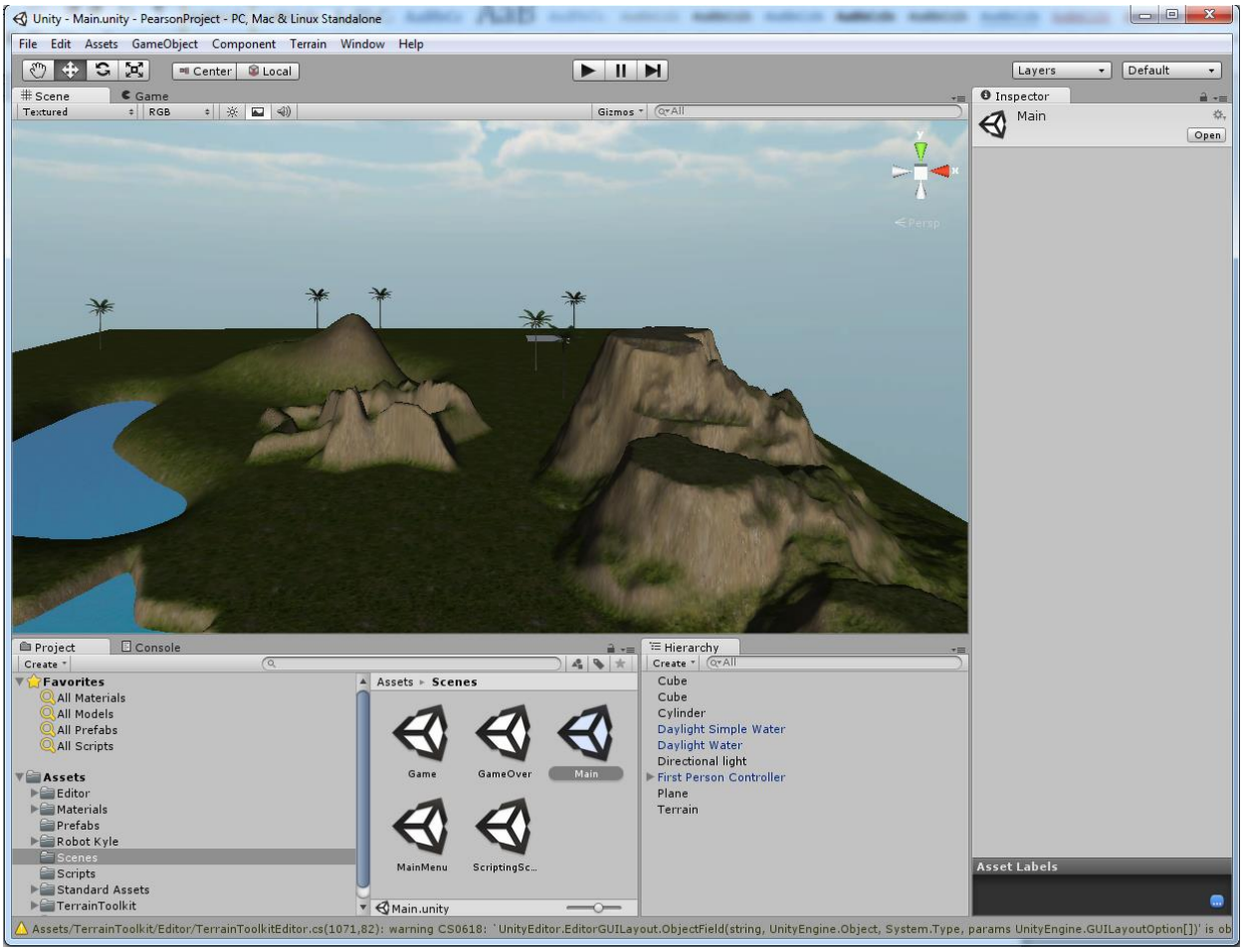


Рисунок 1.1 – Інтерфейс ПЗ "Unity"

На основі проведеного аналізу фахової літератури можна виокремити кілька ключових особливостей, притаманних ігровому рушію Unity. По-перше, цей інструментальний комплекс надає розробникам широкі

можливості для створення візуально насичених і технічно складних ігор, зокрема завдяки підтримці розширених візуальних ефектів і симуляції фізичних процесів. По-друге, розроблені за допомогою Unity проєкти характеризуються високим рівнем кросплатформеності: серед підтримуваних систем – Windows, macOS, Android, iOS, а також консолі Xbox і PlayStation. По-третє, середовище містить розвинуту екосистему допоміжних інструментів, шаблонів та бібліотек, що значно спрощує процес розробки, тестування та подальшого оновлення ігор [14].

Серед ключових переваг, які надає середовище Unity у контексті розробки і модернізації ігрових проєктів, варто виділити такі аспекти:

– середовище Unity забезпечує доступ до широкого спектра інструментів і функціональних модулів, що охоплюють графічне оформлення, фізичне моделювання тіл, звук, компоненти штучного інтелекту, а також інші засоби, необхідні для повноцінного оновлення ігрового контенту;
– широка спільнота користувачів та розробників, що активно працює з Unity, створює сприятливе середовище для обміну досвідом, вирішення технічних питань і отримання підтримки на всіх етапах розробки власних проєктів;
– однією з ключових переваг середовища Unity є його здатність легко адаптувати проєкти для різних операційних систем і платформ без суттєвої модифікації коду;
– середовище Unity інтегрується з сучасними інформаційними технологіями, включно з різноманітними бібліотеками для реалізації штучного інтелекту та функціоналу доповненої реальності, що значно розширює можливості створення інноваційного ігрового продукту [14].

Попри численні переваги, програмне середовище Unity має і низку обмежень, які варто враховувати при виборі інструментарію для реалізації ігрових проєктів. По-перше, для користувачів, які не мають попереднього досвіду у сфері геймдизайну, засвоєння функціоналу середовища Unity може

викликати складнощі, оскільки вивчення його особливостей вимагає певних базових знань з програмування та логіки побудови ігор. По-друге, використання середовища Unity у комерційних цілях часто потребує придбання ліцензійних пакетів або додаткових модулів, що в свою чергу може впливати на вартість реалізації проєкту. По-третє, розширення функціональності гри обмежується можливостями офіційної екосистеми Unity, тому розробники значною мірою залежать від темпів оновлень, політики підтримки та доступних розширень з боку розробників рушія. Крім того, варто зазначити, що під час створення ігор у 2D-просторі можуть виникати технічні труднощі, оскільки відповідна реалізація у середовищі Unity значною мірою базується на 3D-архітектурі, що іноді призводить до неточностей графічних елементів [14].

Варто особливо підкреслити, що архітектура Unity побудована на принципах компонентного підходу. У цьому підході кожен елемент ігрового середовища представлений як об’єкт, до якого можна додавати різноманітні компоненти, що визначають його функціональність, зовнішній вигляд і логіку взаємодії. Для реалізації певних функцій допускається використання готових компонентів із доступних бібліотек. Робота з об’єктами та сценами здійснюється за допомогою візуального редактора, проте повна реалізація потенціалу середовища передбачає володіння мовою програмування C#, яка є основним інструментом для створення скриптів і реалізації складної логіки в середовищі Unity.

Таким чином, Unity є сучасним ігровим рушієм, що надає широкі можливості для реалізації професійних проєктів зі створення та модернізації комп’ютерних ігор. Його функціональність, багатоплатформність і підтримка актуальних технологій роблять його привабливим середовищем для розробників. Водночас варто враховувати, що ефективне опанування Unity потребує певного рівня підготовки, зокрема знання мови програмування C# та загальних принципів розробки ігор, що може ускладнити процес освоєння для початківців. Крім того, для реалізації комерційних проєктів можуть знадобитися платні ліцензії або додаткові модулі. Ще одним обмеженням є

залежність від внутрішніх механізмів та інструментів Unity, що в окремих випадках може обмежити реалізацію нестандартних рішень.

1.1.2 Аналіз середовища Unreal Engine

Unreal Engine є ще одним сучасним програмним засобом для розробки та оновлення ігор.

Серед основних характеристик середовища Unreal Engine необхідно особливо зазначити ключові позиції. Насамперед, рушій Unreal Engine забезпечує високу якість візуалізації, включаючи реалістичне освітлення та фізичну модель, що створює можливість для побудови високодеталізованих графічних сцен. Окрім того, середовище Unreal Engine має інтегровану підтримку технологій віртуальної реальності, що дозволяє розробляти ігрові продукти з орієнтацією на VR-пристрої. Також середовище Unreal Engine надає широкий набір функціональних інструментів, програмних компонентів і вбудованих бібліотек, що полегшують створення складних внутрішніх процесів гри, зокрема механік на основі фізики, елементів штучного інтелекту, а також опрацювання звуку та візуальних ефектів [15].

До основних переваг використання середовища Unreal Engine для розробки та оновлення ігор можна зарахувати наступні аспекти:

- основний принцип функціонування Unreal Engine є висока швидкість роботи рушія;
- Unreal Engine забезпечує наявність високоякісної графіки та візуальних ефектів, що робить середовище самої гри більш реалістичним;
- значна популярність рушія Unreal Engine серед фахівців сприяє формуванню потужної спільноти користувачів і широкого доступу до навчальних матеріалів, технічної документації та консультаційної підтримки [15].

Крім того, необхідно особливо зазначити, що Unreal Engine забезпечує розробника повноцінним інтегрованим середовищем для створення ігор, до складу якого входять потужний візуальний редактор сцен, зрозумілий модуль налаштування матеріалів, а також набір засобів для моделювання тривимірних об’єктів і розробки анімації. Більш того, рушій підтримує розробку логіки проєкту як за допомогою мови програмування C++, так і за допомогою візуальної системи Blueprint. Платформа також надає можливість розгортання проєкту на широкому спектрі цільових пристроїв і операційних систем (див. рис. 1.2).

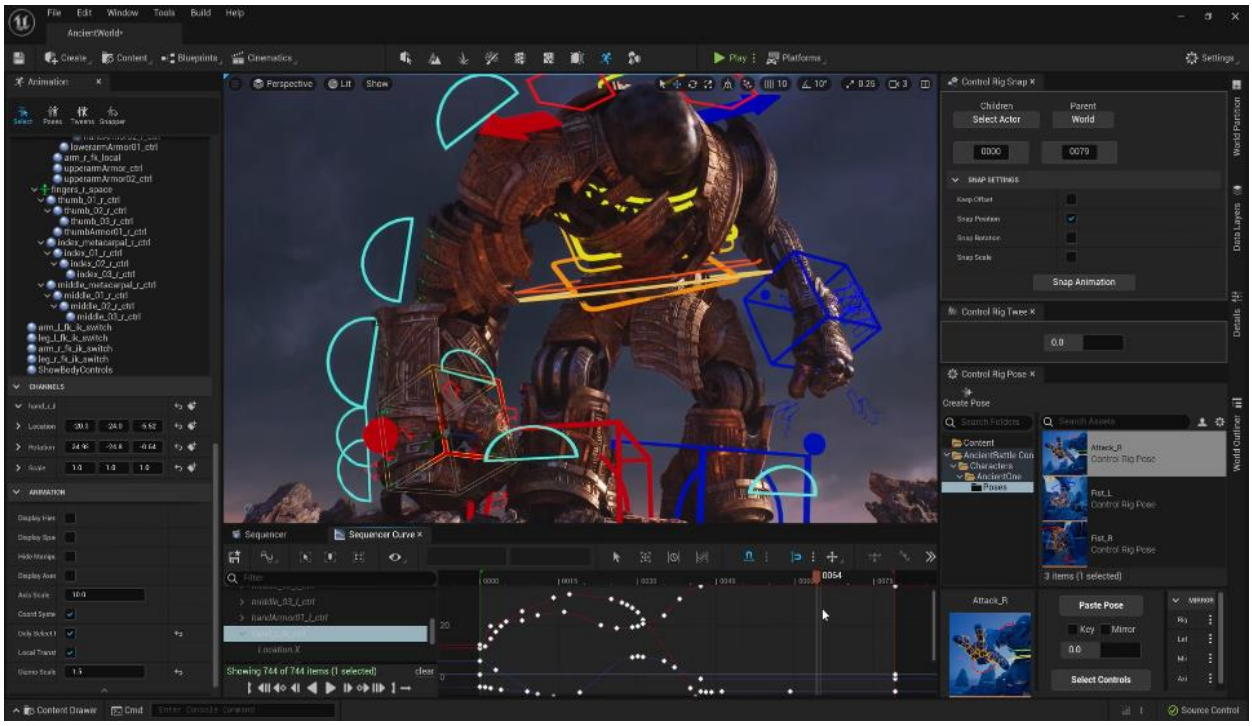


Рисунок 1.2 – Інтерфейс ПЗ Unreal Engine

До недоліків, притаманних середовищу Unreal Engine, слід віднести низку аспектів. По-перше, дана платформа має досить складну архітектуру, що ускладнює її опанування початківцями без попереднього досвіду у сфері ігрової розробки. По-друге, проєкти, створені на базі Unreal Engine, особливо з використанням складної графіки та візуальних ефектів, можуть потребувати значних апаратних ресурсів для коректної розробки, налагодження та тестування, що вимагає відповідного технічного забезпечення [16].

Таким чином, Unreal Engine можна охарактеризувати як високопродуктивне середовище, яке широко використовується для розробки та модернізації комп'ютерних ігор. Рушій ґрунтується на сучасних технологіях рендерингу, що забезпечують реалістичне зображення, підтримує моделювання складної фізики, високоточну анімацію персонажів, а також інтеграцію модулів штучного інтелекту для реалізації поведінкових сценаріїв. Архітектура рушія дозволяє ефективно використовувати ресурси сучасних графічних процесорів. Проте, при впровадженні Unreal Engine у практичну діяльність слід враховувати його високі вимоги до апаратного забезпечення, а також відносну складність у засвоєнні, що може бути суттєвою перешкодою для початківців без попереднього досвіду у сфері ігрової розробки.

1.1.3 Аналіз середовища GameMaker

Середовище GameMaker виступає одним із популярних інструментів, орієнтованих на розробку та модернізацію двовимірних ігор. [8].

Основні характеристики GameMaker включають наступні позиції:

– по-перше, GameMaker забезпечений інтуїтивним візуальним редактором, який дозволяє розробляти або оновлювати гру без необхідності програмування;
– по-друге, GameMaker надає користувачам можливість використовувати власні скрипти та розширення, що характеризується більшою ефективністю при створенні механік та різноманітних функцій самої гри;
– по-третє, ігри, розроблені з допомогою GameMaker, можуть бути адаптовані до різних платформ, наприклад, Windows, macOS, iOS, Android та інші [8].

Виходячи із аналізу літературних джерел, серед переваг середовища GameMaker необхідно особливо зазначити наступні [8]:

- середовище GameMaker є простим у вивченні та освоєнні новачками, які не мають попереднього досвіду в розробці ігор;
- програмне середовище GameMaker містить велику кількість вбудованих модулів, які дозволяють реалізовувати ключові функціональні елементи гри без потреби у використанні складного коду;
- середовище GameMaker забезпечує високий темп розробки, що дозволяє створювати базові версії ігор або концептуальні прототипи з мінімальними витратами часу, що є особливо актуальним на стадії перевірки ідеї та первинного тестування ігрової механіки [8].

Серед додаткових переваг програмного середовища GameMaker варто виокремити кілька важливих аспектів. По-перше, платформа підтримує підключення сторонніх бібліотек та модулів, у тому числі реалізованих різними мовами програмування, що розширює її функціональні можливості. По-друге, система ліцензування GameMaker є досить доступною: базова версія (наприклад, Standard) надається безкоштовно, що робить її привабливою для початківців. По-третє, середовище має інтеграцію з системами контролю версій, що спрощує командну роботу над проєктами.

Також слід відзначити простоту мови GML (Game Maker Language), яка оптимізована для розробки ігрової логіки. Нарешті, GameMaker підтримує розгортання застосунків на популярних платформах, включаючи Steam, Google Play та App Store. GameMaker Studio надає користувачам повноцінне інтегроване середовище розробки, що включає зручний графічний редактор для створення ігрових сцен, об’єктів та анімацій. Крім того, програмне забезпечення оснащене широким набором інструментів для конфігурування фізичних параметрів, звукового супроводу та інших ігрових елементів (див. рис. 1.3).

Аналіз літературних джерел дозволяє виокремити наступні недоліки середовища GameMaker для розробки та оновлення ігор [8]:

- середовище GameMaker спеціалізується на розробці 2D-проєктів, тому для реалізації масштабних 3D-ігор або складних візуальних

ефектів доводиться використовувати сторонні рішення або зовнішні двигуни;

– у GameMaker є власна скриптова мова GML, яка дозволяє глибше налаштовувати логіку гри, проте її продуктивність часто поступається більш потужним мовам, таким як C++ чи C# [8].

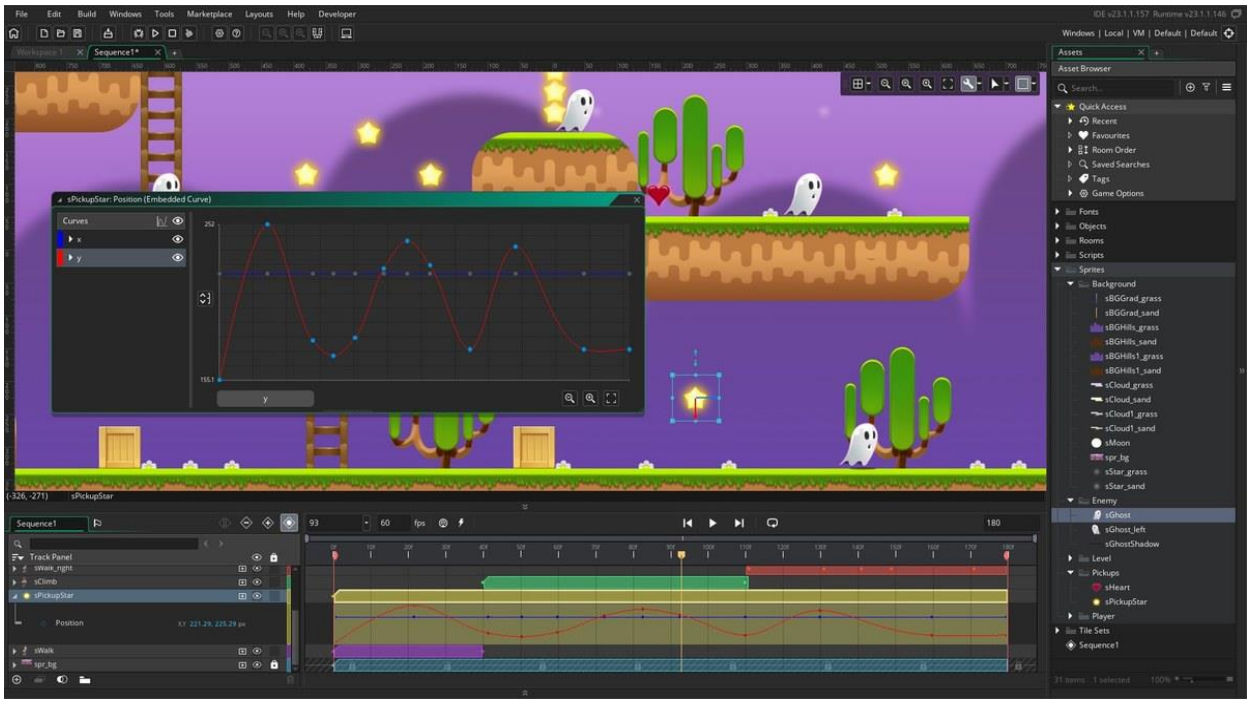


Рисунок 1.3 – Інтерфейс ПЗ GameMaker Studio

На виконання мети кваліфікаційної роботи було проведено аналіз сучасних середовищ розробки ігор, в результаті чого основним інструментом було обрано платформу GameMaker. Дане рішення ґрунтується на вдалому поєднанні зручності, функціональності та адаптації середовища до потреб користувачів з різним досвідом. По-перше, важливими є доступність та адаптивність середовища, інтуїтивний інтерфейс дозволяє використовувати рушій як новачкам, так і досвідченим розробникам. По-друге, у середовищі є продуктивна кросплатформенна підтримка, це означає, що є можливість експорту ігор на різні операційні системи та пристрої без значних змін у коді. По-третє, платформа має сучасний технічний стек, оскільки остання версія рушія була повністю переписана, що забезпечило підвищену продуктивність та стабільність.

Окрім того, рушій надає інтегровані редактори для роботи з графікою, анімацією та шейдерами. Хоча основним способом розробки є візуальне конструювання, існує можливість використовувати мову GML для більш гнучкого програмування.

Проте важливо враховувати, що GameMaker орієнтований переважно на створення 2D-ігор, оскільки реалізація 3D-проєктів вимагає додаткових ресурсів або інтеграції з іншими технологіями.

1.2 Характеристика ігрового рушія GameMaker

В самому загальному сенсі, ігрові рушії – є спеціалізованими програмними комплексами, призначеними для розробки та модифікації відеоігр. Ці платформи інтегрують у собі набір технологічних рішень, які автоматизують базові процеси створення ігрового продукту. Такі системи значно спрощують роботу розробників, дозволяючи їм концентруватись на ключових аспектах проєкту – формуванні захоплюючого ігрового процесу, створенні механік та реалізації загальної концепції гри або її оновлення [12].

Сучасний ринок інструментів для створення ігор пропонує розробникам широкий спектр спеціалізованих рішень. До числа найпопулярніших платформ належать такі продукти, як Unity, Unreal Engine, GameMaker, CryEngine та інші. Кожен із сучасних рушіїв має свої переваги та обмеження, тому остаточний вибір завжди має ґрунтуватися на детальному порівнянні можливостей платформи з конкретними цілями розробки. На виконання мети кваліфікаційної роботи був обраний GameMaker, який є одним з найбільш найпопулярних ігрових рушіїв та дозволяє розробляти додатки для великої кількості платформ.

Варто підкреслити, що комплектація Master Collection пропонує користувачам виняткову зручність і інтуїтивно зрозумілий функціонал. Особливістю даного видання є те, що воно включає не лише актуальні на сьогодні компоненти, а й гарантує отримання всіх запланованих оновлень в

межах лінійки 1.x. Такий підхід значно спрощує процес розробки, оскільки дозволяє працювати з останніми технологіями без необхідності додаткових завантажень чи модифікацій (див. рис. 1.4).

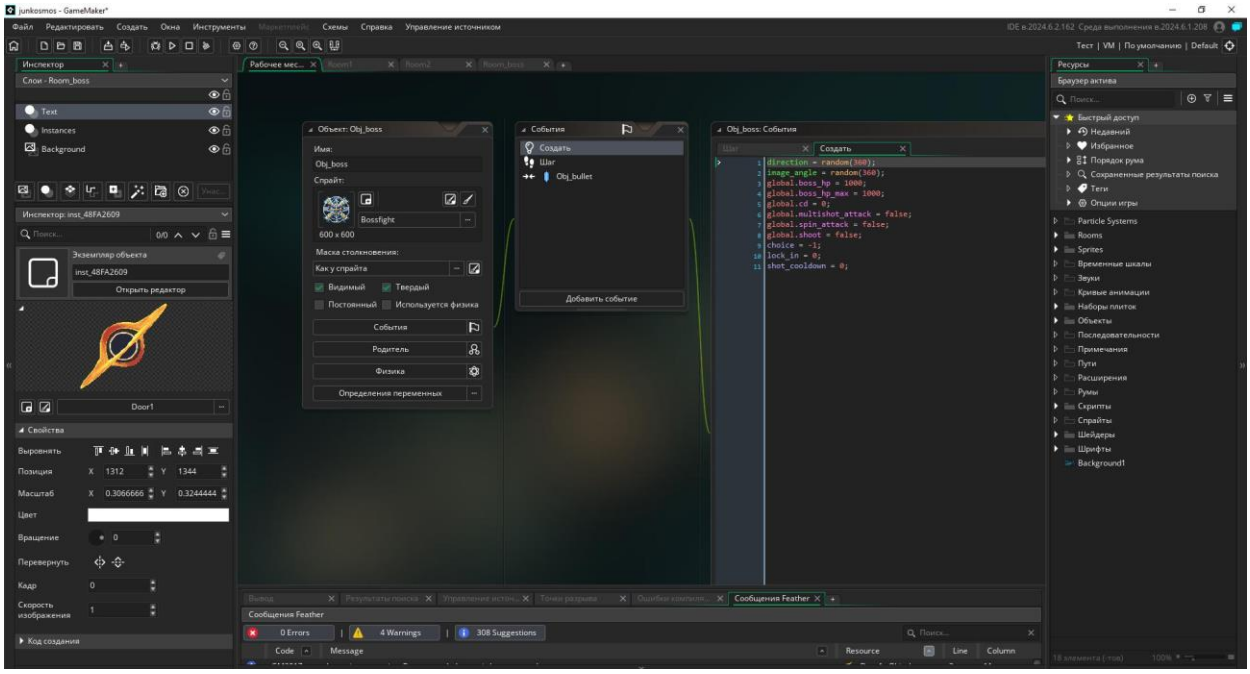


Рисунок 1.4 – Вигляд IDE

GameMaker представляє собою спеціалізоване середовище для створення двовимірних ігрових проєктів. Основою розробки в цій платформі є система взаємодіючих об’єктів, де кожен елемент виконує чітко визначену функцію.

Ключовими компонентами системи є: по-перше, візуальні елементи (спрайти), які відповідають за графічне представлення; по-друге, інтуїтивний інтерфейс візуального програмування; по-третє, скриптова мова GML, яка забезпечує гнучкість при створенні складних механік.

До переваг робочого процесу в середовищі GameMaker можна віднести такі аспекти, як: насамперед, візуальний редактор, який на основі блок-схем спрощує створення базової логіки; крім того, спеціалізовану мову GML, яка поєднує такі риси, як доступність для початківців і потужність для професіоналів; а також, наявність інтегрованої системи об’єктів, яка дозволяє швидко прототипувати ігрові механіки.

Особливістю даного інструментарію є оптимальний баланс між простотою освоєння і можливістю реалізації складних ігрових систем, що робить його популярним вибором серед незалежних розробників.

GameMaker пропонує розробникам інтегровану систему фізичного моделювання, що дозволяє легко застосувати правдоподібну фізику в ігровому середовищі. Серед ключових можливостей платформи варто відзначити: автоматизовану фізичну симуляцію – реалістичну поведінку об’єктів під дією сил тяжіння, імпульсів та інших фізичних явищ; окрім того, точне визначення взаємодії між ігровими елементами та наявність готових рішень для поширених фізичних ефектів від пружних ударів до складних траєкторій руху. Ці вбудовані інструменти значно спрощують процес створення динамічних ігрових сцен, дозволяючи розробникам зосередитись на творчих аспектах проєкту замість реалізації базових фізичних механік.

Робота з мультимедійним контентом становить ключовий етап розробки та модернізації ігрового продукту в GameMaker. Платформа надає розробникам потужний арсенал інструментів для обробки аудіовізуальних компонентів. Так, графічний редактор пропонує: інтуїтивні інструменти для створення та редагування спрайтів, а також, розширені можливості роботи з растровою графікою та функції оптимізації зображень для ігрових потреб. Аудіо-інструментарій, в свою чергу, включає: можливість імпорту та обробки звукових доріжок, інтегровані засоби для створення просторових звукових ефектів та нструменти синхронізації звуку з ігровими подіями. Ці спеціалізовані редактори дозволяють ефективно підготувати всі необхідні ресурси без залучення зовнішніх програм, що значно прискорює процес розробки та спрощує внесення змін у готовий продукт [9].

Важливою перевагою GameMaker є підтримка імпорту сторонніх мультимедійних ресурсів. Ця функція особливо корисна для команд розробників, оскільки дозволяє поєднувати переваги GameMaker із спеціалізованими інструментами для створення контенту, що значно розширює творчі можливості при створенні ігрового продукту.

Середовище GameMaker пропонує розробникам інтуїтивно зрозумілий інструментарій для роботи з мультимедійним контентом, який поєднує у собі дві ключові характеристики: доступність інтерфейсу та ефективність робочого процесу. При цьому, важливо розуміти, що кінцевий результат, в свою чергу, залежить від двох взаємопов'язаних факторів: технічних можливостей вбудованих редакторів та професійних навичок та творчого потенціалу розробника. Ця синергія дозволяє створювати якісний контент при мінімальних витратах часу, що є вирішальною перевагою при ітеративній розробці ігрових проєктів [6].

При виборі GameMaker як основного інструменту розробки важливо брати до уваги ряд технічних аспектів, які можуть вплинути на процес створення гри:

- виконання коду може відбуватися з меншою ефективністю порівняно з компільованими мовами;
- адаптація контенту під різні розміри екранів може вимагати додаткових зусиль;
- окремі функціональні елементи можуть мати недостатнє технічне опрацювання;
- вбудована підтримка охоплює лише основні операційні системи;
- розробка для мобільних пристроїв вимагає додаткового програмного забезпечення;
- перенесення на рідкісні платформи може бути технічно складним;
- помітне зниження швидкості при навантаженні системи;
- обмежений набір інструментів для тривимірної візуалізації;
- відсутність вбудованих механізмів паралельної обробки [6].

Однак, попри певні технічні обмеження, GameMaker залишається ефективним середовищем для розробки та оновлення ігор, особливо для початківців.

Підсумовуючи все вищезазначене необхідно підкреслити те, що даний ігровий рушій зарекомендував себе як ефективне рішення для розробників різного рівня підготовки. Його архітектура поєднує доступність для новачків

із достатньою потужністю для реалізації професійних проєктів. Інструментарій у середовищі GameMaker дозволяє значно скоротити час створення ігрових прототипів, крім того, привабливими характеристиками є підтримка експорту на різні операційні системи та пристрої та поєднання візуального редактора з потужною скриптовою мовою. Спеціалізована мова програмування GameMaker відрізняється лаконічним синтаксисом, широким набором вбудованих функцій та можливістю швидкої імплементації ігрової логіки. У разі того, що саме цей рушій був обраний для виконання мети кваліфікаційної роботи, необхідно більш детально розглянути особливості GameMaker та проаналізувати GameMaker Language.

1.3 Особливості ігрового рушія GameMaker

Серед особливостей рушія GameMaker необхідно виділити те, що саме це середовище надає можливість створення розширень для різних платформ із використанням відповідних мов програмування. Середовище підтримує низку форматів розширень, зокрема: GML, який використовується універсально; DLL для Windows, Windows Phone та Xbox One; JavaScript для HTML5; SO-бібліотеки для Linux і Tizen; DYLIB для macOS; PRX для платформи PS4; SUPRS для PSVita, а також спеціалізовані шаблони для iOS і Android [7].

Ще однією важливою властивістю середовища GameMaker є підтримка створення проксі-розширень, що дає змогу реалізовувати платформо незалежний підхід до кодування. Це дозволяє розробникам використовувати єдині ідентифікатори функцій у програмному коді, при цьому забезпечуючи виклики відповідних бібліотек для кожної конкретної платформи [7].

Крім того, середовище GameMaker надає широкі можливості для обробки даних: реалізована підтримка кодувань Base64, форматів JSON, алгоритмів хешування MD5 та SHA–1. Рушій також дозволяє здійснювати розпакування ZIP-архівів, зчитування та запис даних у форматах .ini, текстових і бінарних файлів, а також керувати файловою структурою. Окремої

уваги заслуговує розвинений функціонал для роботи з мережею, реалізовано підтримку багатокористувацької взаємодії через UDP, TCP та Bluetooth, можливість надсилання HTTP-запитів, завантаження файлів, а також інтеграція зі сторонніми сервісами, такими як Steam API та Facebook [7].

Однією з ключових особливостей рушія GameMaker є впровадження з версії 1.1.1086 підтримки шейдерів, що суттєво розширює графічні можливості платформи. Завдяки цьому інструменту розробники можуть реалізовувати різноманітні візуальні ефекти та здійснювати гнучке керування процесом відображення графіки. Середовище підтримує як вершинні, так і фрагментні шейдери, написані мовами GLSL ES, HLSL версій 9 і 11, а також GLSL, що забезпечує широкі можливості для кастомізації візуального рендерингу під різні апаратні платформи. Ця функціональність перетворює GameMaker з простого 2D-інструменту на потужне середовище для створення якісних візуальних рішень, зберігаючи при цьому основні переваги швидкої розробки [7].

Варто окремо відзначити, що середовище GameMaker забезпечує підтримку взаємодії не лише з традиційними пристроями введення, такими як миша та клавіатура, а й з додатковими інтерфейсами. Зокрема, рушій містить засоби для роботи з джойстиками та ігровими контролерами, а також реалізує функціонал для обробки сенсорних жестів і змін положення пристрою, що є критично важливим при створенні мобільних ігор. Ця система робить GameMaker особливо привабливим середовищем для створення ігор, орієнтованих на різні типи пристроїв введення, забезпечуючи при цьому високий рівень гнучкості та контролю для розробників.

GameMaker інтегрує потужний набір математичних інструментів, що охоплює: по-перше, базові операції, які передбачають насамперед скалярні обчислення з високою точністю, крім того, розширену роботу з векторами (2D/3D) та матричні перетворення; по-друге, спеціалізовані функції, які містять повний спектр тригонометричних операцій, а також логарифмічні та експоненціальні обчислення, методи інтерполяції (лінійна, білінійна, кубічна) та статистичні розподіли та генерацію випадкових чисел; по-третє,

оптимізовані алгоритми, які використовують швидкі методи нормалізації, ефективні обчислення скалярних та векторних добутків та кватерніонні операції для 3D-обертань [7].

У процесі розробки в середовищі GameMaker важливою перевагою є можливість інтеграції з популярними сервісами монетизації, зокрема AdMob, Google Analytics, Google Play Licensing, що реалізується безпосередньо через програмний код. Крім того, рушій підтримує роботу з системами контролю версій, такими як SVN, GIT та Mercurial, що сприяє організації колективної розробки та ефективному відстеженню змін у проєкті [7].

Однією з вагомих переваг сучасних версій рушія GameMaker є поява модуля налагодження The Debug Module, який відкриває нові можливості для діагностики роботи застосунку. Завдяки цьому модулю розробник може встановлювати точки зупину (breakpoints), відстежувати перебіг виконання програми, аналізувати значення змінних, стан об’єктів, вміст буферів, а також контролювати частоту та тривалість виконання окремих скриптів, що значно підвищує ефективність тестування та оптимізації гри.

Таким чином, GameMaker є потужним середовищем для розробки 2D-ігор, Підтримка шейдерів, фізичного рушія Box2D, інструментів для роботи з графікою, анімацією, файлами, мережею та монетизацією свідчить про високий рівень функціональності. Особливо цінною є наявність сучасного модуля налагодження, що дозволяє ефективно аналізувати виконання коду. Завдяки цим характеристикам GameMaker є придатним як для новачків, так і для досвідчених розробників, забезпечуючи баланс між простотою використання та гнучкістю функцій.

1.4 Аналіз Game Maker Language

Вважаємо доречним розпочати аналіз Game Maker Language із визначення поняття. Одже, відповідно до наукових джерел, Game Maker Language (GML) являє собою спеціалізовану мову програмування, розроблену

для використання у середовищі GameMaker з метою створення та модифікації комп’ютерних ігор. Спочатку GML застосовувалася як доповнення до системи подій, реалізованих за допомогою графічного інтерфейсу, однак згодом функціонал подій був інтегрований безпосередньо в саму мову. Це забезпечило розробникам можливість повністю відмовитися від використання кнопкових блоків та зосередитися на скриптовій реалізації логіки ігрового процесу [9].

На підставі аналізу наукових джерел можна зробити висновок, що мова програмування GML є органічною складовою середовища GameMaker і використовується виключно в його рамках. Слід окремо відзначити поширене хибне уявлення про те, що GML дозволяє вбудовувати фрагменти коду іншими мовами, зокрема Pascal, Assembler або C++. Насправді така думка ґрунтується на зовнішній подібності синтаксичних конструкцій GML із зазначеними мовами програмування, однак функціонально GML не підтримує інтеграцію стороннього коду [9].

Однією з характерних рис мови GML є її взаємозв’язок із системою кнопкових подій, які формують структуру бібліотек у середовищі GameMaker. У графічному інтерфейсі ці бібліотеки відображаються як тематичні вкладки з піктограмами відповідних подій, що впорядковані за функціональним принципом. До складу середовища входить низка стандартних бібліотек, які містять базові дії, поширені серед більшості ігрових проєктів. Крім того, GameMaker передбачає можливість створення користувацьких бібліотек за допомогою інструменту Library Maker. Особливо важливою характеристикою є наявність механізму взаємного перетворення між графічними діями та їхнім програмним представленням у GML, що значно спрощує процес навчання та сприяє глибшому розумінню принципів роботи ігрової логіки [1].

Однією з ефективних особливостей використання рушія GameMaker є автоматизоване управління пам’яттю в мові GML. Система динамічно виділяє пам’ять під змінні під час виконання програми, що усуває потребу в ручному керуванні цим процесом. GML використовує динамічну типізацію, яка дозволяє змінним набувати значень різних типів у межах одного виконання –

наприклад, змінна може спочатку зберігати ціле число, а згодом бути переоголошеною як така, що містить рядок. Такий підхід забезпечує гнучкість і спрощує написання коду, особливо для початківців у розробці ігор [13].

Наведемо приклад, який ілюструє цю особливість динамічної типізації в GML:

intNumber = 1; // Спочатку змінна містить ціле число;

intNumber = "Ця змінна наразі вміщує строку"; // Потім те саме ім’я змінної містить рядок.

Цей приклад демонструє, що змінна intNumber може змінювати свій тип даних під час виконання коду, що є однією з ключових особливостей мови Game Maker Language (GML).

Однією з особливостей GML є відсутність інструментів для прямого звільнення пам’яті, яка була виділена під змінні. Проте, пам’ять може бути звільнена непрямим шляхом – присвоєнням змінній меншого за обсягом значення. Наприклад, якщо змінна містить великий текст, то заміна його на порожній рядок сприяє звільненню пам’яті. Аналогічним чином можна звільнити масиви [13]:

data = [1, 2, 3, 4, 5]; // створення масиву (підтримується в GMS2 і вище);

data = 0; // видалення масиву шляхом заміни його на просте число.

Варто звернути увагу на те, що при видаленні об’єкта в середовищі GameMaker автоматично знищуються всі змінні, які були локальними для нього. Водночас глобальні змінні продовжують існувати, незалежно від об’єкта, до якого вони належали. У зв’язку з цим доцільніше використовувати локальні змінні, а глобальні – лише тоді, коли це дійсно виправдано. Для роботи з великими обсягами даних GameMaker надає різноманітні структури: стеки, черги, списки, асоціативні масиви (карти), пріоритетні черги й сітки. Їх створення, редагування та видалення відбувається за допомогою відповідних вбудованих функцій. Більшість із цих структур також підтримує сортування даних. У випадках, коли потрібно зберігати довільні типи інформації, ефективним рішенням є використання буферів, які виступають у ролі блоків пам’яті для збереження різнорідних даних [13].

Таким чином, GameMaker Language (GML) є інтуїтивно зрозумілою мовою сценаріїв, яка створена спеціально для середовища GameMaker. Вона поєднує простоту використання з широкими можливостями для реалізації логіки ігрових процесів. Основною перевагою GML є її гнучкість. Мова дозволяє ефективно працювати з різними типами даних, структурами, а також автоматично керує пам’яттю без потреби у ручному звільненні ресурсів. Завдяки зручному переходу від візуального програмування до коду, GML ідеально підходить для початківців, при цьому не обмежуючи досвідчених розробників у складності реалізації. Постійна еволюція мови разом із рушієм GameMaker забезпечує її актуальність та адаптивність під сучасні вимоги індустрії розробки ігор.

2 ПРОЄКТУВАННЯ РОЗШИРЕНИХ МОЖЛИВОСТЕЙ ГРИ “JUNKOSMOS”

Головним завданням процесу проєктування інформаційної системи є створення оптимальної структури даних, яка дозволяє ефективно зберігати й обробляти інформацію. Така структура повинна враховувати всі сутності, їхні властивості та взаємозв’язки, що будуть використовуватись в межах функціонування системи та відповідати потребам різних груп користувачів. Цього можна досягти шляхом побудови моделі даних, яка має відповідати таким ключовим вимогам [19]:

- легкість сприйняття (модель має бути доступною для розуміння не лише фахівцям, а й користувачам без технічного досвіду);
- чітке структурування (модель повинна дозволяти безпомилково розрізняти сутності, типи зв’язків між ними та встановлені правила);
- відсутність надлишковості (кожен факт фіксується лише один раз, що дозволяє уникнути дублювання);
- готовність до змін (модель має легко адаптуватися до розширення функціоналу або нових вимог);
- наглядність (структура повинна мати можливість бути відображеною у вигляді зрозумілих графічних схем) [19].

Згідно з аналізом наукової літератури, процес проєктування інформаційних систем передбачає проходження трьох етапів [19]:

- концептуальне проєктування: створення концептуальної моделі, незалежної від вибору моделі та конкретної системи керування;
- логічне проєктування: створення на основі концептуальної моделі логічної моделі даних відповідно до обраної моделі організації даних, але незалежно від конкретної системи керування;
- фізичне проєктування: реалізація логічної моделі засобами конкретної системи керування [19].

2.1 Концептуальний рівень проєктування

Відповідно до аналізу наукових джерел, концептуальне проєктування бази даних являє собою процес побудови узагальненої інформаційної моделі предметної області, яка не враховує особливості конкретного програмного або апаратного середовища реалізації [19].

Концептуальне проєктування складається з наступних етапів [19]:

– ідентифікація основних типів сутностей;
– визначення типів зв’язків між сутностями;
– призначення атрибутів і встановлення їх належності до відповідних сутностей або зв’язків;
– встановлення допустимих множин значень (доменів) для кожного атрибута;
– виявлення атрибутів, які можуть бути використані як потенційні або основні ключі;
– виконання спеціалізації або узагальнення сутностей;
– формування графічного представлення моделі у вигляді діаграми типу: сутність-зв’язок;
– перевірка моделі шляхом обговорення її з представниками [19].

Загальна послідовність дій під час реалізації проєкту оновлення гри на концептуальному рівні передбачає виконання таких етапів: формулювання загальної ідеї та концепції оновлення, визначення жанрової приналежності, створення нових ігрових елементів, сценаріїв та механік, що становлять основу геймплею. Після цього здійснюється проєктування дизайну, що охоплює опрацювання візуальної складової, графічного оформлення, звукового супроводу, а також побудову зручного інтерфейсу взаємодії з користувачем [10].

Для наочного розуміння процесу проєктування розширених можливостей гри “Junkosmos” на концептуальному рівні необхідно розглянути загальну діаграму використання (див. рис. 2.1).

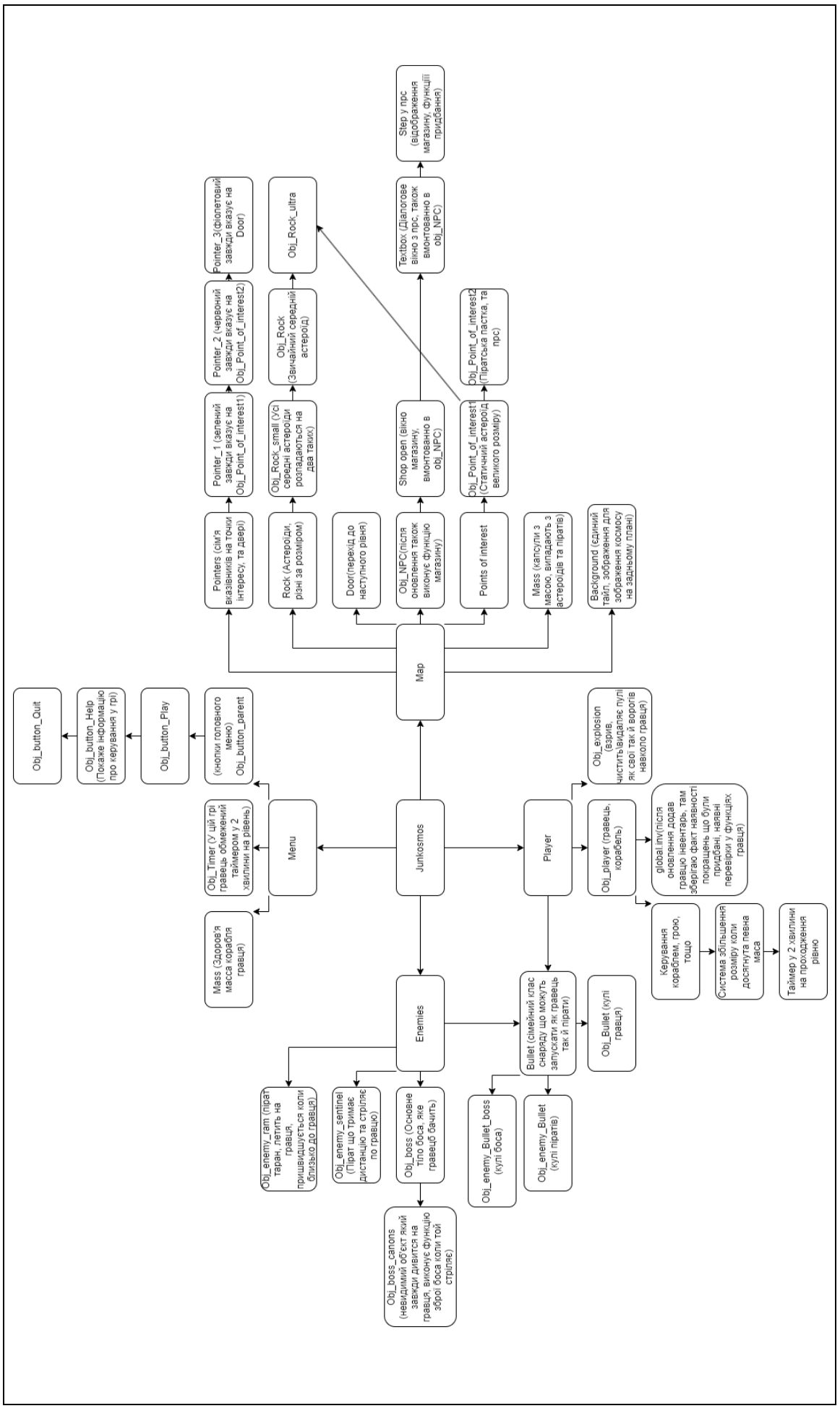


Рисунок 2.1 – Діаграма класів

Дана діаграма відображає усі існуючі об’єкти та важливі функції у їх межах. Після оновлення гри в цю діаграму були додані функції магазину для npc, та доданий інвентар для гравця.

На діаграмі зображена класова структура гри “Junkosmos”. Починаючи з розподілу на чотири групи, усі класи\об’єкти поділені на:

– “Menu” – меню, інтерфейс гри, класи пов’язані з ними;
– “Map” – карта, об’єкти на ній та об’єкти пов’язані з навігацією по ній;
– “Player” – гравець, усі об’єкти пов’язані з гравцем, та частини\функції самого об’єкту гравця;
– “Enemies” – вороги\пірати, їх однойменні об’єкти (наприклад, об’єкт Obj_boss, що є босом гри, для перемоги у грі його треба подолати).

2.2 Логічний рівень проєктування

На основі опрацювання наукових джерел можна зазначити, що логічне проєктування передбачає трансформацію концептуальної моделі у логічну структуру з урахуванням специфіки обраної системи керування базами даних. Основною метою цього етапу є побудова логічної схеми, яка буде адаптована до вимог та можливостей конкретної СКБД. Оскільки переважна більшість сучасних СКБД – реляційні, то і концептуальну модель БД слід відображати на реляційну модель. У основі реляційної моделі використовується поняття «відносини», яке використовується для уявлення набору екземплярів об’єкту (сутність) та відносин (зв’язків) між об’єктами [19].

На підставі проведеного проєктування розширених можливостей гри “Junkosmos” на логічному рівні необхідно було:

– визначити необхідну кількість відношень, яка дорівнює кількості сутностей гри;
– для кожного відношення визначити всі атрибути;
– встановити обмеження для доменів атрибутів;
– визначити тип даних для кожного атрибуту відношення;

– для кожного атрибуту визначити вимоги до підтримки цілісності даних: визначити обов’язковість наявності даних (припустимість значення NULL) [19].

Наступна діаграма детально зображує кількість сутностей та відносин між ними, а також процес логічного використання ігрових функцій для досягнення кінцевої мети ігрового контенту (див. рис. 2.2).

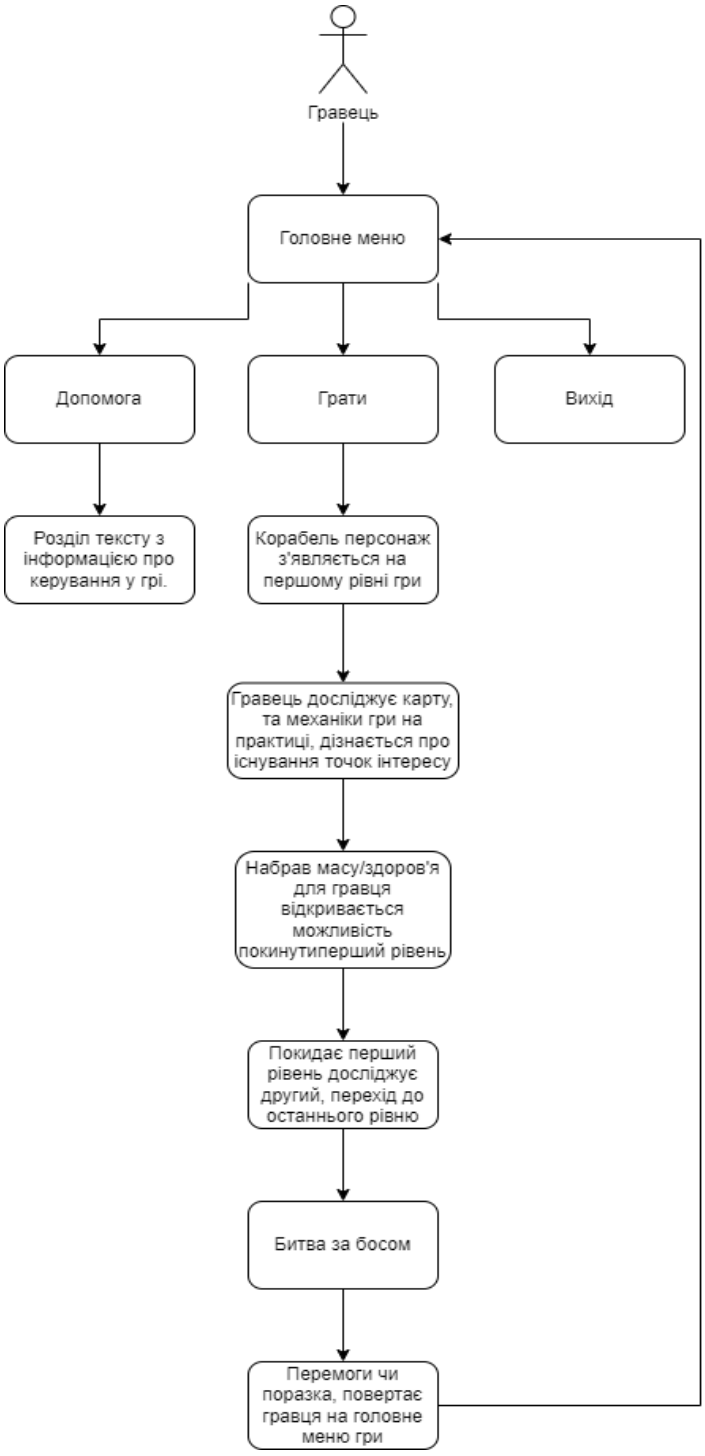


Рисунок 2.2 – Діаграма використання

За діаграмою та ідеєю, гравець відкриє гру, та побачить головне меню гри. На головному меню розташовані три кнопки: «Грати», «Допомога», «Вихід». Якщо гравець бажає почати грати одразу, то може натиснути «Грати», але якщо гравець невпевнений у своїх знаннях клавіш керування, то може натиснути «Допомога», що відповідно відобразить невелике меню під усіма кнопками, яке пояснює які клавіши можна натискати, та що вони роблять. Але якщо гравець вже втомився та хоче завершити роботу гри, то може натиснути «Вихід». Це вимкне гру.

Переходимо до початку гри. Після натиснення «Грати», гравця перенесе на перший рівень. Гравець побачить що його аватар це невеликий космічний корабель. Також бачить таймер у 120 секунд, що відповідно є часом на проходження рівню. Гравець не має певних вказівників, тому почне досліджувати зону. У процесі можна зрозуміти, що є 2 маркери, зелений та червоний. Перший вказує на точку у якій з’являється величезний астероїд. А друга до пастки з декількома ворогами. У результаті можна навчитися збивати астероїди, піратів та набрати масу\здоров’я. Що приводить до збільшення розмірів корабля, та появлення нового пурпурного маркеру. Він веде до переходу гравця на новий рівень. Там побачить що таймер оновився та знову має 120 секунд.

У грі три рівні, перший другий, та кімната з босом. На другому рівні візуально майже нічого не змінюється, тільки інше зображення космосу, а на точці пастки з піратами буде чекати торговець. Після бою можна почитати діалог та купити покращення для корабля.

Далі переходимо до кімнати з босом. Одразу після переходу можна побачити велику піратську станцію, що одразу відкриє вогонь по гравцеві. У момент коли станція переможена, гравець отримує сповіщення про перемогу та через деякий час повертається у головне меню гри. Таким є процес гри уявлений розробником, відображений на діаграмі.

Таким чином, діаграма зображує схематичне пояснення реалізації ігрових функцій саме таким чином, як це спроєктував розробник.

2.3 Фізичний рівень проєктування

Згідно з аналізом наукових джерел, на стадії фізичного проєктування відбувається пристосування реляційної структури до особливостей обраної системи керування базами даних, а також реалізація програмної частини гри відповідно до технічних вимог середовища. Для впровадження оновлень гри на фізичному рівні необхідно конкретизувати ключові елементи ігрового процесу, зокрема: способи керування персонажем, логіку фізичної взаємодії об’єктів у середовищі, механізми штучного інтелекту, а також інші функціональні компоненти, що впливають на ігровий досвід [10].

З метою удосконалення та розширення функціональності гри “Junkosmos” було реалізовано нові можливості, зокрема модуль магазину та оновлену систему ігрових покращень, доступних для користувача. До переліку реалізованих доповнень також увійшло створення нових ігрових локацій, розробка завдань та інших інтерактивних компонентів, що сприяють підвищенню інтересу гравців і загальному покращенню ігрового процесу.

Після впровадження оновлень гри було проведено тестування з метою виявлення можливих помилок, недоліків та удосконалення ігрового процесу. Процес налагодження реалізованого функціоналу спрямований на підвищення стабільності гри та оптимізацію її використання. Документація програмного забезпечення включає супровідні матеріали, що містять необхідні технічні та функціональні описи для ефективного використання програмного продукту. До таких матеріалів відносяться програмний код, а також пояснювальні та описові документи, що розкривають особливості реалізації та застосування гри.

Завершальним етапом стало оптимізування оновленого функціоналу гри з метою досягнення максимальної продуктивності та забезпечення коректної роботи на різних апаратних і програмних платформах. Особливу увагу було приділено виявленню та усуненню потенційних технічних проблем, пов’язаних із завантаженням ресурсів, швидкодією та загальною стабільністю ігрового процесу.

Таким чином, загальний порядок виконання проєкту з удосконалення гри “Junkosmos” включає послідовні етапи від формування ідеї та розробки дизайну до кодування, перевірки працездатності та підвищення продуктивності. Такий структурований підхід сприяв організованому процесу розробки та створенню якісних і функціональних оновлень для гри.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗШИРЕНИХ МОЖЛИВОСТЕЙ ГРИ “JUNKOSMOS”

3.1 Характеристика функціоналу гри

Розроблений продукт являє собою оновлення відеогри “Junkosmos” для широкого кола гравців [18]. Головною метою гри є досягнення перемоги над фінальним босом та перевірка навичок гравця через глибоке занурення у ігровий світ. Відповідно, основні вимоги до оновлення включають:

– впровадження різноманітних ігрових механік;
– використання випадкової генерації ресурсів для унікального ігрового досвіду при кожному проходженні;
– створення управління, що дозволяє керувати грою однією рукою;
– коректне відображення усіх візуальних і текстових елементів гри.

Таким чином, оновлення гри спрямоване на підвищення різноманітності геймплею та покращення зручності керування, що забезпечує більш захоплюючий і комфортний ігровий досвід. Впровадження випадкової генерації ресурсів дозволяє зробити кожне проходження унікальним, а належне відображення візуальних і текстових елементів гарантує цілісність ігрового світу. Таким чином, оновлення сприяє не лише зростанню інтересу гравців, але й підвищенню якості та функціональності гри загалом.

3.2 Написання коду

Під час розробки оновлень гри особливе значення має написання коду, яке забезпечує реалізацію основної ігрової логіки. Код відповідає за обробку подій, керування персонажами, взаємодію з ігровими об’єктами та інші важливі процеси. Використання рушія GameMaker дозволило ефективно організувати та контролювати різні компоненти оновлення. Варто

підкреслити, що саме на етапі кодування ідеї трансформуються у працездатний продукт з розширеним функціоналом. Далі розглянемо ключові складові коду, що були задіяні при впровадженні оновлень гри.

В об’єкті не ігрового персонажа (торгівця), реалізовано відображення магазину в розділі GUI (див. рис. 3.1).

```
if (!shopOpen) exit;

//drawback
draw_set_color(c_black);
draw_set_alpha(0.8);
draw_rectangle(0, 0, guiWidth, guiHeight, 0);
draw_set_alpha(0.5);
draw_rectangle(menuMargin, 0, menuMargin+menuWidth, guiHeight,
0);
draw_set_alpha(1);
draw_set_color(c_white);

//draw items
for (var i = 0; i<itemCount; i++){
	//Get Item
	var arr = items[| i];
	var item = arr[0];
	//Draw position
	var _x = menuMargin + menuWidth/2;
	var _y = (guiHeight/2) + (i-selectedAnim)*40;
	//Draw scale
	var s = 2;
	if (i==selected) s = 2.4;
	//Selected item name
	if (i==selected) item = "> " + item + " <";

	//Draw
	draw_set_halign(fa_center);
	draw_set_valign(fa_middle);
	draw_text_transformed(_x, _y, item, s, s, 0);
	draw_set_valign(fa_left);
	draw_set_halign(fa_left);
}

//Get selected item's data
var arr = items[| selected];
var item = arr[0];
var price = arr[1];
var desc = arr[2];
//Draw position
var _x = (menuMargin + menuWidth) + previewWidth/2;
var _y = (guiHeight/2);
//Draw sprite
var spr= asset_get_index("sItem_" + item);
if (sprite_exists(spr)) {
	//Draw scale
	var s = 4;
	//Draw
	draw_sprite_ext(spr, 0, _x-100, _y - 250, s, s, 0, -1,
1);
}

//Draw description
draw_set_halign(fa_center);
draw_set_valign(fa_middle);
draw_text_transformed(_x + 40, _y + 120, desc, 3, 3, 0);
draw_set_valign(fa_left);
draw_set_halign(fa_left);

//draw player's money(scrap\hp)
//draw_text_transformed(8, 8, "Scrap: " + string(price), 3, 3,
0);

//draw price
draw_set_valign(fa_bottom);
draw_text_transformed(menuMargin + menuWidth + 420, guiHeight -
400, "Price: " + string(price), 3, 3, 0);
draw_set_halign(fa_right);
draw_text_transformed(guiWidth-380, guiHeight-400, "Press Q to
buy!", 3, 3, 0);
draw_set_halign(fa_left);
draw_set_valign(fa_top);
```

Рисунок 3.1 – Відображення магазину в розділі GUI

Також при створенні об’єкту в інстансі, задаються певні параметри та список товарів (див. рис. 3.2).

```
/// @description Insert description here
shopOpen = false;
global.shop = false;
myTextbox = noone;
dialoguetext[0] = "Watch out for pirates!";
dialoguetext[1] = "Thanks for dealing with them for me.
Checkout my wares by pressing E! And be near docking range
too!";
dialoguetext[2] = "Checkout my wares by pressing E! And be
near docking range!";

selected = 0;
selectedAnim = 0;

// items
items = ds_list_create();

ds_list_add(items, ["multi_shot", 300, "Triple your
efforts!"]);
ds_list_add(items, ["outburst", 100, "Clear the playing
field by pressing SPACEBAR!"]);
ds_list_add(items, ["1_up", 400, "Another chance!"]);

itemCount = ds_list_size(items);

//GUI
guiWidth = display_get_gui_width();
guiHeight = display_get_gui_height();

menuWidth = guiWidth * 0.3;
menuMargin = guiWidth * 0.1;

previewWidth = (guiWidth - (menuWidth+menuMargin));
```

Рисунок 3.2 – Параметри та список товарів

Реалізація магазину є у функції виконаної кожний кадр (див. рис. 3.3, 3.4).

```
/// @description Insert description here
// You can write your code in this editor
if (distance_to_object(Obj_player) <= 600){
	myTextbox = instance_create_layer(x,y, "Text" ,
textbox);
	myTextbox.text = dialoguetext;
}else instance_deactivate_object(textbox);

// open shop
if (!shopOpen && place_meeting(x, y+10, Obj_player) &&
keyboard_check_pressed(ord("E"))) {
	shopOpen = true;
	global.shop = true;
}

//close shop
else if (shopOpen && keyboard_check_pressed(vk_escape)) {
	shopOpen = false;
	global.shop = false;
}
```

Рисунок 3.3 – Покадрові функції

```
//shop is open
if (shopOpen) {
	selectedAnim = lerp(selectedAnim, selected, 0.1);

	//down
	if (keyboard_check_pressed(ord("S"))) {
		selected++;
		if (selected == itemCount) selected = 0;
	}

	//up
	if (keyboard_check_pressed(ord("W"))) {
		selected--;
		if (selected < 0) selected = itemCount-1;
	}

	//buy
	var arr = items[| selected];
	var item = arr[0];
	var price = arr[1];

	if (keyboard_check_pressed(ord("Q")) &&
global.hp_mass > price) {
		//check if already bought an item
		var bought = false;
		if ds_list_empty(global.inv) == true {}
		else {
			for (var i = 0;
i<ds_list_size(global.inv); i++) {
				var arr_player = global.inv[|
i];
				//show_debug_message("got
here!")
				if (string(item) ==
string(arr_player)) {
					bought = true;

	//show_debug_message("Tested: " + string(arr_player)
+ " with " + string(item));
					break;
				}
			}
		}

		if bought == false {
			//add to inventory
			ds_list_add(global.inv, item);

			//take the price
			global.hp_mass -= price;
		}
		else {//add to inventory
			ds_list_add(global.inv, item);

			//take the price
			global.hp_mass -= price;}
	}
}
```

Рисунок 3.4 – Покадрові функції

В об’єкті гравця реалізована зупинка управління на час використання магазину (див. рис. 3.5).

```
if (global.shop) {speed = 0; exit;}
```

Рисунок 3.5 – Пауза гравця

Реалізації покращень, що покупаємо в магазині також є в об’єкті гравця (див. рис. 3.6 – 3.8).

```
if keyboard_check(vk_space)
{
	var available = false;
	if ds_list_empty(global.inv) == true {}
		else {
			for (var i = 0;
i<ds_list_size(global.inv); i++) {
				var arr_player = global.inv[| i];

				if ("outburst" ==
string(arr_player)) {
					available = true;
					break;
				}
			}
		}
	if available == false {}
	else {
		if explosion_cooldown > 0 {}
		else
		{
			instance_create_layer(x-
(120*global.exp_size), y-(120*global.exp_size),
"Instances", Obj_explosion);
			explosion_cooldown += 480;
		}
	}
}
```

Рисунок 3.6 – Очистка пуль навколо себе

```
if global.hp_mass <= 0 { // death in a fight
	var revive = false;
	if ds_list_empty(global.inv) == true {}
		else {
			for (var i = 0;
i<ds_list_size(global.inv); i++) {
				var arr_player = global.inv[| i];

				if ("1_up" == string(arr_player))
{
					ds_list_delete(global.inv,
i)
					revive = true;
					break;
				}
			}
		}
	if revive == false {
		effect_create_above(ef_explosion, x, y, 2,
c_teal);
		global.player_defeated = true;
		instance_destroy();
	}
	else {
		global.hp_mass = 200;
		global.player_defeated = false;
		flash = 0.8
		instance_create_layer(x-(120*global.exp_size),
y-(120*global.exp_size), "Instances", Obj_explosion);
	}
}
```

Рисунок 3.7 – Друга спроба, воскресіння

```
if ds_list_empty(global.inv) == true {}
else {
	for (var i = 0; i<ds_list_size(global.inv); i++) {
		var arr_player = global.inv[| i];
		if ("multi_shot" == string(arr_player)) {
			global.multishot = true;
			break;
		}
	}
}
```

Рисунок 3.8 – Потрійний постріл

Система управління персонажем представлена на рисунку 3.9. Кадри невразливості та повільне зниження швидкості руху описано на рисунку 3.10.

```
// WASD controls a
if keyboard_check(ord("W"))
{
		if speed > 5 { if speed > 8 {} else
{motion_add(image_angle, 0.1);} }
		else {motion_add(image_angle, 0.15);}
}
if keyboard_check(ord("A"))
{
	image_angle += 4;
}
if keyboard_check(ord("D"))
{
	image_angle -= 4;
}
if stage >= 2
{
	if keyboard_check(ord("S"))
	{
		if speed > 5 { if speed > 8 {} else
{motion_add(image_angle+180, 0.1);} }
		else {motion_add(image_angle+180, 0.15);}
	}
	if keyboard_check(vk_down)
	{
		if speed > 5 { if speed > 8 {} else
{motion_add(image_angle+180, 0.1);} }
		else {motion_add(image_angle+180, 0.15);}
	}
}
```

Рисунок 3.9 – Система управління персонажем

```
if invul_time > 0
{
	invulnerable = true;
	invul_time -= 1
}
else {invulnerable = false;}


if speed > 0
{
	speed -= 0.02
}
if speed < 0
{
		speed = 0;
}
```

Рисунок 3.10 – Кадри невразливості та повільне зниження швидкості руху

3.3 Створення графічних елементів

Були проведені деякі оновлення гри “Junkosmos”. А саме:

– була оновлена графіка;

– додано задній план космосу;

– перероблені текстури астероїдів, супротивників, та гравця.

До оновлення інтерфейс мав наступний вигляд, як представлено на рисунках 3.11 та 3.12.

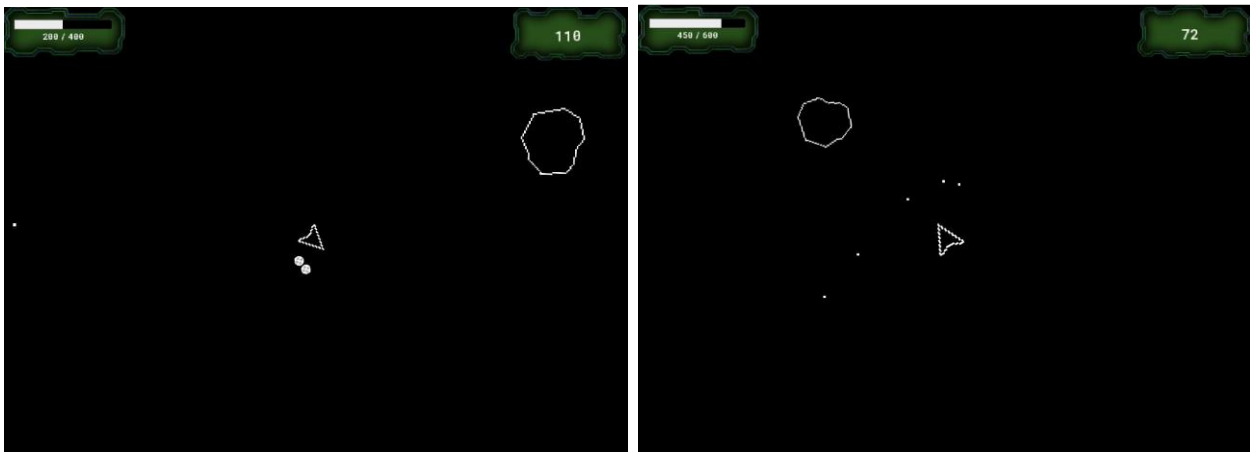


Рисунок 3.11 – Інтерфейс до оновлення



Рисунок 3.12 – Приклад вигляду супротивників до оновлення

Після оновлення інтерфейс набув вигляду, як представлено на рисунках 3.13 та 3.14.

Також для реалізації повноцінного оновлення гри необхідно реалізувати головне меню. Було додано головне меню гри, з вкладкою допомоги у якій є інформація з базового керування персонажем гравця (див. рис. 3.15).

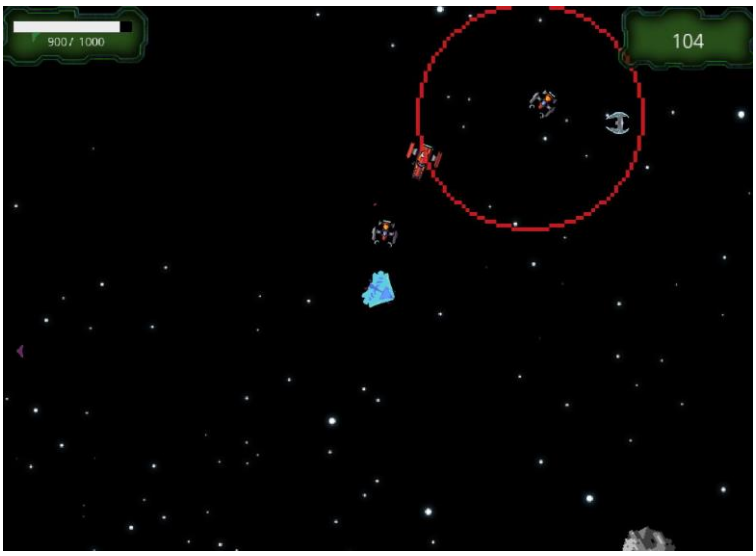


Рисунок 3.13 – Приклад оновленого вигляду супротивників

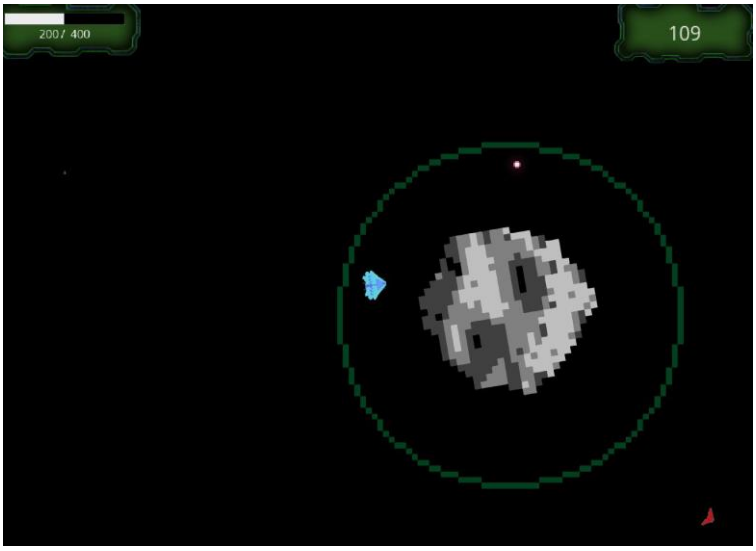


Рисунок 3.14 – Оновлений інтерфейс з появою нових точок інтересу



Рисунок 3.15 – Головне меню гри

Після натискання на кнопку «Допомога» головне меню набуло вигляду, як представлено на рисунку 3.16.



Рисунок 3.16 – Головне меню з інформацією базового керування

В останньому оновлені, було додано магазин для придбання здібностей, та перероблена система отримання здібностей гравцем. Раніше персонаж отримував здібності в залежності від розміру корабля та в залежності від накопиченого металобрухту. Приклади діалогових вікон представлено на рисунках 3.17 – 3.19.

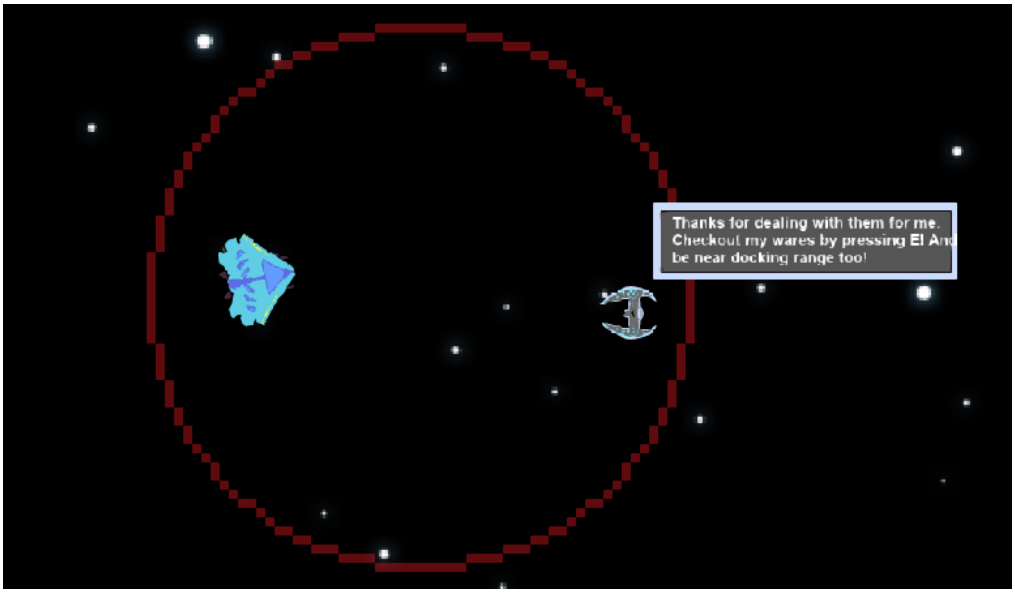


Рисунок 3.17

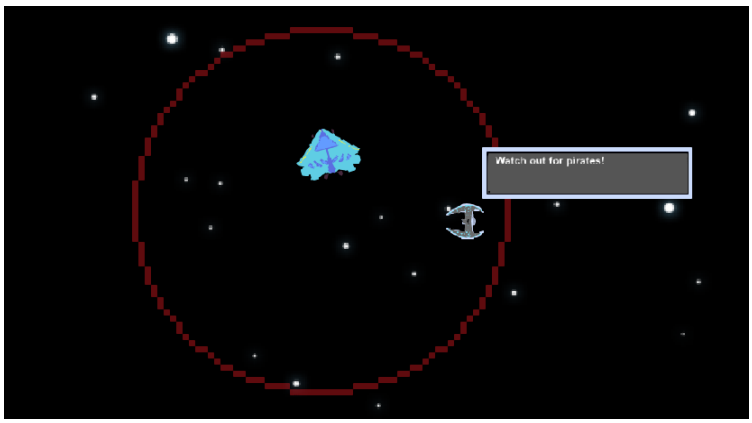


Рисунок 3.18

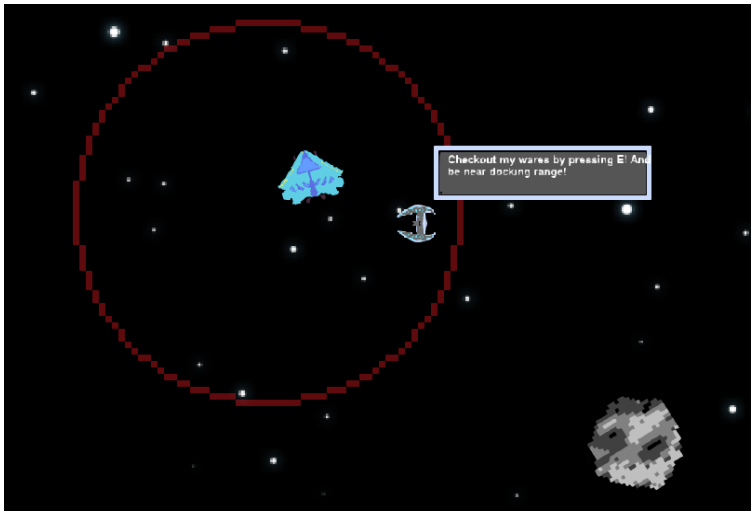


Рисунок 3.19

Після оновлення за металобрухт/здоров’я персонажа, гравець може придбати здібності в торгівця на другому рівні (див. рис. 3.20 – 3.22).



Рисунок 3.20 – Приклад меню магазина

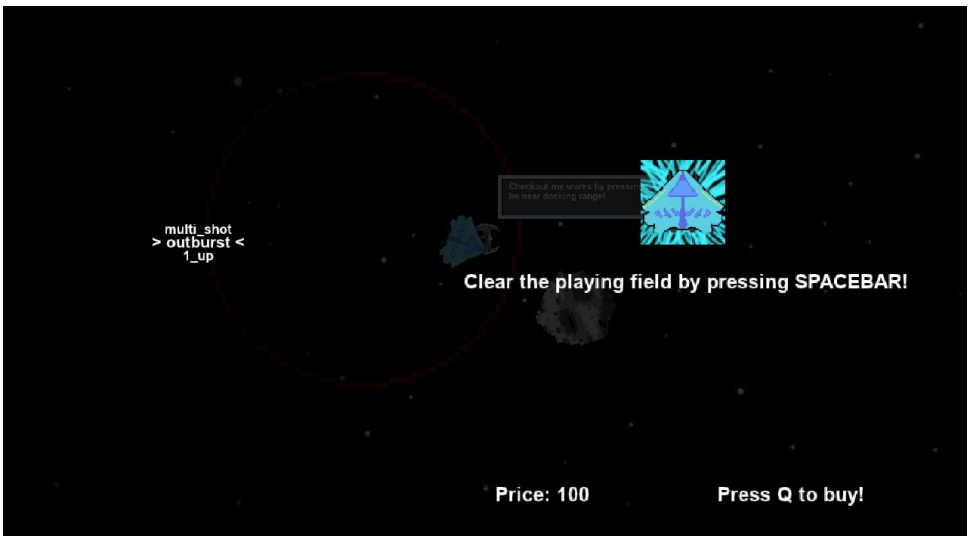


Рисунок 3.21 – Приклад здібності з магазина

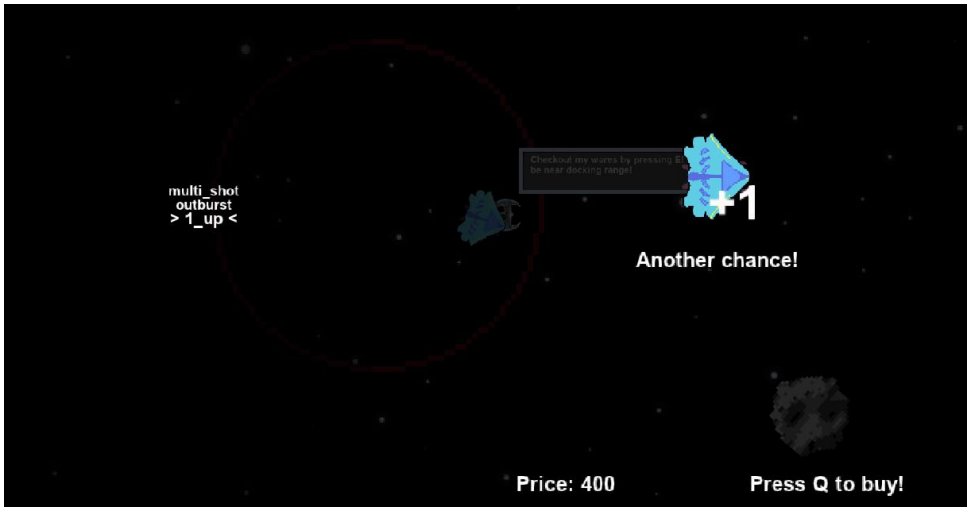


Рисунок 3.22 – Приклад оновлення

Завдяки візуальним елементам інтерфейсу меню, користувач може призупиняти гру, повторно проходити рівні, переходити до наступних етапів або повертатися до головного меню.

3.4 Тестування функціональних вимог після оновлень

Для гарантування стабільної роботи гри після впровадження оновлень було створено комплекс тестів. Результати виконаних тестувань узагальнені та наведені у вигляді таблиць 3.1 – 3.4.

Таблиця 3.1 – Тестування функції запуску гри

Назва функції	**Тестування функції запуску гри**
Стартовий етап	Гравець перебуває на екрані головного меню
Виконані дії	Гравець натискає на кнопку Play
Результат	Гра запускається
Виявлений баг	Не виявлено
Корекція багу	Функція не потребує корекції
Відповідність вимогам	Відповідає функціональним вимогам

Таблиця 3.2 – Перевірка функціональності переміщення об’єкта за допомогою клавіш клавіатури

Назва функції	**Перевірка функціональності переміщення об’єкта за допомогою клавіш клавіатури**
Стартовий етап	Об’єкт перебуває на одному з ігрових рівнів
Виконані дії	Гравець використовує клавіші на клавіатурі для керування рухом об’єкта
Результат	Об’єкт рухається у напрямку, що відповідає натиснутим клавішам
Виявлений баг	Не виявлено
Корекція багу	Функція не потребує корекції
Відповідність вимогам	Відповідає функціональним вимогам

Таблиця 3.3 – Тестування отримання шкоди від супротивника

Назва функції	**Тестування отримання шкоди від супротивника**
Стартовий етап	Гравець перебуває на одному з ігрових рівнів, де є супротивник
Виконані дії	Гравець наближається до супротивника
Результат	Персонаж отримує шкоду
Виявлений баг	Не виявлено

Продовження таблиці 3.3

Назва функції	**Тестування отримання шкоди від супротивника**
Корекція багу	Функція не потребує корекції
Відповідність вимогам	Відповідає функціональним вимогам

Таблиця 3.4 – Тестування можливості придбати покращення

Назва функції	**Тестування можливості придбати покращення**
Стартовий етап	Гравець перебуває на другому рівні гри
Виконані дії	Гравець звертається до npc
Результат	Персонаж може придбати більше однієї копії покращення
Виявлений баг	Виявлено некоректну функцію звернення до неіснуючого списку
Корекція багу	Баг був виправлений після зрозуміння, що інвентар має лише ім’я, а не є повним списком, яким був інвентар торговця
Відповідність вимогам	Відповідає функціональним вимогам

На даному етапі тестування був виявлений баг, сутність якого полягає у тому, що функція звернення гравця на отримання більше однієї копії покращення не була коректною (див. рис. 3.23).



Рисунок 3.23 – Результат звернення гравця на придбання покращення

Стало зрозумілим, що інвентар має лише ім’я, а не є повним списком як був інвентар торгі\овця. До оновлення покупка могла призвести до крашу гри через неправильний тип змінної.

У цілях дебагу було додано тимчасовий код для показу вмісту інвентаря гравця (див. рис. 3.24).

```
// inventory draw, draws items on player position debug use
only
for (var i=0; i<ds_list_size(global.inv); i++){
      draw_text(x, y+i*30, global.inv[| i]);
}
```

Рисунок 3.24 – Тимчасовий код для показу вмісту інвентаря

Для корекції виявленого багу був змінений код (див. рис. 3.25).

```
if (keyboard_check_pressed(ord("Q")) && global.hp_mass >
price) {
            //check if already bought an item
            var bought = false;
            if ds_list_empty(global.inv) == true {}
            else {
                  for (var i = 0;
i<ds_list_size(global.inv); i++) {
                        var arr_player = global.inv[| i];
                        //show_debug_message("got here!")
                        if (string(item) ==
string(arr_player)) {
                              bought = true;

      //show_debug_message("Tested: " + string(arr_player)
+ " with " + string(item));
                              break;
                        }
                  }
            }

            if bought == false {
                  //add to inventory
                  ds_list_add(global.inv, item);

                  //take the price
                  global.hp_mass -= price;
            }
            else {//add to inventory
                  ds_list_add(global.inv, item);

                  //take the price
                  global.hp_mass -= price;}
      }
```

Рисунок 3.25 – Код виправлення багу

Отримані результати тестування функціональних можливостей дозволили сформувати цілісне уявлення про роботу гри після впроваджених оновлень. У процесі перевірки були виявлені окремі помилки, які своєчасно усунено, що дало змогу забезпечити відповідність гри заданим вимогам. Завдяки цьому користувач може взаємодіяти з оновленою версією гри з комфортом, без труднощів у навігації та керуванні.

Таким чином, у результаті проведеної роботи було реалізовано оновлення гри, яке охоплює всі ключові етапи розробки від концептуального проєктування до фінального тестування. Здійснено додавання нового функціоналу, покращено ігрову механіку, розширено можливості взаємодії з користувачем. Завдяки впровадженим змінам гра стала більш динамічною, цікавою та зручною для гравців. Проведене тестування підтвердило працездатність оновлень і відповідність функціоналу вимогам, що дозволяє вважати проєкт реалізованим успішно.

ВИСНОВКИ

В межах даної кваліфікаційної роботи здійснено комплексний аналіз сучасних середовищ розробки ігрового програмного забезпечення, зокрема Unity, Unreal Engine та GameMaker. В результаті проведеного порівняльного огляду визначено ключові переваги та недоліки кожного середовища з позицій функціональних можливостей, складності освоєння, продуктивності та доцільності використання для реалізації двовимірних ігор.

На основі аналізу обґрунтовано вибір ігрового рушія GameMaker як оптимального інструменту для реалізації оновлення гри “Junkosmos”. Визначальними факторами такого вибору стали орієнтація рушія на розробку 2D-проєктів, зручний візуальний інтерфейс, низький поріг входження для розробників, а також підтримка спеціалізованої мови програмування Game Maker Language (GML), яка забезпечує гнучку побудову ігрової логіки та ефективну взаємодію з об’єктами. Характеристика функціональних можливостей GameMaker та аналіз мови GML підтверджують доцільність його застосування в межах поставленої мети роботи, що дозволяє забезпечити високу якість розробки та подальшу підтримку ігрового продукту.

Наступним етапом було проведено проєктування розширених можливостей гри на трьох рівнях: концептуальному, логічному та фізичному, що дозволило чітко структурувати процес розробки та закласти основу для ефективної реалізації поставлених задач.

У ході виконання роботи було повністю реалізовано оновлення гри “Junkosmos”, що охоплює повний життєвий цикл проєктування програмного забезпечення – від аналізу вимог та розробки концепції до написання коду, тестування та оптимізації. Було створено нові функціональні можливості, включаючи систему покращень, магазин, нові локації та завдання, що значно розширили ігровий процес та підвищили інтерес гравця. Усі етапи розробки супроводжувалися ретельним аналізом та перевіркою якості реалізації.

Завдяки тестуванню було виявлено та усунуто помилки, що дозволило досягти стабільної та зручної у використанні версії гри. Отже, оновлення не лише покращило загальну якість гри, а й продемонструвало ефективність застосованого підходу до проєктування та реалізації програмного продукту.

Загалом, виконана робота демонструє системний підхід до розробки ігрового оновлення, що забезпечує оптимальне поєднання теоретичних засад та практичної реалізації, і створює передумови для подальшого розвитку проєкту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Anderson B. GameMaker Language: An In-Depth Guide. HeartBeast Studios, 2015. 208 p.

2. Auckett N. GameMaker Essentials. Birmingham : Packt Publishing, 2015. 136 p.

3. Birn J. Digital Lighting and Rendering. Boston : Pearson Education, 2013. 464 p.

4. Buss S. R. V.2 Bump mapping. *3D Computer Graphics : A Mathematical Introduction with OpenGL*. Cambridge : Cambridge University Press, 2003. 371 p.

5. David B. Hands-On Game Development Patterns with Unity. Birmingham : Packt Publishing, 2019. 242 p.

6. GameMaker Studio 2. URL: https://www.yoyogames.com (дата звернення: 25.03.2025).

7. Habgood J. Overmars M. The Game Maker's Apprentice: Game Development for Beginners. Berkeley : Apress, 2006. 336 p.

8. Make 2D Games With GameMaker : Free Video Game Maker. URL: https://gamemaker.io/en (дата звернення: 14.03.2025).

9. Matthew D. GameMaker Game Programming with GML. Birmingham : Packt Publishing, 2014. 350 p.

10. Rabin S. Improvements to A Search for Real-Time Applications. Game AI Pro 360: Guide to Pathfinding. Boca Raton : CRC Press, 2018. 314 p.

11. Ronny B. What Unreal Engine 5 Means for the Games Industry. URL: https://www.reddit.com/r/PS5/comments/urxrax/what_unreal_engine_5_means_for_the_games_industry/ (дата звернення: 16.04.2025).

12. Schell J. The Art of Game Design: A Book of Lenses : 2nd edition. Jesse Schell, 2014. 600 p.

13. Tyers B. Practical GameMaker: Studio. Berkeley : Apress, 2016. 313 p.

14. Unity Real-Time Development Platform: 3D, 2D, VR & AR Engine.

Unity RealTime Development Platform: 3D, 2D, VR & AR Engine. URL: https://unity.com (дата звернення: 04.02.2025).

15. Unreal Engine: The most powerful real-time 3D creation tool. *Unreal Engine*. URL: https://www.unrealengine.com/en-US (дата звернення: 08.02.2025).

16. Unreal Engine. *Epic Games*. URL: https://www.unrealengine.com/ (дата звернення: 24.04.2025).

17. Wolf M. J. P. Encyclopedia of Video Games: The Culture, Technology, and Art of Gaming Santa Barbara. CA : Greenwood, 2012. 740 p.

18. Junkosmos. URL: https://github.com/Louskil/Junkosmos/tree/main (дата звернення: 03.05.2025).

19. Гогерчак Г. І. Інформаційні системи та бази даних : навч. посібник. Київ : Видавництво «Лікей», 2019. 400 с. URL: https://isdb.itolymp.com/Hoherchak_Informational_system_and_databases.pdf (дата звернення: 11.05.2025).

ДОДАТОК А

Система управління персонажем

Система управління персонажем. На рисунку А.1 представлено відповідний код.

```
// WASD controls a
if keyboard_check(ord("W"))
{
            if speed > 5 { if speed > 8 {} else
{motion_add(image_angle, 0.1);} }
            else {motion_add(image_angle, 0.15);}
}
if keyboard_check(ord("A"))
{
        image_angle += 4;
}
if keyboard_check(ord("D"))
{
        image_angle -= 4;
}
if stage >= 2
{
      if keyboard_check(ord("S"))
      {
            if speed > 5 { if speed > 8 {} else
{motion_add(image_angle+180, 0.1);} }
            else {motion_add(image_angle+180, 0.15);}
      }
      if keyboard_check(vk_down)
      {
            if speed > 5 { if speed > 8 {} else
{motion_add(image_angle+180, 0.1);} }
            else {motion_add(image_angle+180, 0.15);}
      }
}
```

Рисунок А.1

Кадри невразливості та повільне зниження швидкості руху. На рисунку А.2 представлено відповідний код.

```
if invul_time > 0
{
      invulnerable = true;
      invul_time -= 1
}
else {invulnerable = false;}


if speed > 0
{
      speed -= 0.02
}
if speed < 0
```

Рисунок А.2

ДОДАТОК Б

Скрипт бою з босом

Скрипт бою з босом, рандомізований вибір патерів атак. На рисунках Б.1 та Б.2 представлено відповідній код.

```
if lock_in > 0 {lock_in -= 1}
if shot_cooldown > 0 {shot_cooldown -= 1}
image_angle += 0.5;
if global.cd > 0
{
	global.cd -= 1;
}
else if global.cd < 0 {global.cd = 0;}

if global.boss_hp <= 0
{
	global.boss_defeated = true;
	instance_destroy(Obj_boss_canons);
	instance_destroy();
}

if global.cd = 0
{
	choice = round(random(1))
	if choice == 0
	{
		global.multishot_attack = true;
	}
	else if choice == 1
	{
		global.spin_attack = true;
		//global.shoot = false;
	}
}
else
{
	//choice = -1;
	//global.shoot = true;
}
if global.spin_attack == true
{
	if lock_in <= 0 {lock_in += 320; global.cd += 640;}
	if lock_in == 4 {global.spin_attack = false;}
	if shot_cooldown <= 0
	{
		global.bullet_angle = image_angle;
		instance_create_layer(x, y, "Instances",
Obj_enemy_bullet_boss);
		global.bullet_angle = image_angle + 90;
		instance_create_layer(x, y, "Instances",
Obj_enemy_bullet_boss);
		global.bullet_angle = image_angle + 180;
		instance_create_layer(x, y, "Instances",
Obj_enemy_bullet_boss);
		global.bullet_angle = image_angle + 270;
		instance_create_layer(x, y, "Instances",
Obj_enemy_bullet_boss);
		shot_cooldown += 15;
	}
}
```

Рисунок Б.1

```
direction = random(360);
image_angle = random(360);
global.boss_hp = 1000;
global.boss_hp_max = 1000;
global.cd = 0;
global.multishot_attack = false;
global.spin_attack = false;
global.shoot = false;
choice = -1;
lock_in = 0;
shot_cooldown = 0;
```

Рисунок Б.2