**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ**
**ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ**
**КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ**

# КВАЛІФІКАЦІЙНА РОБОТА

**на тему:** Проєктування комп'ютерної гри на базі рушія GameMaker: Studio

Здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи ЦДБ-20-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Технології цифрового дизайну»

_____ Задорожний Тарас Володимирович
(підпис) (прізвище, ім'я, по батькові)

Науковий керівник _____ Омельчук Антон Анатолійович, к.т.н.
(підпис) (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Керівник освітньо-професійної програми _____ Ратушняк Тетяна Володимирівна, к.ф.-м.н., доцент
(підпис) (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних технологій і систем протокол від «22» травня 2024 № 15

Завідувач кафедри _____ Філоненко Михайло Миколайович, к.ф.-м.н., доцент
(підпис) (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Ірпінь – 2024

Реєстрація №46 31.05.2024 _____
(дата) (підпис відповідальної особи)

Результати перевірки до захисту 22.05.2024 _____
(до захисту, на доопрацювання) (дата) (підпис)

Результат захисту 5 (відмінно) 12.06.2024 _____
(оцінка) (дата) (підпис)

Голова комісії: _____ В.М. Франчук
(підпис)

Члени комісії: _____ М.М. Філоненко
(підпис)

_____ С.М. Грищенко
(підпис)

_____ В.О. Ніжегородцев
(підпис)

# БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

| Об'єкти контролю | Діапазон оцінки в балах | Фактична кількість балів | Примітки | Дата, П.І.Б.: керівника, голови комісії членів комісії,секретаря, (підписи) |
|---|---|---|---|---|
| ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ | | | | |
| 1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій) | від 0 до 30 | 29 | | 12.06.2024 Франчук В.М. Філоненко М.М. Грищенко С.М. Ніжегородцев В.О. Суховецька Л.П. |
| 2. Дотримання вимог щодо оформлення | від 0 до 10 | 9 | | |
| 3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни | від 0 до 5 | 4 | | |
| 4. Рецензент | від 0 до 5 | 5 | | |
| РАЗОМ БАЛІВ | від 0 до 50 | 47 | | |

**Оцінювання кваліфікаційної роботи науковим керівником - 45 балів.**

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

| Оцінка за 100-бальною шкалою | Оцінка за шкалоюЄКТС | Оцінка за національною шкалою | Дата | П.І.Б.: голови, членів комісії, секретаря, (підпис) |
|---|---|---|---|---|
| 92 | A | 5(відмінно) | 12.06.2024 | Франчук В.М. Філоненко М.М. Грищенко С.М. Ніжегородцев В.О. Суховецька Л.П. |

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

| Оцінка за 100-бальною шкалою | Оцінка за шкалою ЄКТС | Оцінка за національною шкалою |
|---|---|---|
| 90-100 | A | 5 (відмінно) |
| 80-89 | B | 4 (добре) |
| 70-79 | C | |
| 60-69 | D | 3 (задовільно) |
| 50-59 | E | |
| 35-49 | FX | 2 (незадовільно) |

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп'ютерні науки»
Освітня програма: «Технології цифрового дизайну»

**Затверджую**
Завідувач кафедри Філоненко М.М.
«23» жовтня 2023

**Завдання на кваліфікаційну роботу**
**здобувачу вищої освіти**
**Задорожному Тарасу Володимировичу**
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Проєктування комп'ютерної гри на базі рушія GameMaker: Studio».
Затверджена наказом ректора від «20» жовтня 2023 № 1411
2. Строк здачі закінченої роботи «21» травня 2024
3. Мета роботи: розробка комп'ютерної гри у жанрі Платформер.
Об'єкт дослідження: комп'ютерні ігри, розроблені за допомогою рушія GameMaker: Studio.
Предмет дослідження: теоретичні та практичні аспекти розробки комп'ютерної гри на базі рушія GameMaker: Studio.
Орієнтовна база дослідження: функціональні можливості рушія Game Maker: Studio для створення ігор.
4. Вихідні дані до роботи: тема гри та її жанр, цільова аудиторія, ігрові механіки, ігровий світ, сюжет, аудіо, технічні обмеження.
5. Перелік графічного матеріалу: скріншоти гри, алгоритми ігрової логіки, концепт-арт, текстури та спрайти, скріншоти кодів.
6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом):
аналіз розвитку комп'ютерних ігор, технології розробки комп'ютерних ігор, аналіз рушія GameMaker: Studio, структура проєкту комп'ютерної гри, розробка графічних компонентів гри, дизайн рівнів і ігрового середовища, алгоритм функціонування гри, тестування програмного забезпечення, підготовка до публікації проєкту на Kickstarter, вдосконалення і технічна підтримка проєкту.

7. Календарний план виконання роботи

| № п/п | Назва етапів кваліфікаційноїроботи | Строк виконання етапів роботи | |
|---|---|---|---|
| | | За планом | Фактично |
| 1 | 2 | 3 | 4 |
| 1 | Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи | До 15.10.2023 | До 15.10.2023 |
| 2 | **І розділ** (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень) | 29.12.2023 | 29.12.2023 |
| 3 | **ІІ розділ** (аналіз експериментальної частини про методики досліджень) | 16.02.2024 | 16.02.2024 |
| 4 | **ІІІ розділ** (проведені теоретичні і (або) експериментальні дослідження та проектні рішення) | 16.04.2024 | 16.04.2024 |
| 5 | **Висновки** | 30.04.2024 | 30.04.2024 |
| 6 | Доопрацювання роботи з урахуванням зауважень наукового керівника | 10.05.2024 | 10.05.2024 |
| 7 | Отримання відгуку керівника | 30.05.2024 | 30.05.2024 |
| 8 | Отримання зовнішньої рецензії | 03.06.2024 | 03.06.2024 |

8. Дата видачі завдання «23» жовтня 2023

9. Керівник кваліфікаційної роботи

(підпис) Омельчук А.
(прізвище, ініціали)

10. Керівник освітньо-професійної програми

(підпис) Ратушняк Т.І
(прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти

Задорожний

# АНОТАЦІЯ

"Проєктування комп'ютерної гри на базі рушія GameMaker: Studio"

*Мета роботи полягає* у створенні комп’ютерної гри у жанрі Платформер
*Завдання роботи:*

- Теоретичне дослідження основ розробки комп’ютерних ігор;
- Проєктування гри;
- Реалізація гри;
- Аналіз результатів створеного продукту.

*Предметом дослідження* є теоретичні та практичні аспекти розробки комп'ютерної гри на базі рушія GameMaker: Studio.
*Об’єктом дослідження* є комп'ютерні ігри, розроблені за допомогою рушія GameMaker: Studio.
*Базою дослідження* є функціональні можливості рушія Game Maker: Studio для створення ігор.
*За результатами дослідження сформульовані висновки про* особливості проєктування та можливості рушія GameMaker: Studio для розробки комп'ютерних ігор, практичні рекомендації.

*Одержані результати можуть бути використані* для вдосконалення, розробки нових комп’ютерних ігор, навчання майбутніх розробників, подальших досліджень.

Кваліфікаційна робота містить 88 сторінок, 11 рисунків, 5 таблиць, список використаних джерел містить 40 найменувань.

*Ключові слова:* платформер, код, дизайн, GameMaker: Studio.

*Рік виконання кваліфікаційної роботи* 2024

*Рік захисту роботи* 2024

## ABSTRACT

‘Designing a computer game based on the GameMaker: Studio’

The aim of the work is to create a computer game in the genre of Platformer

Tasks of the work:

- Theoretical study of the basics of computer game development;
- Game design;
- Implementation of the game;
- Analysis of the results of the created product.

The subject of research is the theoretical and practical aspects of developing a computer game based on the GameMaker: Studio engine.

The object of research is computer games developed using the GameMaker: Studio engine.

The basis of the study is the functionality of the Game Maker: Studio engine for creating games.

Based on the results of the study, conclusions are made about the design features and capabilities of the GameMaker: Studio engine for developing computer games, as well as practical recommendations.

The obtained results can be used for improvement, development of new computer games, training of future developers, and further research.

The qualification work consists of 87 pages, 11 figures, 5 tables, the list of references includes 40 items.

Keywords: platformer, code, design, GameMaker: Studio.

Year of completion of the qualification work 2024

Year of defence 2024

ЗМІСТ

ВСТУП ..... 3
РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ..... 6
1.1 Аналіз розвитку комп’ютерних ігор ..... 6
1.2 Технології розробки комп’ютерних ігор ..... 14
1.3 Аналіз рушія GameMaker: Studio ..... 19
РОЗДІЛ 2. РЕАЛІЗАЦІЯ ПРОЄКТУ ..... 24
2.1 Структура проєкту комп’ютерної гри ..... 24
2.2 Розробка графічних компонентів гри ..... 30
2.3 Дизайн рівнів і ігрового середовища ..... 34
2.4 Алгоритми функціонування гри. ..... 39
РОЗДІЛ 3. АНАЛІЗ ОТРИМАНОГО РЕЗУЛЬТАТУ ..... 49
3.1 Тестування програмного забезпечення ..... 49
3.2 Підготовка до публікації проєкту на Kickstarter ..... 51
3.3 Вдосконалення і технічна підтримка проєкту ..... 53
ВИСНОВОК ..... 56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ..... 58
ДОДАТОК А ..... 62
ДОДАТОК Б ..... 64
ДОДАТОК В ..... 66

# ВСТУП

Сучасна індустрія комп'ютерних ігор переживає бурхливий розвиток, стаючи не лише сферою розваг, але й потужним інструментом освіти, тренування та досліджень. Швидкий та постійний розвиток технологій дозволяє розробникам створювати все більш складні, захоплюючі та інноваційні ігрові продукти. У зв'язку з цим зростає потреба у кваліфікованих фахівцях, які володіють знаннями та навичками розробки ігор. Розробка комп'ютерних ігор є динамічно зростаючою сферою, яка поєднує в собі елементи програмування, дизайну, мистецтва та оповіді. Створення захоплюючої та інтерактивної гри потребує не лише технічних навичок, але й творчого підходу та розуміння психології гравців.

Дана кваліфікаційна робота призначена для детального аналізу та практичного використання процесу розробки комп'ютерних ігор, зосередженого на використанні рушія GameMaker: Studio. У ній буде розглянуто теоретичні основи розробки ігор, охарактеризовано можливості рушія GameMaker: Studio, а також описано процес проєктування та створення комп'ютерної гри. У ході дослідження будуть розглянуті основні принципи та концепції створення комп'ютерних ігор, а також особливості використання рушія GameMaker: Studio. Будуть проаналізовані існуючі ігри, розроблені з використанням даного інструменту, для отримання прикладів кращих практик та інсайтів щодо ефективного використання програмного забезпечення. Очікується, що результатом даної кваліфікаційної роботи буде повноцінна комп'ютерна гра, розроблена з використанням рушія GameMaker: Studio, яка відповідає сучасним стандартам і має цікавий геймплей для користувачів. Вибір саме цього рушія зумовлений для розробки гри зумовлений його великою популярністю серед початківців у галуз розробки ігор, простотою використання, доступністю та широкими можливостями. Цей рушій дозволяє створювати як 2D, так і 3D ігри, а також пропонує широкий спектр інструментів для програмування, анімації, дизайну рівнів та створення ігрового контенту.

Варто відзначити декілька вдалих проєктів створених за допомогою рушія GameMaker: Studio. Одними з таких є проєкт «Hotline Miami» жанру shoot`em up, Chicory: A Colorful Tale - пригодницька кросплатформенна гра, екшн-платформер Katana ZERO, Hyper Light Drifter жанру RPG, головоломка Toodee and Topdee, окрім яких існує ще досить багато проривних ігор різних жанрів, про які детально буде описано в підрозділі 1.1.

Головною метою дослідження є систематизація та узагальнення теоретичних та практичних знань з проєктування комп'ютерних ігор на базі рушія GameMaker: Studio, показати процес створення комп'ютерної гри на базі даного рушія з використанням сучасних методів та підходів. Розглянемо створення концепції, механіки та візуального оформлення. Дослідимо технічні аспекти реалізації гри, такі як програмування геймплею та обробка введення користувача. Також розглянемо процес тестування та виправлення помилок для забезпечення якості гри та задоволення користувачів. Вивчення цього процесу дозволить зрозуміти ключові аспекти розробки ігор та використовувати їх у подальшій практиці.

Подальші кроки у цьому дослідженні включатимуть аналіз та порівняння існуючих ігор, створених з використанням GameMaker: Studio, детальне проєктування гри, програмування, створення графічного та звукового контенту, тестування та оптимізація. Всі ці аспекти будуть ретельно розглянуті з метою збагачення розуміння процесу розробки ігор та підготовки до подальшої кар'єри в цій сфері.

Процес розробки ігор - це поєднання технічної експертизи, креативності та співпраці. Через аналіз та практичне використання процесу розробки гри на прикладі конкретного проєкту, ця дипломна робота спрямована на надання важливих відомостей та навичок у сфері розробки комп'ютерних ігор. Дослідження цього процесу дозволить зрозуміти особливості та використовувати їх у майбутній практиці роботи в ігровій індустрії. Ця робота має велике значення для розвитку інтерактивної індустрії та внесення свого внеску в галузь геймдизайну. Дане дослідження сприятиме не лише

поглибленню розуміння процесу створення комп'ютерних ігор, але й стимулюванню творчого підходу до створення ігор та розвиток ключових навичок, необхідних для успішної роботи в цій захоплюючій галузі. Результати роботи можуть бути використані для створення навчальних курсів з розробки комп'ютерних ігор, а також для подальшого розвитку ігрової індустрії в Україні.

# РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Аналіз розвитку комп’ютерних ігор

Сьогодні комп’ютерні ігри стали не лише формою розваги, а й важливою складовою сучасної культури та технологічного прогресу. Світ комп'ютерних ігор постійно розвивається, що в першу чергу визначається технологічним прогресом і зміною уподобань і смаків гравців. Різноманітність жанрів, від простих аркад до складних імерсивних світів, демонструє стрімке зростання цієї галузі. Вивчення еволюції індустрії комп’ютерних ігор, включаючи платформу GameMaker: Studio, має вирішальне значення для розуміння тенденцій і потенціалу цього зростаючого сектора. Вивчення його історії та прогресу допоможе зрозуміти поточний стан галузі та спрогнозувати її майбутнє [36].

Хронологію розвитку комп’ютерних ігор можна поділити на 6 основних етапів [3]:

1. **Зародження** (1950-1960 роки).

Початковий період розвитку комп’ютерних ігор був представлений тим фактом, що перші експериментальні ігри створювалися на великих дорогих комп’ютерах, які в основному використовувалися для наукових обчислень і військових цілей. Вони працювали на базі вакуумних ламп і магнітних барабанів, а їх потужність обчислень була дуже обмеженою порівняно з сучасними стандартами.

Одним із значних досягнень цієї епохи стало створення гри «Spacewar!» (рис. 1.1) у 1962 році, яка вважається однією з перших комп’ютерних ігор. Ця гра, розроблена на комп’ютері PDP-1 студентами у Массачусетському технологічному інституті, була симулятором космічного бою, що сприяло подальшому розширенню ігрової індустрії. Крім того, у цей період також з’явилися текстові ігри, такі як The Adventure (також відома як The Great Cave Adventure), розроблена Вільямом Кроузером у 1976 році. Гра встановила стандарт для текстових пригодницьких ігор, у яких гравці взаємодіють із

віртуальним світом, вводячи команди на екрані та отримуючи текстові відповіді [16].




Рис. 1.1 Гра «Spacewar!» на комп’ютері PDP-1.

2. **Епоха аркадних ігор** (1970-1980 роки).

Період 1970-1980 років в історії відеоігор відзначився різким зростанням популярності аркадних ігор, які стали суттєвим аспектом поп-культури та розважальної індустрії. Цей період відзначився значними досягненнями в галузі ігрової технології, впливом глобальних хітів та конкуренцією між розробниками, що сприяло інноваціям та зростанню індустрії.

Аркадні автомати (рис. 1.2) стали одним із основних механізмів доступу до відеоігор для широкого загалу на протязі цього періоду. Наприклад, гра "Pong", створена компанією Atari у 1972 році, була однією з перших ігор, яка отримала широкий комерційний успіх та встановила стандарт для багатьох подальших аркадних проєктів.

Ключовим аспектом цього періоду був вплив глобальних хітів, таких як "Space Invaders" (1978) та "Pac-Man" (1980), на ігрову індустрію. Конкуренція між розробниками стимулювала інновації в галузі, що призвело до випуску нових ігор зі складними геймплейними механіками та вражаючими графічними ефектами.




Рис. 1.2 Приклади перших аркадних автоматів.

3. **Розвиток домашніх комп'ютерів та консолей** (1980-1990 роки).

Цей період в історії комп'ютерних ігор відзначився значним розвитком домашніх комп'ютерів, початком епохи домашніх ігрових консолей та їх впливом на ігрову індустрію. 1980-і роки були періодом, коли домашні комп'ютери стали доступнішими для широкого споживача. Великі компанії, такі як Apple, Commodore, та IBM, випустили свої персональні комп'ютери, такі як Apple II, Commodore 64, та IBM PC, що відкрило нові можливості для розваг та ігор у домашньому середовищі [4].

Паралельно з розвитком домашніх комп'ютерів, ігрові консолі також здобули популярність серед споживачів. Компанії Nintendo, Sega, та Atari випустили популярні ігрові системи, такі як Nintendo Entertainment System (NES), Sega Master System, та Atari 2600.

Розвиток домашніх комп'ютерів та ігрових консолей у 1980-1990 роках стимулював значний розвиток ігрової індустрії. Розробники ігор використовували нові можливості технологій для створення більш складних та захоплюючих ігор. Цей період був свідком випуску численних класичних ігор,

таких як "Super Mario Bros", "The Legend of Zelda", "Tetris", які стали символами свого часу та вплинули на подальший розвиток ігрової індустрії.

4. **Індустрія 3D графіки** (1990 – 2000 роки)

У період з 1990 по 2000 роки індустрія відеоігор пережила значний технологічний прорив, який спричинив революцію в галузі 3D-графіки. Цей період відзначився значними досягненнями в області графіки, звуку та ігрової механіки, що привело до появи нового покоління ігор та ігрових консолей. У 1990-і роки було випущено кілька ключових технологій, які революціонізували галузь комп'ютерної графіки. Однією з найважливіших інновацій була використання графічних прискорювачів, які дозволяли обробляти складні 3D-моделі та ефекти з більшою швидкістю та ефективністю.

У цей період було досягнуто значних успіхів у розвитку технологій рендерингу та освітлення. Рендеринг у реальному часі став все більш доступним завдяки використанню алгоритмів, які ефективно використовували обчислювальні можливості комп'ютерів. Освітлення було значно поліпшено завдяки використанню фізично заснованих алгоритмів, які відтворювали реальні властивості світла та тіней. З розвитком технології збільшення обчислювальної потужності комп'ютерів і графічних прискорювачів, розробники стали здатні створювати більш деталізовані та реалістичні 3D-моделі персонажів, об'єктів та оточення. Це дозволило створити ігри з вражаючими візуальними ефектами та забезпечило гравців більш імерсивним ігровим досвідом.

У цей період було випущено кілька ігрових консолей, які стали популярними серед гравців та відкрили нові можливості для розробників ігор. Найважливішими серед них були Sony PlayStation, Nintendo 64 та Sega Saturn. Деякі з найвідоміших ігор цього періоду, які продемонстрували можливості 3D-графіки, включають Doom (1993), Quake (1996), Grand Theft Auto (2001), God of War (2005) та Half-Life (1998).




Рис. 1.3 Скріншот з гри «Grand Theft Auto».

5. **Епоха онлайн-ігор** (2000 – 2010 роки)

У період з 2000 по 2010 роки галузь відеоігор пережила значне зростання популярності онлайн-ігор. Цей період був характеризований розвитком інтернет-технологій та поширенням широкосмугового Інтернету, що сприяло появі нових форм геймінгу та вплинуло на взаємодію людей у віртуальних середовищах.

У цей період спостерігалося значне зростання популярності MMO-ігор (Massively Multiplayer Online), що представляли собою великі масштабні віртуальні світи, де тисячі гравців могли спілкуватися та взаємодіяти в реальному часі. Важливими представниками цього жанру стали World of Warcraft, Lineage II, та EverQuest, що відзначалися великими обсягами контенту та соціальним взаємодією гравців. Онлайн-ігри стали місцем для активної соціальної взаємодії та формування великих спільнот гравців. Гравці об'єднувалися в гільдії, клани та інші групи, щоб спільно виконувати завдання, змагатися або просто спілкуватися. Це відкривало нові можливості для розвитку віртуальних соціальних структур та взаємодії між людьми з різних країн та культур.

6. **Сучасність** (2010 роки – дотепер)

Цей період характеризується стрімким ростом використання новітніх технологій, розширенням географічного розповсюдження гравців, постійними змінами в архітектурі апаратного забезпечення, програмних інструментах, методологіях розробки ігор та стратегіях маркетингу.

Однією з ключових технологічних інновацій є впровадження техніки трасування променів (real-time ray tracing), що дозволяє реалістичніше моделювати освітлення та відображення у відеоіграх (рис. 1.4). Ця технологія вимагає значних обчислювальних ресурсів, але забезпечує вражаючі візуальні ефекти, підвищуючи імерсію гравців у гру. У період сучасності спостерігається розширення жанрового спектру комп'ютерних ігор. Окрім традиційних жанрів, таких як екшн, стратегія та пригодницькі ігри, виникають нові, експериментальні жанри, які поєднують у собі елементи різних жанрів та надають гравцям нові типи геймплею.




Рис. 1.4 Скріншот з гри «Cyberpunk 2077».

На основі проведеного аналізу було визначено, що найуспішнішими сучасними комп'ютерними іграми є:

- Grand Theft Auto V (2013): Дохід склав понад 6 мільярдів доларів.
- Fortnite (2017) - 20 мільярдів доларів.
- Minecraft (2009) - 23 мільярди доларів.

- PlayerUnknown's Battlegrounds (PUBG) (2017) - 7 мільярдів доларів.

Значною подією в сучасності є перехід до нового покоління ігрових консолей, таких як PlayStation 4, Xbox One та Nintendo Switch. Ці консолі відрізняються від попередніх моделей значними покращеннями у графіці, продуктивності та можливостями для розвитку ігорних уявлень. Приклади таких ігор, як "The Last of Us", "The Legend of Zelda: Breath of the Wild" та "Red Dead Redemption 2", ілюструють вплив нового покоління консолей на ігрову індустрію [32].

Сучасність відзначається значними досягненнями в галузі віртуальної реальності (VR). Поява таких пристроїв, як Oculus Rift, HTC Vive та PlayStation VR, відкрило нові горизонти для ігрової індустрії.

Завдяки технологічному розвитку смартфонів та планшетів, мобільні ігри стають все більш доступними та привабливими для широкої аудиторії. "Candy Crush Saga" та "Pokémon GO" - це лише декілька прикладів успішних мобільних ігор, які зробили значний вплив на культуру геймінгу та суспільство в цілому.

З виникненням платформ для онлайн-ігор, таких як Steam, ігрові розробники мають можливість створювати великі масштаби віртуальних світів для мільйонів гравців.

Ігри, які працюють на різних пристроях і платформах, стали все більш популярними. Гравці можуть грати в одну і ту ж гру на різних пристроях, включаючи персональні комп'ютери, консолі та мобільні пристрої, що робить геймінг більш доступним та універсальним.

Крім того, кіберспорт (esports) став важливим аспектом галузі, з організацією міжнародних турнірів та змагань, які привертають увагу мільйонів глядачів по всьому світу та мають колосальні призові фонди. Напкриклад, The International 12 - дванадцятий чемпіонат світу з комп'ютерної гри Dota 2 жанру MOBA, організований компанією-розробником гри Valve та турнірним оператором PGL, який відбувся з 12 по 29 жовтня 2023 року мав призовий фонд на 30 жовтня 3380455 $.

У сучасній індустрії комп'ютерних ігор існує велика кількість жанрів, кожен з яких має свої особливості та аудиторію. На сьогоднішній день можна виділити такі популярні жанри:

- **Шутери (FPS, TPS):** Це ігри, в яких гравець керує персонажем та стріляє у ворогів з різних видів зброї.
- **Рольові ігри (RPG):** Гравець відтворює роль певного персонажа і взаємодіє з ігровим світом, розвиваючи свої навички та вирішуючи завдання.
- **Стратегії:** Це ігри, де гравець має планувати та керувати стратегічними діями, щоб досягти перемоги над супротивником.
- **Ігри-головоломки:** Це ігри, в яких гравцеві потрібно розв'язати різні головоломки або логічні завдання.
- **Симулятори:** Це ігри, що моделюють реальні або уявні ситуації, де гравцеві потрібно керувати певними об'єктами або процесами.

Різноманіття жанрів свідчить про багатогранність інтересів гравців та можливостей розробників.

Розвиток технологій, зокрема швидкість обробки даних, графічні рушії, штучний інтелект та віртуальна реальність, має значний вплив на характер і зміст комп'ютерних ігор. Високоякісна графіка, реалістична фізика та інші технічні новації стають стандартом для сучасних ігор, що відкриває нові можливості для творчості та інновацій. Завдяки швидкому зростанню потужності обчислювальних систем, розробники можуть створювати більш реалістичні та імерсивні світи для гравців. Крім того, технології віртуальної реальності відкривають нові можливості для ігрової індустрії, дозволяючи гравцям занурюватися у цілком нові ігрові середовища.

Останнім часом спостерігається збільшення популярності інді-ігор, які розробляються невеликими незалежними. Ці ігри відрізняються оригінальними концепціями та нестандартним підходом до геймплею, що знаходить відгук серед широкого кола гравців.

Аналіз розвитку комп'ютерних ігор показує, що галузь постійно змінюється та розвивається відповідно до технологічних та культурних тенденцій. Розуміння цих тенденцій є важливим для ефективного проєктування та реалізації ігор, а також для створення продуктів, які відповідають потребам та очікуванням сучасних гравців.

## 1.2 Технології розробки комп’ютерних ігор

Розробка комп'ютерних ігор вимагає використання різноманітних технологій та інструментів, які допомагають розробникам створювати ігри з високою якістю, захоплюючим геймплеєм та вражаючою графікою. Технології розробки комп’ютерних ігор відіграють ключову роль у створенні ігор, які вражають гравців своєю глибиною, імерсією та візуальною привабливістю.

Якщо раніше для розробки ігор потрібні були значні кошти та ресурси, то сьогодні доступні інструменти дозволяють навіть невеликим командам створювати вражаючі проєкти зі складною графікою та глибоким сюжетом. Останні роки принесли значний прогрес у технологіях розробки ігор. У розробці комп'ютерних ігор використовуються різноманітні технології, серед яких ключове значення мають ігрові рушії (англ. game engines).

**Ігрові рушії** – це програмні засоби, призначені для створення та реалізації ігрового середовища, включаючи графіку, фізику, штучний інтелект, звукові ефекти та інші аспекти ігрового досвіду. Нижче наведено перелік одних з найвідоміших кросплатформених рушіїв:

1. **Unity**. Розроблений компанією Unity Technologies, цей інструментарій здійснює вплив на широкий спектр різноманітних сегментів індустрії розваг та додатків, починаючи від мобільних ігор і закінчуючи віртуальною реальністю. Однією з ключових характеристик Unity є його інтегроване середовище розробки, яке надає розробникам широкий спектр інструментів для створення графіки, анімації, фізики, штучного інтелекту та інших аспектів ігрового досвіду. З врахуванням високої поширеності та активності спільноти розробників, Unity також відзначається широким спектром додаткових ресурсів та плагінів, що доповнюють його базовий функціонал. Прикладами успішних ігор створених на

цьому рушії є такі проєкти, як Cuphead, Pokemon Go, Inside та Hearthstone. Однією з найпопулярніших комп'ютерних ігор, розроблених за допомогою Unity, є "Hollow Knight" розроблена компанією Team Cherry, за яку розробники отримали дохід близько 42 млн $ [11].

2. **Unreal Engine**. UE є комплексним інструментарієм розробки відеоігор та інтерактивних додатків, розробленим Epic Games. Цей рушій визначається своєю високою функціональністю, ефективністю та високоякісними графічними можливостями. Заснований на власному програмному забезпеченні Unreal Engine, він надає широкий набір інструментів для розробки відомих графічних ігор. Unreal Engine включає в себе інтуїтивно зрозумілі редактори для моделювання, текстурування, анімації, а також графічний інтерфейс для створення штучного інтелекту, фізичної поведінки об'єктів та управління звуками. Крім того, він має велику бібліотеку готових ресурсів, включаючи 3D-моделі, текстури, звукові ефекти. Даний рушій надає потужні можливості візуалізації, що дозволяє створювати інноваційні графічні образи, включаючи реалістичне освітлення, тіні, водні ефекти, анімацію та багато іншого (рис. 1.5) [12, 26].




Рис 1.5 Створення персонажу в рушії Unreal Engine.

Приклади ігор розроблених за допомогою UE:

- Fortnite;
- Gears of War 5;

- Hellblade: Senua's Sacrifice;
- Sea of Thieves;
- BioShock Infinite;
- The Vanishing of Ethan Carter;

3. **CryEngine**. Є одним з високоактуальних гральних рушіїв, визнаним за його передові графічні та фізичні можливості. Розроблений компанією Crytek, цей рушій демонструє високий рівень технічної компетентності в області комп'ютерної графіки та програмування. Спеціально створений для гри "Far Cry" (рис. 1.6) у 2004 році, CryEngine відтоді пройшов значний шлях у розвитку та підвищенні своїх можливостей. Суттєва особливість CryEngine полягає у використанні передових методів рендерингу, включаючи в себе техніки, такі як *deferred shading* та *screen-space reflections*. Ці технології дозволяють досягти вражаючих рівнів візуальної реалістичності та деталізації, створюючи ефекти освітлення та відображення, що наближені до реального світу. Значна частина привабливості CryEngine походить від його потужної фізичної системи, яка базується на принципах імпульсної динаміки. Ця система надає можливість точного моделювання руху об'єктів та їх взаємодії. Необхідно зазначити, що CryEngine демонструє високий рівень оптимізації [8,13].



Рис. 1.6 Скріншот з гри «Far Cry».

Новітні технології, такі як штучний інтелект, віртуальна та доповнена реальність, а також блокчейн, значно розширюють можливості розробників ігор. Вони дозволяють створювати ігри з більш глибокою взаємодією з гравцями, інноваційними ігровими механіками, реалістичнішими ігровими світами, новими способами монетизації та спілкування з аудиторією. Наприклад, штучний інтелект може контролювати поведінку ворожих персонажів або реагувати на дії гравця в реальному часі, створюючи непередбачувані сценарії гри.

У відповідності з швидким розвитком індустрії відеоігор та постійним зростанням вимог до їх якості та функціональності, вибір мови програмування для розробки ігор є ключовим аспектом у процесі створення гри. Кожна мова має свої переваги та недоліки, а також власні області застосування, що робить їх інструментами з відмінною сферою впливу у галузі розробки ігор. Розглянемо кілька ключових мов програмування, які широко використовуються у цій сфері.

1. **C++**

C++ є однією з найпопулярніших мов програмування у галузі розробки ігор завдяки своїй ефективності та високій швидкості виконання. Вона володіє потужними можливостями низькорівневого доступу до апаратних ресурсів комп'ютера, що дозволяє розробникам максимально оптимізувати продуктивність гри. Крім того, C++ забезпечує великий контроль над ресурсами, такими як пам'ять і процесор, що робить її відмінним вибором для створення високоякісних та ресурсомістких ігор. Цю мову програмування активно використовують у таких ігрових рушіях, як Unreal Engine та CryEngine.

2. **Java**

Java є платформонезалежною мовою програмування, яка знаходить широке застосування у розробці мобільних ігор та ігор для веб-серверів. Вона відома своєю високою рівнем безпеки та зручним механізмом управління пам'яттю. Java також має потужну екосистему інструментів та бібліотек, що сприяє швидкій розробці та розгортанню ігор. Зокрема, мова програмування Java

застосовується у Unity, що використовує у собі .NET Framework, що підтримує C#, але також має підтримку Java.

3. **Python**

Python відзначається своєю простотою та легкістю в освоєнні, що робить її популярним вибором для розробки прототипів та швидких ігор. Високорівневий синтаксис Python сприяє швидкому розвитку проектів, а велика кількість готових бібліотек та фреймворків дозволяє розробникам швидко реалізувати потрібні функціональність. Python використовується у багатьох іграх та ігрових рушіях для різноманітних завдань, включаючи розробку ігрових скриптів, інструментів розробки та автоматизації. Наприклад, в ігровому рушії Godot Engine, яке підтримує скрипти Python, розробники можуть використовувати Python для створення складних ігрових логік та взаємодії об'єктів.

4. **C#**

C# є ще однією з найпоширеніших об'єктно-орієнтованих мов програмування. Була розроблена компанією Microsoft у 2001 році та є вдосконаленим прототипом першої об'єктно-орієнтованої мови програмування C. Часто використовується для розробки ігор для платформ Microsoft Windows та Xbox, завдяки інтеграції з Microsoft Visual Studio та платформою .NET. Вона має багатий набір інструментів для розробки графічного інтерфейсу та доступу до графічних бібліотек, таких як Unity, що робить її привабливою для ігрової розробки. C# також широко використовується для розробки ігор для розширеної та віртуальної реальності. Платформи, такі як Microsoft HoloLens та Oculus Rift, підтримують розробку ігор з використанням C# та фреймворків, таких як Unity.

5. **Мова програмування Lua**

Lua часто використовується як скриптова мова в іграх з великим світом або MMO, завдяки своєму простому синтаксису та легкій інтеграції з існуючими ігровими двигунами. Вона дозволяє розробникам швидко впроваджувати та змінювати геймплей за допомогою скриптів. Мову програмування Lua можна знайти у відомих ігрових рушіях, таких як World of Warcraft (WoW) та Roblox. У WoW, Lua використовується для написання *аддонів*, які розширюють

функціональність гри. У Roblox, Lua використовується як мова скриптів для створення ігор та інтерактивних додатків в середовищі гри.

Інтегровані середовища розробки надають розробникам зручний інструментарій для написання, візуального редагування та анімції, тестування коду, гри на різних етапах розробки та інших процесів. Visual Studio є одним з найпопулярніших IDE для розробки ігор на мовах програмування C++ та C#. Він має багато корисних функцій, таких як автодоповнення коду, візуальне налагодження та інтеграцію з різними системами контролю версій. JetBrains Rider також є потужним інструментом для розробки ігор на основі C# та інших мов програмування.

Крім основних компонентів розробки ігор, розробники також можуть використовувати різні додаткові технології для покращення функціональності та враження від гри. Наприклад, аудіо рушії, такі як FMOD та Wwise, дозволяють розробникам створювати складний звуковий дизайн для своїх ігор, включаючи реалістичний звук оточення, музику та діалоги персонажів. Фізичні рушії, наприклад PhysX та Havok, дозволяють реалістично моделювати фізичні взаємодії у грі, такі як колізії, рух та зіткнення об'єктів.

### 1.3 Аналіз рушія GameMaker: Studio

GameMaker: Studio - це інтегроване середовище розробки (IDE) та рушій для створення 2D-ігор, яке виробляється компанією YoYo Games. Він відіграє важливу роль в індустрії розробки ігор, забезпечуючи широке коло розробників інструментами для швидкого та ефективного створення 2D-ігор. Цей рушій відомий своєю простотою використання та потужними можливостями для розробки різноманітних ігрових проєктів, від простих аркад до складних платформерів та стратегій.

GameMaker: Studio є нащадком відомого інструменту розробки ігор GameMaker, розробленого Марком Овермарсом у 1999 році. Оригінальний GameMaker був спрямований на початківців та недосвідчених розробників ігор і надавав простий інтерфейс та мову скриптування GML (GameMaker Language) для створення ігор. Протягом років GameMaker еволюціонував і, нарешті, у 2012

році був випущений GameMaker: Studio, що представив вдосконалену платформу розробки, зокрема, з підтримкою мультиплатформенності.

Однією з ключових переваг GameMaker: Studio є його здатність спростити процес розробки ігор, зменшивши бар'єри для вступу в галузь розробки комп’ютерних ігор. Розробники можуть швидко перетворювати свої ідеї в реальність за допомогою інтуїтивного візуального редактора, готових модулів та бібліотек, а також мови програмування GML, яка дозволяє швидко прототипувати та реалізовувати концепції гри. Хоча GameMaker: Studio відомий своєю простотою, він також пропонує значну гнучкість та можливості для досвідчених розробників. Від високорівневих можливостей керування рухом та колізіями до низькорівневих налаштувань графіки та аудіо, рушій надає широкий спектр інструментів для створення ігор різної складності та жанрів [25].

GameMaker: Studio дозволив безлічі талановитих розробників з усього світу створити успішні ігри, які здобули популярність у геймерській спільноті. Його простий інтерфейс та широкі можливості дозволяють втілювати навіть найскладніші ідеї в реальність, забезпечуючи незабутні ігрові досвіди для гравців. Рушій GameMaker: Studio проявляє відмінну придатність для реалізації ігор жанру платформер завдяки своїй потужній функціональності. Цей інструмент надає розробникам зручності для створення складних рівнів, налаштування фізики руху об'єктів та взаємодії гравця з оточенням. Завдяки багатій палітрі можливостей, включаючи анімацію, обробку звуків, фізичну модель та штучний інтелект, цей рушій виявляється придатним інструментом для розробки платформерів [15].

Розглянемо деякі приклади ігор, які були створені за допомогою Gamemaker: Studio:

- **"Hyper Light Drifter" (2016)**

  "Hyper Light Drifter" - це екшн-RPG, яке поєднує в собі захоплюючу атмосферу, стильне піксельне мистецтво та динамічний геймплей. Гра була розроблена американською інді-студією Heart Machine та випущена в 2016 році. У грі гравці відправляються в подорож у фантастичний світ,

який заселений небезпечними істотами та загадковими руїнами. Вони керують головним героєм, який має особливі здібності та володіє мечем і вогнепальною зброєю. Гравці змушені досліджувати світ, збирати ресурси, битися з ворогами та розв'язувати головоломки, щоб пройти крізь різноманітні локації та дізнатися таємниці цього світу. Її успіх демонструє потужність та гнучкість Gamemaker: Studio для створення складних ігор.

- **"Undertale" (2015)**

  "Undertale" - це інді-рольова гра, створена однією людиною, Тобі Фоксом за Studio і випущена в 2015 році. Ця гра набула широкої популярності завдяки своєму унікальному підходу до геймплею, непересічній емоційій глибині та виборів гравців, які впливають на хід сюжету. У "Undertale" гравці взаємодіють з різними персонажами у світі, де люди та монстри живуть урізні. Гравці можуть вибирати, чи битися з монстрами чи шукати мирні рішення, що впливають на хід сюжету та його закінчення. Гра відзначається яскравими персонажами, запаморочливими саундтреками та несподіваними поворотами сюжету, що створюють унікальний досвід для кожного гравця. "Undertale" виявила, що навіть з допомогою інструменту для незалежних розробників, яким є Gamemaker: Studio, можна створити ігри з непересічною глибиною і емоційною силою.

- **"Hotline Miami" (2012)**

  "Hotline Miami" - це екшн-гра, розроблена шведською інді-студією Dennaton Games та випущена в 2012 році. Гра відрізняється своєю швидкістю та безкомпромісністю, а також захоплюючим саундтреком і атмосферою. У "Hotline Miami" гравець виступає в ролі антигероя, який отримує таємні завдання та виконує їх, використовуючи насильство та інтенсивні бої. Гра відзначається стильною піксельною графікою, швидким геймплеєм та високим рівнем складності. Вона стала відомою своєю атмосферою 80-х років та захоплюючим геймплеєм, що дозволяє гравцям відчути себе в ролі героя екшн-фільму. "Hotline Miami" стала

відомою своїм непересічним геймплеєм та несподіваними поворотами сюжету.

GameMaker: Studio надає розробникам широкий набір інструментів та функціональності для створення 2D-ігор. Основні функції включають:

- Візуальний редактор рівнів: Рівні можна створювати за допомогою перетягування та розміщення об'єктів на сцені.
- Редагування спрайтів та об'єктів: GameMaker: Studio надає інструменти для створення та редагування спрайтів (зображень) та об'єктів гри.
- Мова програмування GML: Розробники можуть використовувати мову скриптування GML для програмування логіки гри та взаємодії об'єктів.
- Підтримка різних платформ: GameMaker: Studio дозволяє розробникам створювати ігри для різних платформ, включаючи Windows, macOS, iOS, Android та інші.
- Інтегрованість зі сторонніми сервісами: Рушій підтримує інтеграцію з іншими сервісами, такими як Steamworks для реалізації Steam-функцій, а також платформами монетизації, такими як AdMob та інші.

Переваги включають:

- Простий для вивчення інтерфейс, що робить його популярним серед початківців та недосвідчених розробників.
- Широкий набір функцій для створення різноманітних 2D-ігор.
- Підтримка мультиплатформенності, що дозволяє розробникам створювати ігри для різних пристроїв.

Недоліки включають:

- Обмежені можливості для створення складних 3D-ігор, оскільки рушій більш спеціалізований на 2D-іграх.
- Не така велика швидкість та продуктивність порівняно з деякими іншими графічними рушіями.

Окрім самого рушія, GameMaker: Studio має активну та дружню спільноту розробників. Форуми, чати та вебінари надають можливість обмінюватися

знаннями, досвідом та підтримкою з іншими членами спільноти, що дозволяє розробникам швидко розвиватися та вирішувати проблеми під час розробки своїх ігор.

GameMaker: Studio - це потужний інструмент для розробки 2D-ігор, який надає простий для вивчення інтерфейс та широкі можливості для створення різноманітних ігрових проєктів. Він здатний задовольнити потреби як початківців, так і досвідчених розробників, проте має свої обмеження у вигляді обмеженої підтримки 3D-графіки та продуктивності.

З огляду на постійне оновлення та підтримку з боку YoYo Games, GameMaker: Studio залишається важливим інструментом для розробників усіх рівнів. Перспективи розвитку цього рушія включають постійне удосконалення функціональності, підтримку новітніх технологій та розширення можливостей для створення інноваційних ігор [10].

# РОЗДІЛ 2. РЕАЛІЗАЦІЯ ПРОЄКТУ

## 2.1 Структура проєкту комп’ютерної гри

Даний проєкт був розроблений за чітким планом, структура якого поділяється на основних 4 етапи [9]:

1. **Аналіз та концептуалізація**: Початковий етап, під час якого проводиться аналіз концепції гри, визначаються цільові аудиторія та основні механіки гри. Тут також визначаються тематика, сюжет та інші ключові аспекти проєкту.
2. **Проєктування та розробка**: На цьому етапі розробляються детальні плани гри, включаючи створення графічних елементів, програмування геймплею та встановлення необхідних аудіо та візуальних ефектів.
3. **Тестування та виправлення помилок**: Після завершення розробки гри проводяться тестування для виявлення будь-яких помилок чи недоліків. Знайдені помилки виправляються, а ігра тестується знову, щоб переконатися, що вона працює належним чином.
4. **Випуск та реліз**: Останній етап - це випуск гри на ринок або публікація її у магазинах цифрового вмісту. Після випуску гра доступна для гравців, а розробник може продовжити підтримку, виправляти помилки та випускати оновлення.

Метою даного проєкту є розробка комп'ютерної гри у жанрі платформера, що ґрунтується на фундаментальних принципах та концепціях, характерних для класичних представників даного жанру. Обрано саме такий жанр з метою відтворення та вдосконалення найбільш характерних елементів, що роблять платформери популярними серед гравців протягом багатьох років та ігровими механіками, які викликають зацікавленість у широкому колі гравців. Відомо, що платформери часто пропонують гравцям інтенсивний геймплей, оснований на рефлексах та точності рухів. Таким чином, вибір жанру платформера для даного проєкту є обґрунтованим з точки зору його ігрових можливостей та потенціалу для створення високоякісної гри [37] .

На даний момент гра включає в себе три повноцінних рівні, розроблених з урахуванням різноманітних елементів геймдизайну та графічних аспектів. GameMaker дозволяє створювати багаторівневу гру за допомогою румів та системи шарів (рис. 2.1 ). Руми визначають простір, в якому відбувається гра, включаючи розміщення об'єктів, перешкод, гравця та інших елементів гри. Система шарів дозволяє організовувати руми в структури, що мають взаємодію та зміщення. Кожен рум може містити в собі **екземпляри** (instances), які використовуються для позначення конкретного об'єкта в грі, і **фони** (background), які являють собою фонові зображення у кімнаті [38].

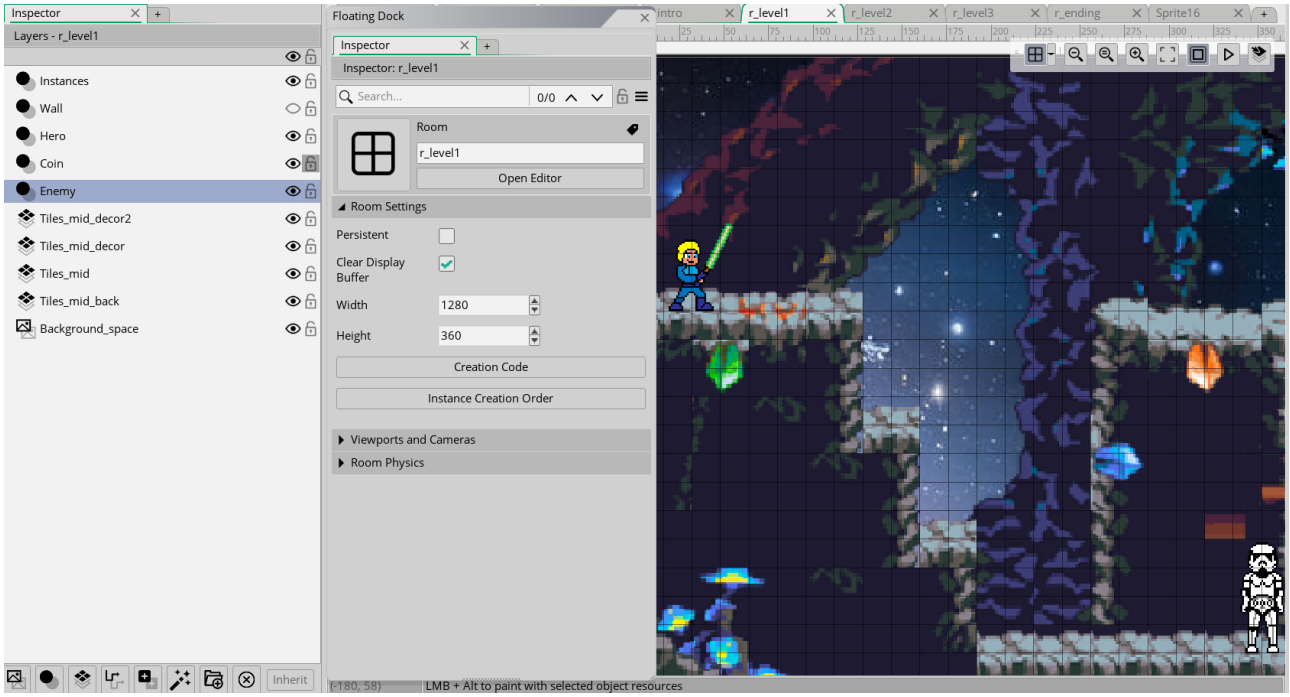



Рис. 2.1 Вікно налаштування кімнати в GameMaker: Studio.

Окрім спрайтів, до кожного руму можна додавати декорації за допомогою тайлових наборів. **Тайлові набори** у GameMaker: Studio - це потужний інструмент для створення ігрових рівнів і локацій з використанням тайлованих зображень. **Тайли** (Tiles) - це невеликі графічні елементи, які складаються з різних об'єктів, текстур або частин зображень. Вони використовуються для створення поверхонь та декорацій у відеоіграх. Тайлові набори дозволяють

розміщувати тайли на поверхні рівня, що дозволяє швидко і легко створювати складні ігрові області.

В основу графічного оформлення покладено використання піксель-арту. В грі, яка використовує піксель-арт, рушій GameMaker: Studio надає широкий набір інструментів для зручного реалізації цього підходу. Інструменти, такі як вбудовані засоби роботи з графікою, палітра кольорів, можливості створення та редагування спрайтів, тайлів, сприяють у створенні відповідного візуального ефекту (рис. 2.2). Крім того, функціональність, пов'язана з анімацією об'єктів і спрайтів, дозволяє реалізувати плавні рухомі елементи.

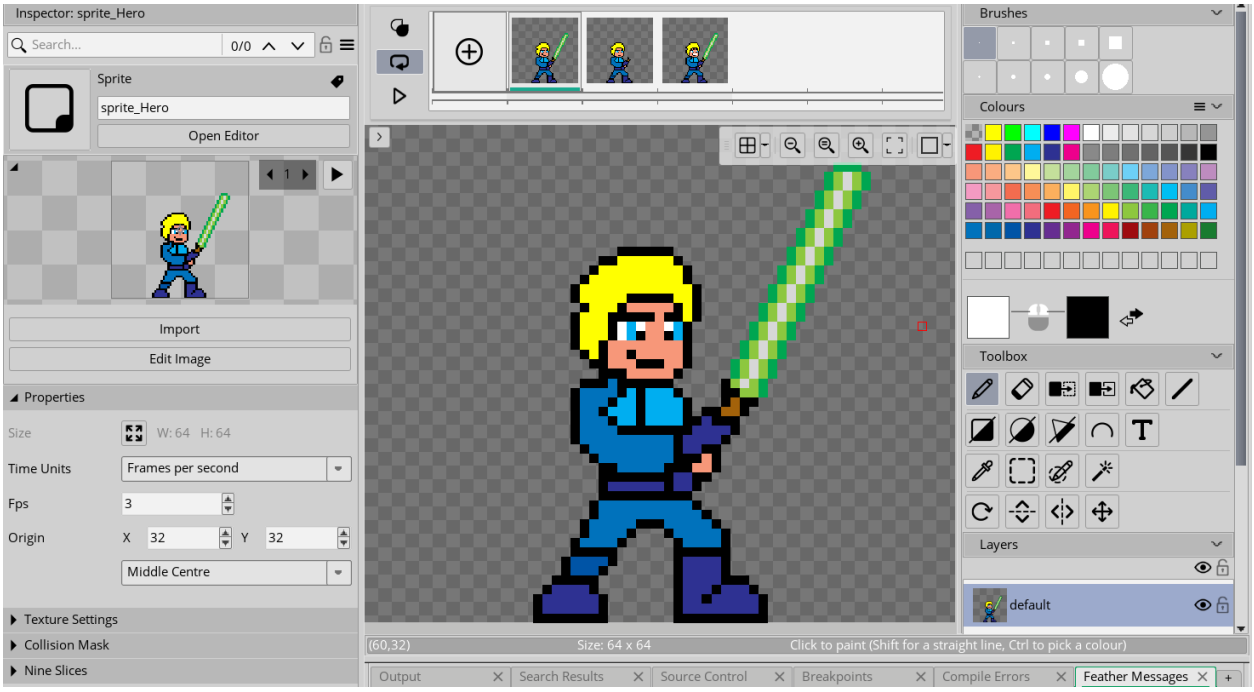



Рис. 2.2 Інструмент створення спрайтів у GameMaker: Studio.

Суттєвим аспектом даного проекту є здійснення оптимізації гри з метою забезпечення її працездатності на пристроях з обмеженими технічними характеристиками. Це включає в себе оптимізацію використання ресурсів процесора, пам'яті та графічної системи з метою забезпечення стабільної та плавної роботи гри на різних пристроях, від слабких до потужних. Однією з особливостей GameMaker є вбудована система оптимізації. Це включає в себе автоматичну оптимізацію рендерингу, управління пам'яттю та обробку подій, що

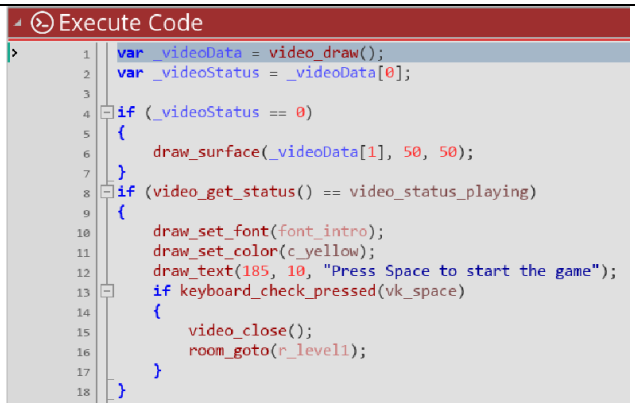
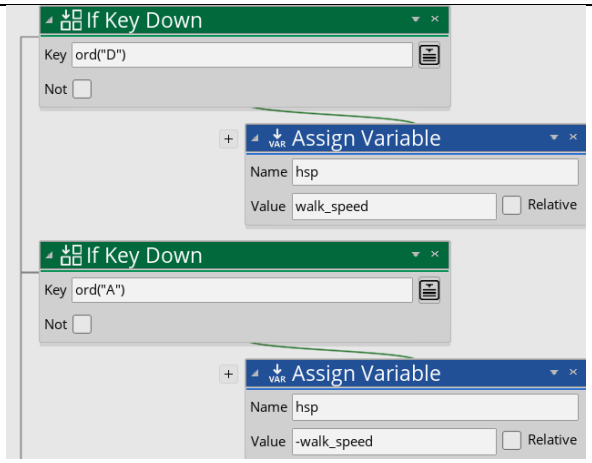
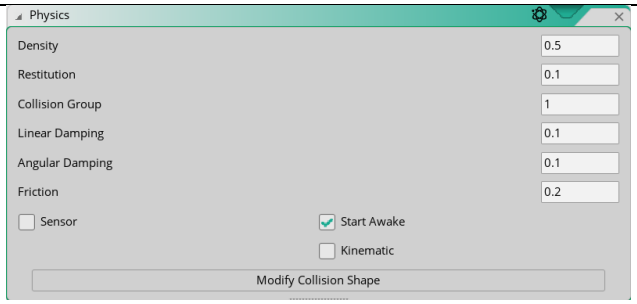
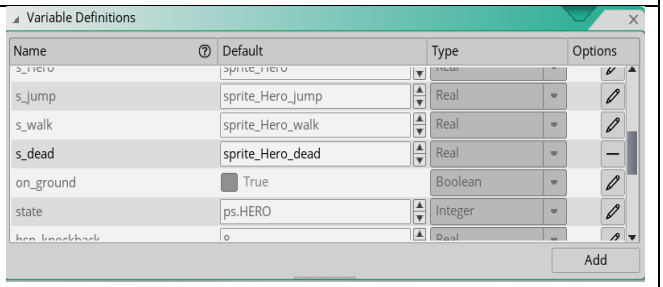
дозволяє знизити навантаження на апаратне забезпечення пристрою. GameMaker пропонує інструменти для тестування та аналізу продуктивності гри, що дозволяє розробникам виявляти та виправляти проблеми швидкодії під час розробки. Такий підхід дозволяє оптимізувати гру для різних пристроїв та платформ, забезпечуючи максимальну продуктивність та задоволення гравців [17].

У інтегрованому середовищі розробки GameMaker надається можливість створення алгоритмів за допомогою написання кодів на мові Game Maker Language (табл. 2.1а) або за допомогою з‘єднання готових запрограмованих блоків команд (табл. 2.2б). Автор даної роботи не обмежився одним з методів та використовував обидва. Після створення окремого об'єкту для кожного спрайта, розробник може налаштувати фізичні параметри (табл. 2.1в) кожного персонажа відповідно до вимог гри. Це включає в себе встановлення параметрів, таких як швидкість руху, сила гравітації, опір повітря, а також обмеження руху об'єкта в межах кожної кімнати. Додатково, можливо налаштувати реакцію персонажа на зіткнення з іншими об'єктами та оточуючим середовищем, таким як платформи, стіни або вороги. У вікні «Variable defenitions» користувач може створити змінні будь-якого типу та задати їм певні значення. Такі змінні можуть бути використані у створенні алгоритмів (табл. 2.1г). Події (events) (табл. 2.1д) - це програмні конструкції, що визначають реакцію об'єктів гри на різноманітні умови та події, які виникають під час гри. Керування активами (Asset Browse) (табл. 2.1є) – вбудований інструмент, що дозволяє вам керувати всіма активами (assets) у проекті. Це вікно дає зручний доступ до різних типів активів, таких як спрайти, звуки, фони, об'єкти, ресурси, шрифти та багато іншого [28].

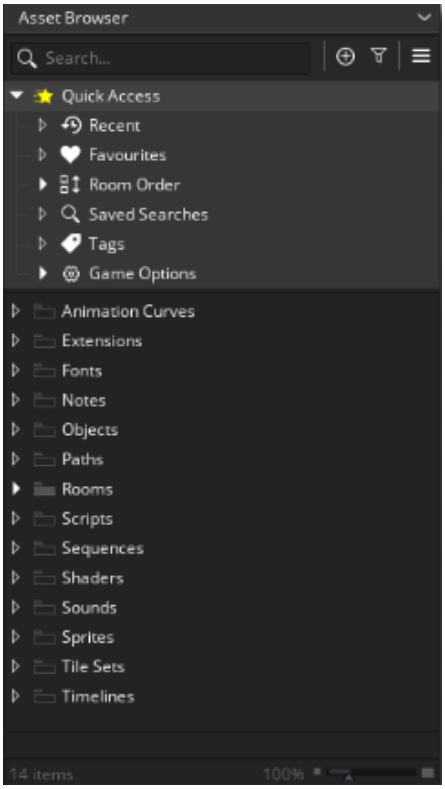
Ці події охоплюють різні аспекти геймплею, такі як натискання клавіш, зіткнення об'єктів, зміни стану гри тощо, визначають поведінку об'єктів у грі відповідно до виникаючих ситуацій.

*Таблиця 2.1*

**Об’єкти інтерфейсу GameMaker: Studio**

| № | Об’єкт інтерфейсу | Приклад |
|---|---|---|
| А) | Код на GameMaker Language. |  |
| Б) | Запрограмовані блоки команд. |  |
| В) | Фізичні властивості об’єктів |  |
| Г) | Приклад створених змінних для об’єкту. |  |

*Продовження таблиці 2.1*

| | | |
|---|---|---|
| Д) | Вікно подій для об’єкту. |  |
| Є) | Вікно керування активами |  |

У ролі головного героя у грі "Skywalker" виступає кіноперсонаж Люк Скайвокер, відомий завдяки своїй участі у фільмі "Зоряні війни". Люк Скайвокер є центральною постаттю в епічній сазі "Зоряні війни", створеній Джорджем Лукасом. Він символізує боротьбу за добро та протистояння силам темряви в галактиці. Люк подорожує складним шляхом від звичайного хлопця до героя, який має врятувати галактику від руйнування. Його боротьба за свободу та правду, а також його готовність до самопожертви стали ключовими рисами його характеру. Модель головного героя була розроблена на основі зовнішності актора Марка Хемілла, який зіграв роль Люка Скайвокера у фільмі. Вибір цього персонажу обґрунтовується бажанням привернути до гри фанатів фільму та забезпечити їм імерсивний досвід у віртуальному середовищі. Проєкт "Skywalker" розроблено з використанням тематики "Зоряних війн" з метою використання вже відомого та широко популярного фільму як основи для створення ігрового продукту.

## 2.2 Розробка графічних компонентів гри

Створення спрайтів є ключовим етапом у розробці графічних компонентів гри. Спрайти представляють собою графічні об'єкти, які можуть відображати головних персонажів, об'єкти у грі, а також фонові елементи. Процес створення спрайтів включає в себе наступні кроки [19]:

1. **Створення нового спрайту**: У редакторі GameMaker: Studio обираємо опцію створення нового спрайту та вказуємо його параметри, такі як розмір та колір.
2. **Намальювання основних контурів**: За допомогою інструментів редактора малюємо основні контури та форму головного героя. Можна використовувати базові кольори та форми для швидкого намальовування.
3. **Додавання деталей**: Поетапно додаємо деталі до спрайту, такі як обличчя, волосся, одяг тощо. Використовуючи різні інструменти, можна створювати складніші деталі та текстури.
4. **Колір та тінь**: Додаємо колір та тінь для підсилення реалістичності спрайту. Це може включати використання градієнтів, тіней та світінь.

5. **Оптимізація та експорт**: Перевіряємо спрайт на наявність зайвих деталей та оптимізуємо його, якщо необхідно. Після цього експортуємо спрайт у формат, придатний для використання у грі.

У проекті було створено декілька спрайтів, у тому числі анімації дій персонажів. У табл. 2.2 проілюстровано зображення трьох основних спрайтів, котрі будуть взаємодіяти між собою по ходу гри.

*Таблиця 2.2*

**Основні спрайти**

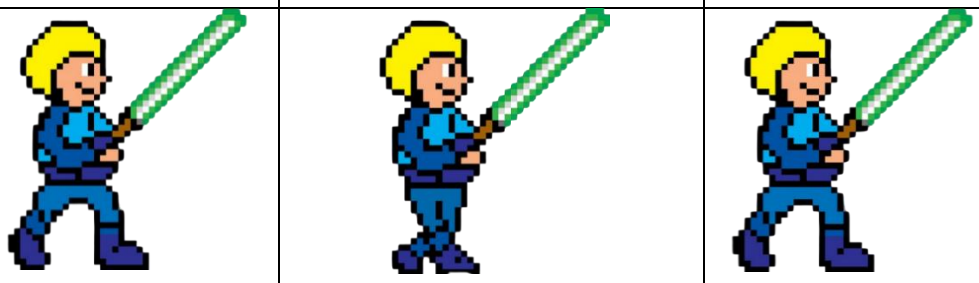
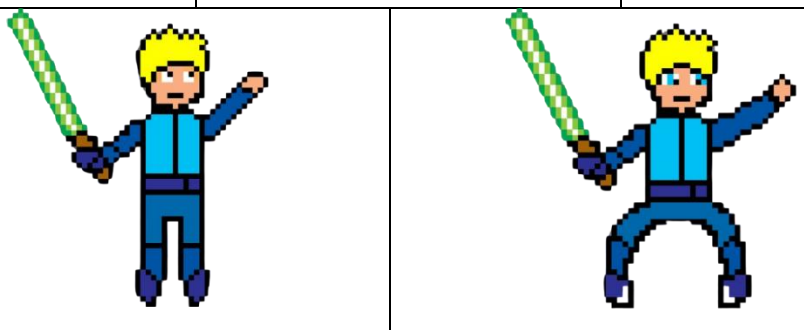
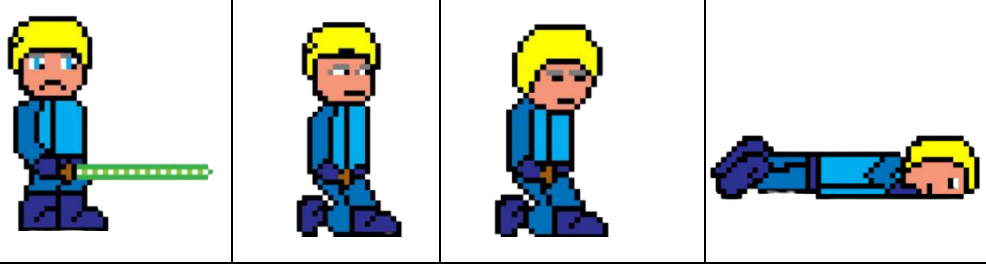
| № | Об’єкт гри | Спрайт об’єкту |
|---|---|---|
| А) | Головний герой |  |
| Б) | Ворог |  |
| В) | Монета |  |

Схожість між створеним спрайтом та кінематографічним героєм полягає в світлому кольорі волосся, тонкій та атлетичній статурі, блакитному кольорі очей, класичному одязі джедая: синя туніка, штани та головному атрибуту – світлового меча. Окрім спрайту головного героя було додано анімації простою, стрибку, бігу та смерті, використовуючи створення декількох зображень в спрайті. Для всіх спрайтів було використано розмір 64x64 пікселі.

В табл. 2.3 вказано всі анімовані дії пов'язані з головним героєм гри:

*Таблиця 2.3*

**Анімації головного героя**

| № | Дія | Анімація |
|---|---|---|
| А) | Простій |  |
| Б) | Біг |  |
| В) | Стрибок |  |
| Г) | Смерть |  |

Ще одним ключовим елементом гри є створення ворогів, оскільки вони створюють атмосферу боротьби та взаємодії для гравця та будуть завдавати шкоди головному герою. В ролі таких було створено спрайт зі зображенням у вигляді імперського штурмовика (табл. 2.2б), єдиного спрайту ворога на даному етапі гри. Аналогічним чином, як і з головним героєм, було додано анімовані дії даного персонажа, такі як ходьба та смерть.

До даного проєкту було додано внутрішньоігрову валюту – золоті монети (табл. 2.2в). Вони відіграють важливу роль у грі як грошова одиниця та елемент економічної системи, а їхні властивості можуть вплинути на геймплей та інтеракцію гравця з грою. Монета має круглий вигляд без будь-яких зображень на ній. При зіткненні монети з гравцем з'являється анімація зблиску, спрайт котрого створений за допомогою 8-ми послідовних зображень.

Окрім основних 3-х спрайтів, об'єкти яких будуть взаємодіяти між собою, було створено інші додаткові спрайти для кращого візуального розуміння та сприйняття гри.

Одним з таких є космічний корабель, який був запозичений з сайту безкоштовних спрайтів без авторських прав www.klipartz.com. Корабель досить великий, за висотою у 8 разів перевищує головного героя, має витягнуту, сигароподібну форму з загостреними кінцями, має синій колір з білими та чорними акцентами. На корпусі корабля можна розрізнити різні антени, датчики та інші елементи обладнання.

Спрайт кнопки головного меню має прямокутну форму з білим та сірим контуром. Кнопка має помірний розмір, щоб її було легко побачити та натиснути, не має жодних видимих ефектів, таких як тінь або світіння. Завдяки елегантному та простому дизайну вона добре вписується в загальний візуальний стиль гри.

Також було створено спрайт з використанням готової ілюстрації космосу та космічного корабля, яка була створена за допомогою штучного інтелекту на сайті www.deviantart.com. На цьому фото зображено космічний корабель, що летить над блакитною планетою. Корабель має довгастий, обтічний корпус з металевою поверхнею. Він оточений темним простором, де видно численні зірки

та інші планети. Зображення створює відчуття захоплення та благоговіння перед величчю космосу та таємницями, які він приховує. Деталізоване промальовування космічного корабля та планети робить зображення реалістичним, а темний фон підкреслює самотність мандрівника у безмежному просторі. Цей спрайт буде використовуватись як фонове зображення протягом усього ігрового процесу [14].

### 2.3 Дизайн рівнів і ігрового середовища

У грі було створено 6 румів включаючи головне меню, заставка, 3 рівні та фінальна сцена. Загальний розмір кожного окремого рівня рівний 1280 на 360 пікселів, а камера, яка слідкує за головним героєм та інші кімнати мають розмір 640 на 360 пікселів.

Для створення ігрового середовища проекту та покращення дизайну було використано безкоштовний тайловий набір з сайту www.openpixelproject.com під назвою «Cave» (рис. 2.3).

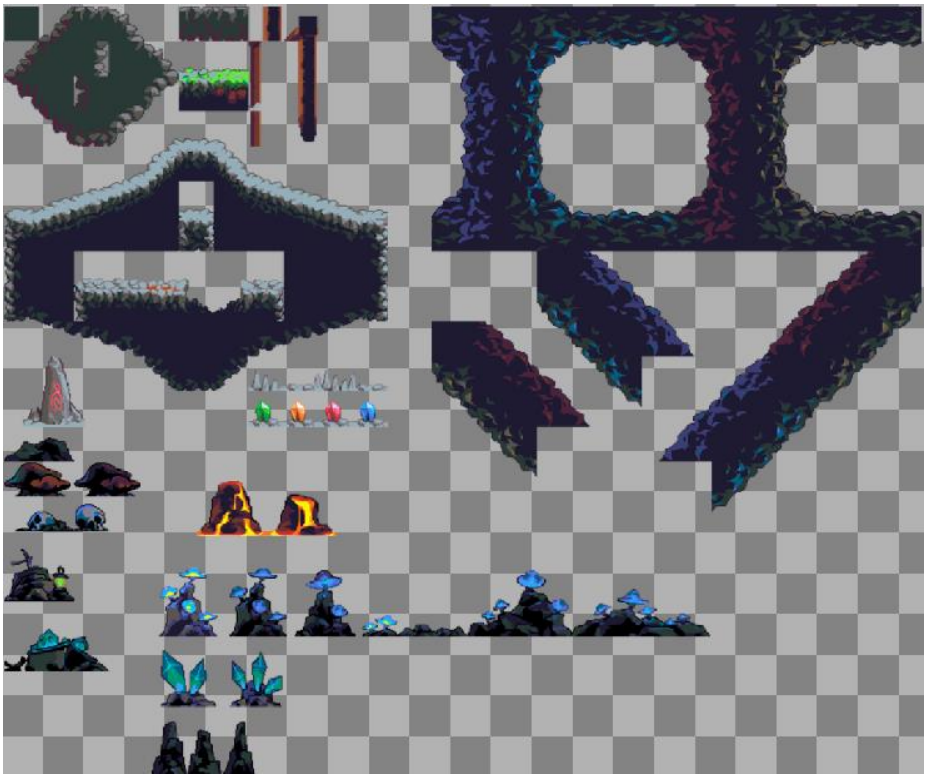


Рис. 2.3 Тайловий набір «Cave».

Було реалізовано створення спеціальних шрифтів, для використання їх для різних написів в ігровому процесі. За допомогою активу «Fonts» розроблено п’ять шрифтів: «font_coins», «font_menu», «font_intro», «font_floating_text» та «font_Skywalker».

Початкове вікно гри, що відображається при запуску програми, є головним меню в мінімалістичному стилі: має лаконічний дизайн з мінімальною кількістю елементів, яке надає гравцю можливість вибору різних опцій і функцій гри. Це інтерактивний інтерфейс, що включає в себе різноманітні елементи керування, такі як кнопки, пункти меню та інші візуальні елементи, які дозволяють користувачеві вибирати початок гри («Play), керування («Help») чи вихід з гри («Quit»), шрифти яких були створені аналогічним чином, використовуючи «Square721 Cn BT». Вони написані фіолетовим кольором на чорному тлі [6].

При обранні пункту меню "Play" користувачу надається можливість переглянути знамениту видозмінену сцену з початку фільму "Зоряні війни" переведена на українську мову у вигляді відео, яке триває 30 секунд., яка буде супроводжуватись відповідною мелодією. Проте, з лівого боку зверху можна спостерігати напис, в якому зазначається, що якщо користувач бажає пропустити цю сцену, він може це зробити, натиснувши клавішу "Пробіл", після чого починається гра. Відео було створено на сайті www.makermoon.com.

Після сцени гравець одразу переходить до першого рівня. В основу дизайну ігрового середовища проєкту на першому рівні гри було покладено зображення темної печери на вигаданій планеті Набу (табл. 2.4а). При завантаженні гри спостерігається головний персонаж, Люк Скайвокер (табл 2.2а), розташований на екрані, разом із платформами для переміщення та ворогами у формі імперських солдатів. У верхньому лівому куті гри відображається статистика зібраних монет, які можна здобути після ліквідації кожного ворога. Ці монети будуть служити ігровою валютою, котру можна буде скористатися в майбутніх версіях гри. Також зверху посередині екрану розміщений напис «Score:», який буде змінюватися на 10 одиниць при знищенні одного ворога. У моменти смерті гравця, потрапляння його в пастку чи в будь-

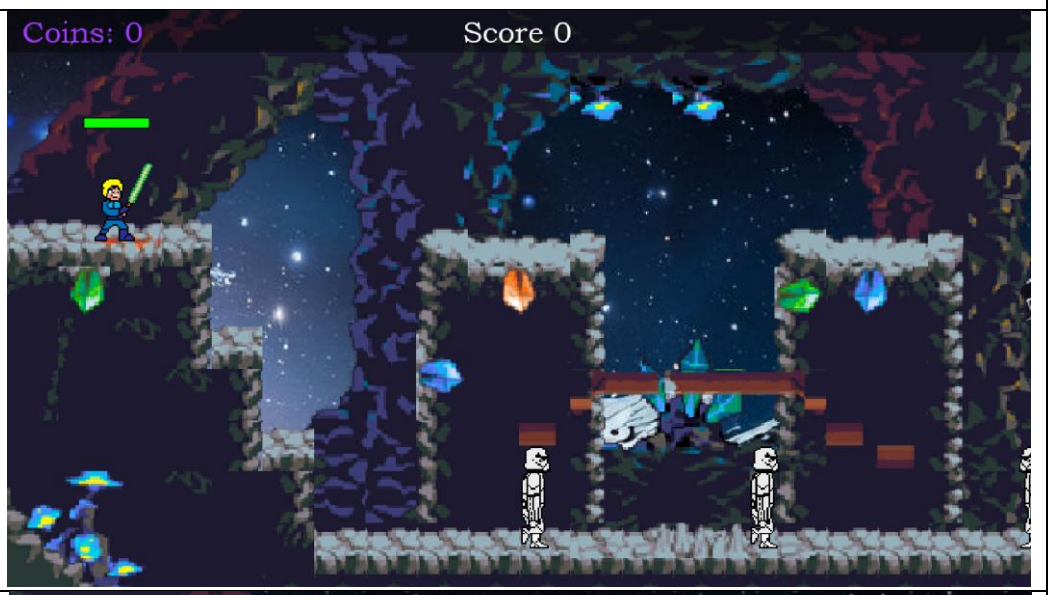
який момент за бажання, гравець може почати гру з головного меню за допомогою клавіші «R».

Дії в другому рівні відбуваються в тому ж середовищі, з тими ж елементами дизайну, але на відміну від першого тут гравець поступово вибирається з печери. На цьому рівні окрім ворогів знаходиться три ями, одна з яких наповнена лавою, падаючи в які, гравець мусить почати гру заново, так як вибратися з них неможливо.

Третій рівень не наповнений декораціями, ворогами та будь-якими небезпеками, на ній зображено лише вільний космос та космічний корабель в кінці рівня (табл 2.4в), зіткнувшись з яким, гравець переходить до фінальної сцени. Після цього всі елементи декорації зникають та на фоні ігрового поля з'являється напис "The end", що позначатиме завершення ігрового процесу. Для цього напису було використано неоновий колір та шрифт «Century». Після фінальної сцени буде  підкреслено успішне завершення гри та здобуті за період гри досягнення головним героєм у вигляді загальної суми усіх монет та очок рахунку. Через 5 секунд після закінчення гри гравець потрапляє у головне меню, де може розпочати гру знову.

*Таблиця 2.4*

**Ігрові рівні гри «Skywalker»**

| Номер рівня | Вигляд рівня під час гри |
|---|---|
| А |  |
| Б |  |
| В |  |

В кожній кімнаті в місцях зіткнення головного героя з поверхнею встановлено прихований об’єкт o_Wall, котрий відіграє роль твердої поверхні. Для нього було створено окремий шар в кожній кімнаті під назвою «Wall», як і для інших об’єктів розташованих в різних кімнатах.

Також гра наділена звуковим супроводом. В проєкті було використано чотири мелодії, одну з яких запозичено з ozzed.net – сайту з загальнодоступною 8-ми бітною музикою, а саме «The Final End» з альбому «8-bit Empire», яка використовується у третьому рівні та фінальній сцені. Увесь інший музичний супровід являється творінням знаменитого американського композитора Джона Вільямса, котрий написав музику для такої відомої картини як «Зоряні війни». Для головного меню було обрано пісню «The Birt of the Twins and Padme`s Destiny». В румі заставки грає відповідна мелодія з заставки фільму під назвою «Main Theme». Під час проходження першого та другого рівнів гравця супроводжує симфонія «Anakin vs. Obi-wan», яка створює героїчну атмосферу.

Окрім музики було додано звукові ефекти, короткі динамічні звуки, які використовуються для позначень подій у грі, а саме: приземлення монети на землю, підіймання монети головним героєм, приземлення головного героя на землю, смерть головного героя, нанесення шкоди герою, приземлення головного героя на ворога, стрибок головного героя, біг головного героя, смерть ворога. Звукові ефекти було створено за допомогою BXFR (див. рис. 2.9 ) - безкоштовного програмного інструменту для генерації звуків і звукових ефектів, який був спеціально розроблений для розвитку відеоігор.

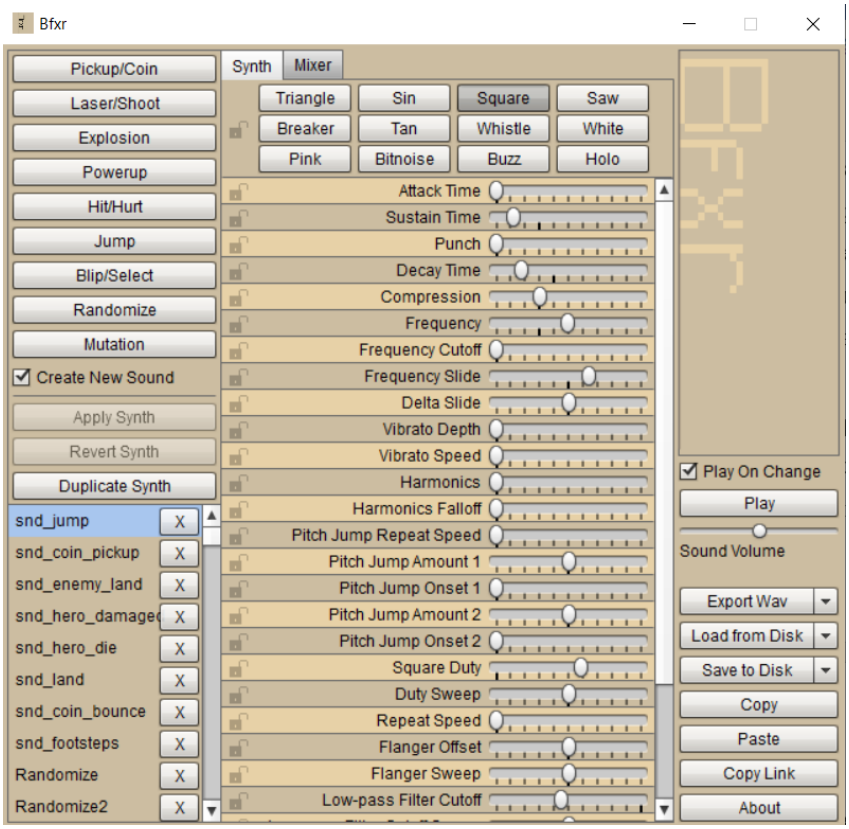



Рис. 2.4 Програма для створення звукових ефектів BXFR.

## 2.4 Алгоритми функціонування гри.

Спочатку було створено параметри переміщення та фізичні властивості головного героя. Об’єкт «o_entity» було створено, як батьківський для «o_Enemy» та «o_Hero», який має в собі змінні (Додаток 2), котрі задіюються в створенні алгоритмів для обох персонажів. Тому об’єкт головного героя має лише дві власні змінні - «walk_speed = 3», «jump_speed = 7» та деякі відредаговані змінні.

В події «Step» об'єкта головного героя створено деяку блок схему з використанням блоків команд, взятих з інструментарію. При натисканні клавіши «Ф» змінній «hsp» присвоюється значення змінної «walk_speed», а при натисканні «В» - значення «-walk_speed» (табл. 2.1б). Щоб герой виконував дію стрибка, поряд з блоками перевірки натискання клавіш для руху було створено ще один, що перевіряє чи натиснута клавіша «Пробіл» з присвоєнням змінній «vsp» значення «-jump_speed», який був поміщений в умовний блок для перевірки чи у повітрі знаходиться герой, щоб, знаходячись у повітрі, гравець міг

здійснити стрибок тільки один раз. Щоби об’єкт реагував на натискання, створено блок «Assign Variable», який відповідає за присвоєння змінних, в якому новоствореній змінній «x» присвоєно значення «hsp», при цьому поставивши галочку навпроти напису «relative» у даному блоці - цей параметр дозволяє перемістити елемент відносно його початкового положення до застосування властивості, залишаючи попереднє місце елемента на сторінці незмінним. Для того, щоб рух не був безкінечним, було створено один командний блок, де змінній «hsp» присвоєно значення «hsp*drag» [2].

Після події створення руху по горизонтальної поверхні (x = hsp), для зупинки руху при зіткненні героя з поверхнею по координаті X було створено цикл з перевіркою умови, що при торканні гравця з об’єктом o_Wall, він перестає рухатись, інакше – продовжує рух. Аналогічні дії з іншими змінними було зроблено для створення вертикального зіткнення.

Для реалізації автоматичних анімацій станів, щоб контролювати, що дозволено під час конкретних дій, у об’єкті «o_game» створено подію «Create», а в ній програмний код, в якому описано перелік станів, в яких може перебувати персонаж [5]:

```
//РУХИ ГЕРОЯ
enum ps {
	HERO,
	WALK,
	JUMP,
	KNOCKBACK,
	DEAD
}

//РУХИ ВОРОГА
enum es {
	ENEMY,
	WALK,
	DEAD
}
```

Паралельно в об’єкті «o_entity» створено змінну «ps.HERO» типу Integer. В об’єкт «o_Hero» у подію «Step» додано перемикач з кейсами «ps.Hero»,

«ps.JUMP», «ps.WALK» та «ps.KNOCKBACK», «ps.DEAD», які будуть використовуватись при взаємодії з ворогом. Увесь комплекс командних блоків переноситься до кожного кейсу. Для переміщення між кейсами створено умову з внутрішньою перевіркою, яка перевіряє, чи гравець стоїть на землі та змінна «hsp» = 0, то виконується кейс ps.HERO, інакше ps.WALK, якщо ні одна з умов не задовольняється, то здійснюється перехід до кейсу ps.JUMP.

Щоб забезпечити відображення бездіяльності головного героя, у комірку "Collision Mask" було внесено спрайт, котрий відображає анімацію простою «sprite_Hero». Після всіх перевірок створено ще один перемикач з кейсами, які будуть відповідати за виконання анімацій в конкретні моменти. В першому кейсі «ps.HERO» створено блок «Set Instance Variable», який дозволяє встановлювати значення змінної конкретного екземпляра об'єкта. В цьому кейсі вбудованій змінній «Sprite» присвоєно значення «sprite_Hero», що відповідає за відображення анімації стояння. Відповідні дії було виконано з кейсами ps.WALK з присвоєнням змінній «Sprite» значення «sprite_Hero_walk» та ps.JUMP з присвоєнням цій же змінній значення «sprite_Hero_jump», під яким створено перевірку: якщо «vsp» набуває значення менше 0.2, то відображається перший спрайт анімації стрибку під нульовим індексом, інакше - другий під індексом 1. Щоб спрайт персонажу розвертався у відповідний напрям руху, до кожного кейсу додано умовний блок з внутрішньої умовою : якщо «hsp» != 0, виконується умова «якщо «hsp» > 0, то перемінній «facing» присвоєно значення 1, інакше присвоєно значення -1. Для того, щоб це візуалізувати, створено подію «Draw», а в ній блок «Draw Sprite Transformed» з заданими параметрами. Цей блок відповідає за масштаб, поворот та інші зміни вигляду спрайту. Також до цієї події було створено умовний блок та «Draw Healthbar», який відповідає за відображення шкали життя головного героя [22].

Початок створення повноцінного об’єкту «o_Enemy», який буде відповідати за алгоритми дій ворога, було здійснено за допомогою дублювання «o_Hero» з його алгоритмами, зміною назви основного кейсу, та кейсу, що відповідає за анімацію з «ps.WALK» на «es.WALK», видаленням кейсів

«ps.JUMP», «ps.HERO» та блоків, які відповідають за управління, перевірку стрибка, перебування на поверхні. Для того, щоб ворог пересувався автоматично та виконував розворот, створено блок з присвоєнням змінній «hsp» значення «walk_speed * facing», який був поміщений в умову, в якій при значенні 0 змінної «hsp», «facing» множиться на -1, що відповідає за зміну напрямку [23].

Подія «Collisions» між «o_Hero» та «o_Enemy» в об’єкті «Hero» відповідає за смерть ворога при здійснені стрибку на нього головним героєм. В ній створено ряд умов, які перевіряють, чи ворог ще не мертвий, чи значення змінної «bbox_bottom» більша за «bbox_top +10» та «vsp > 0». «bbox_bottom» та «bbox_top» - це вбудовані змінні, які використовуються для отримання координат верхньої та нижньої меж області зіткнення об'єкта. Якщо всі умови виконуються, то змінній «vsp» присвоєно значення «-jump_speed», змінній «Image Index» - значення «0», «hsp» - «0» та «state» - «es.DEAD», після чого вони поміщуються в блок «Apply to» та застосовуються до «other» - це функція, яка дозволяє копіювати властивості або дії одного об'єкта і застосовувати їх до іншого. В подію «Step» об’єкта «o_Enemy» додано ще один кейс під назвою «es.Dead». Для того, щоб зупинити анімацію смерті ворога при відображенні останнього спрайту, в кейсі було створено умову: якщо «image_index»(індекс зображення) > «image_number(кількість зображень) – 1», то «image_speed(швидкість відтворення анімації) = 0». До цього ж кейсу було додано перевірку зіткнення з поверхнею по горизонталі та відображення анімацій ворога, до перемикача якого створено ще один кейс, який відповідає за анімацію смерті ворога.

В об’єкт o_Game додано ще одну подію під назвою «Key Pressed – R», а в ній створено один командний блок «Restart Game». Це вказує на те, що гра перезапускається при натисканні клавіши «R».

Для того, щоб вороги могли завдавати шкоди головному герою при зіткненні, в події «Collisions» було створено додатковий комплекс командних блоків. Перш за все, перед усім алгоритмом створено перевірку, чи головний герой ще живий наступним чином: якщо змінна «state» не дорівнює «ps.DEAD»,

то команди виконуються далі. Щоб вороги наносили шкоди герою саме при торканні ліворуч, чи праворуч, додано алгоритм, який буде виконуватись, за умови, що ворог ще живий. Цей алгоритм складається з перевірки, чи «state» не дорівнює «ps.KNOCKBACK», якщо умова задовольняється, то змінній «hsp» присвоюється значення «sign(x-other.x)*hsp_knockback», яке дає знати, що ворог знаходиться збоку та відштовхує героя. Також, щоб герой під час відштовхувався по діагоналі, до алгоритму додано присвоєння змінній «vsp» значення «-vsp_knockback». Після цього, змінній «hp» присвоєно «-1», що говорить про віднімання одного життя, яких герой має всього три.  В кінці додано блок зміни стану на відштовхування. В події «Step» об'єкту «o_Hero» створено додатковий кейс під назвою «ps.KNOCKBACK», який буде відтворювати анімацію стрибка під час відштовхування. Даний кейс складається з обчислення руху, перевірки знаходження на поверхні, алгоритмів зіткнення, створення анімацій та з перевірки стану, перед яким створено умову, в якій зазначено, що «on_ground» рівне «true» та «vsp» більше «0», для того, щоб не робити перевірку, перед тим, як герой приземлиться після відштовхування. До кейсу з анімації додано ще одну, яка буде відображати відштовхування. Вона складається складається з тих же алгоритмів, що і анімація стрибку, але без перевірки направлення.

Відображення смерті головного героя відбувається за допомогою перевірки умови: якщо «hp >= 0», то «state = ps.DEAD», яка додана до всіх головних кейсів події «Step» після перевірки стану. За допомогою копіювання кейсу смерті ворога, було додано анімацію смерті головного героя перейменуванням кейсу на «ps.DEAD», який було доповнено обчисленням руху, щоб зупиняти гравця, коли той помре та зміною перемикача з анімаціями на анімації головного героя, де було створено ще один кейс який буде відображати смерть героя.

До гри було створено трясіння екрану, яке відображається після стрибка героєм на голову ворога. В об'єкті «o_game» створено три наступні змінні, які будуть відповідати за трясіння типу даних «Boolean» - «screen_shake» з початковим значенням «False», його тривалість та зміщення типу «Integer» -

«shake_time», «shake_amount» зі значеннями «15» та «5» відповідно. Також у об’єкті створено подію «Alarm[0]», яка відіграє роль таймера, що використовується для виклику певної події через певний період часу, в ній створено присвоєння змінній «screen_shake» значення «false». Після цього, в події «Step» створено умови, за яких буде відбуватися трясіння: якщо «screen_shake» дорівнює «true» та «alarm[0]» >= «0», то спрацьовує зворотний відлік таймера зі значенням «shake_time». Після виконання умови рівності, або більшого значення змінної таймера ніж 0 створено два командних блоки під назвою «Set View Variable», які використовуються для управління положенням, розміром та масштабом перегляду гри на екрані. В першому блоці вказано переміщення по осі X значенням «random_range(-shake_amount, shake_amount)», що означає переміщення на довільну кількість пікселів від -5 до 5. В другому блоці створено аналогічні умови, але для осі Y. Щоб трясіння екрану було відображно, в події «Collisions» об’єкту головного героя, після виконання смерті ворога, створено блок «Apply to o_game», в якому змінній «screen_shake» присвоєно значення «true».

Для того, щоб задній фон нашої гри переміщався разом з головним героєм, але повільніше, в події «Step» перед усім готовим алгоритмом було створено наступний код:

```
/// переміщення заднього фону
var layer_id = layer_get_id("Background_space");
//швидкість переміщення заднього фону
var scroll_speed = 0.7;
//рух слою
layer_x(layer_id, camera_get_view_x(view_camera[0]) * scroll_speed);
```

Для монет, які будуть вилітати з ворогів після смерті, приземлятися та відскакувати ввід землі створено окремий об’єкт – «o_coin», який відображає вже готовий спрайт монети. Також створено об’єкт «o_item_par», який буде батьківським для об’єкту монети – це зроблено для того, щоб у наступних версіях гри було легше створювати предмети, які також будуть вилітати з ворогів. В

об’єкті було створено змінні «grav = 0.3», «hsp = 0», «vsp = 0», «drag = 0.96» та «bounce = 3» - усі типу «Real». У батьківському об’єкті створено подію «Step», а в ній визначення сили тяжіння та перевірку зіткнення з землею до якої було додано умову, за якої буде здійснюватись відскок: якщо змінна «bounce» більше «0», то змінній «hsp» присвоєно значення «hsp/2», а змінній «bounce» - «-1». Потім, в події «Collisions» в об’єкті «o_Hero», після присвоєння ворогу значення «es.DEAD», описано створення об’єкту «o_coin» в шарі «Coin» з розташуванням по осі X від -10 до 10 та по осі Y – bbox_top, тобто зверху над головою противника, яке буде виконуватись довільну кількість разів від 4 до 6 за допомогою блоку «Repeat» та вписаній у нього функції «irandom_range(4,6)». В об’єкті «o_coin» було відредаговано значення змінних «hsp» та «vsp» на random_range(-5,5) та random_range(-6,3) відповідно, що означає радіус на який може відлетіти монета.

Створено об’єкт «o_coin_pickup», який буде грати роль відображення зблиску після зіткнення головного героя з монетою. В ньому створено подію «Create» з командним блоком «Set Instance Variable» з заданими параметрами у вигляді змінної «Image Rotation» зі значенням «random(360)», що здійснює обертання анімації зблиску по колу від 0 до 360 градусів. В події «Animation End» створено блок «Destroy Instance», щоб анімація була закінчена після одного програвання.

Щоб гравець міг підбирати монети шляхом зіткнення з ними, в об’єкті «o_coin» було створено подію «Collision» з об’єктом «o_Hero». В ній змінній «global.coin» (глобальній змінній, яку було створено в об’єкті «o_game») присвоєно значення 1. Потім створено ще один блок під назвою «Change Instance» зі значенням «o_coin_pickup», для того, щоб спрайт монети змінювався на анімацію зблиску. Цей алгоритм було поміщено під умову, що змінна «bounce» менше «2».

Для створення очок, які гравець заробляє після знищення кожного ворога, було виконано окремий алгоритм. В об’єкті «o_Enemy» створено змінну «points» зі значенням 10 типу даних «Real». Відразу після присвоєння змінній «state»

значення «es.DEAD» в події «Collision» об’єкта «o_Hero» створено призначення змінній «score» значення «points», тобто, за кожного ворога головний герой отримує 10 очок. Наступним блоком було створення шару «o_floating_text»

Було реалізовано графічний інтерфейс кількості зібраних монет шляхом відображення їх на ігровому полі. Для цього в об’єкті «o_game» створено подію «Draw», а в ній алгоритм: «Set Font», в якому обрано шрифт «font_coins», «Set Font Colour» - в ньому обрано колір шрифту «$FFFF3AA3» та блок «Draw Value», в якому визначено назву напису («Coins:»), значення яке він відображає («global.coins») та координати (X = 20, Y = 8). За допомогою блоку «Set Text Alignment» було встановлено вирівнювання тексту по лівій стороні зверху.

Для відображення очок, які гравець отримує за кожного знищеного ворога було виконано той самий алгоритм, що і для монет зі зміненими параметрами в блоках, а саме: у блоці «Draw Value» назву напису змінено на «Score:», а значення на «score», колір змінено на «$FFFFFFFF», розташування встановлено на верх середини екрану, а шрифт лишився той самий.

Візуальний фідбек у вигляді напису після піднімання монет та зароблених очок героєм був створений наступним чином. Створено об’єкт «o_floating_text» та змінні в ньому, які можна переглянути в Додатку 2. В ньому створено дві події – «Step» та «Draw». У «Draw» описано алгоритм створення візуального відображення іншим шляхом, відмінним від способу створення візуалізації кількості монет та очок. Призначено напис зі значенням змінної «txt», який являє собою змінну типу «Real» зі значенням «0», шрифтом «font_floating_text», визначенням кольору вписуванням готової змінної «col», координати було визначено місцем перебування монети. Наступним блоком, після призначення змінній «score» значення «points» в об’єкті «o_Hero», було створення шару «o_floating_text» з визначеними координатами X = 0, Y = «bbox_top» та створеною в блоці тимчасової змінної «inst», яка використовується для вказівки цільового об'єкта, до якого буде додано новий екземпляр, а його координати будуть відносно цільового об'єкта. Під цим блоком створено «Apply to inst» з призначенням змінній «txt» значення «”+” + string(other.points)», що буде

відображати напис кількості зароблених очок (+10). В події «Collisions» об’єкту «o_coin» після всього алгоритму створено відображення об’єкту «o_floating_text» в шарі «Instances» з координатами X = 0 та Y = «bbox_top» зі створенням тимчасової змінної «inst», до якої застосовується призначення змінній «txt» значення «+1» та змінній «col» значення «c_yellow»: увесь цей алгоритм буде створювати напис «+1» жовтого кольору після реалізації анімації зблиску монети [27].

Останнім кроком для реалізації повноцінної гри було створено переміщення головного героя між кімнатами та його розміщення. Для спрайта у вигляді напівпрозорого синього кольору блоку створено об’єкт «o_trigger_transition», який розміщено на початку та кінці кожної кімнати та відіграє роль телепорту з однієї кімнати в наступну. В об’єкті «o_game», події «Create» до написаного там коду додано командний блок в якому встановлено координати, де з’являється головний герой та в яку сторону він буде повернений після запуску гри (global.hero_start_x = 32, global.hero_start_y = 10 та global.hero_facing = 1). В об’єкті «o_Hero» створено подію «Room Start», а в ній блок з призначенням змінним «x», «y» та «facing» значень «global.hero_start_x», «global.hero_start_y» та «global.hero_facing». Після цього гравець буде розпочинати гру  в першій кімнаті за конкретними координатами. Перейшовши в кімнату, два рази натиснувши на об’єкт «o_trigger_transition» є можливість написати для нього алгоритм, який буде працювати тільки з даним екземпляром об’єкту. В першій кімнаті розташовано один такий об’єкт, який знаходиться в кінці: для нього створено блок призначення змінним, які означають координати розміщення героя та наступну кімнату при зіткненні об’єктів. В другій кімнаті для цього ж об’єкту іншого екземпляра створено такий же блок але заданими координатами, на яких буде розташовуватись гравець при поверненні в першу кімнату. Аналогічні дії було виконано для переходів з інших кімнат одна в одну. Для того, щоб глобальні змінні розташування головного героя змінювались відносно кімнат в які він переходить, в «o_trigger_transition» створено подію зіткнення з «o_Hero», в якій змінним розташування задано значення «target_x»,

«target_y», «target_facing», а під ним блок переходу в наступну кімнату зі значенням «target_room», де префікс «target_» використовується для позначення різних елементів, що вказують на певний об'єкт або ціль у грі [40].

# РОЗДІЛ 3. АНАЛІЗ ОТРИМАНОГО РЕЗУЛЬТАТУ

## 3.1 Тестування програмного забезпечення

Тестування програмного забезпечення є необхідною складовою в процесі розробки для забезпечення його якості та стабільності. Цей етап передбачає виявлення помилок, перевірку відповідності функціональних вимог та забезпечення високої якості продукту перед його випуском. Цей процес включає в себе ряд дій та методів, спрямованих на виявлення та усунення помилок, перевірку відповідності вимог та забезпечення коректної роботи програмного забезпечення в різних умовах експлуатації [35].

- Організації тестових сценаріїв

Одним з важливих аспектів тестування є розробка тестових сценаріїв, які охоплюють різні аспекти функціональності програми чи гри. Ці сценарії визначають послідовність кроків, які виконує користувач для перевірки правильності роботи програми та виявлення можливих помилок.

- Використання інструментів автоматизованого тестування

Для покращення ефективності тестування розробники використовують інструменти автоматизованого тестування. Ці інструменти дозволяють автоматизувати виконання тестових сценаріїв, що дозволяє знизити час, необхідний для тестування, та покращити його якість.

- Відлагодження

Відлагодження (debugging) є важливою частиною процесу тестування. Під час відлагодження розробники виявляють та виправляють помилки в програмному коді, що допомагає забезпечити стабільну та безперебійну роботу програми.

Більшість тестів було виконано паралельно з розробкою алгоритмів функціонування гри, і вони завершилися успішно. Після завершення імплементації кодованих алгоритмів гри був проведений останній тест, що включав в себе низку найважливіших перевірок (табл. 3.1).

*Таблиця 3.1*

**Тестування окремих компонентів гри**

| **Компонент гри** | **Вихідні дані** | **Очікуваний результат по кожному кроку** | **Фактичний результат** |
|---|---|---|---|
| Зіткнення з ворогом зверху. | 1.Виконати стрибок головним героєм на голову ворога. | 1.Смерть ворога, шляхом відповідної анімації, випаданням з нього монет. | Успішно |
| Реакція при зіткненні з ворогом збоку. | 1.Наблизитися головним героєм до ворога.<br>2. Очікувати зіткнення.<br>3.Спостерігати результат зіткнення. | 1. Приземлення на поверхню до якої дістається ворог.<br>2. Відштовхування гравця в протилежний бік направлення.<br>3. Зменшення шкали здоров’я з кожним зіткненням та в результаті смерті головного героя. | Успішно |
| Зміна інтерфейсних даних головного героя. | 1.Вбити ворога.<br>2.Підібрати монету, що випала з ворога. | 1. Анімація вилітаючого тексту «+10» на місці вбивства та зміна інтерфейсу відображення очок на 10 одиниць.<br>2. Анімація вилітаючого тексту «+1» та зміна інтерфейсу відображення зібраних монет на 1 одиницю. | Успішно |

## 3.2 Підготовка до публікації проєкту на Kickstarter

Основна мета кампаній на Kickstarter полягає у зборі фінансових ресурсів для реалізації проєкту. Однак слід мати на увазі, що проведення кампанії також пов'язане з витратами, і відповідно до умов платформи, може бути утримано до 30% від зібраних коштів на комісійні та інші витрати. Другорядна, але значуща мета кампаній полягає у формуванні спільноти навколо проєкту. Kickstarter має за мету не лише залучити фінансову підтримку від гравців, а й створити активну спільноту прихильників, які підтримають та активно рекламують проєкт, діляться своїми враженнями та надають зворотний зв'язок. Третій аспект стосується підвищення інтересу до проєкту за допомогою кампанії на Kickstarter. Це дозволяє використовувати платформу як ефективний маркетинговий інструмент, забезпечуючи широкий охоплення аудиторії, можливість співпраці з впливовими особами та засобами масової інформації, а також створення загального хайпу навколо проєкту [33].

Необхідно чітко визначити ціль, щоб керувати очікуваннями учасників кампанії. Кампанія, спрямована виключно на збір грошей, відрізняється від кампанії, спрямованої на популяризацію гри. Важливо визначити основну ціль саме перед початком кампанії, оскільки це визначатиме стратегію та тактику, які будуть використовуватися в процесі. Підготовка кампанії передбачає ретельне визначення цілей та їхнє узгодження з потребами та очікуваннями аудиторії. Такий підхід дозволяє забезпечити успішний старт та максимально використовувати ресурси для досягнення поставлених завдань [1, 34].

Етапи публікації проєкту на Kickstarter можна умовно розділити на три основні фази:

1. Підготовка (6-3 місяці до початку кампанії):
    - Розробка концепції та визначення цілей кампанії.
    - Підготовка контенту для презентації проєкту.
    - Визначення стратегії маркетингу та просування.
2. Сама кампанія (30 днів):
    - Публікація проєкту на Kickstarter.

- Активна просування та реклама серед цільової аудиторії.
- Взаємодія з потенційними спонсорами та відповіді на їхні запитання.
- Моніторинг та аналіз результатів кампанії.

3. Посткампанія (30-60 днів):
   - Збір коштів та визначення наступних кроків для реалізації проєкту.
   - Подяки спонсорам та підтримка спільноти.
   - Звіт про результати кампанії та плани на майбутнє.

Після етапу підготовки було розпочато запуск кампанії. Перш за все, зареєстровано аккаунт на сайті kickstarter.com. Після цього створено проєкт в категорії «Video Games» та «Graphic Design». В переліку країн місцезнаходження виробника для отримання коштів з проєкту, обрано локацію «Київ, Україна». Взявши до уваги правила, яких повинен дотримуватися кожен проєкт Kickstarter, було розпочато етап підготовки до публікування.

Спочатку описано основні відомості про проєкт. Обрано заголовок назви проєкту, його короткий опис. Додано зображення проєкту, яке чітко його відображає, добре виглядає за будь яких розмірів. Згодом це зображення з’явиться на сторінці проєкту, на веб-сайті Kickstarter. Зазначено відеоматеріал, що демонструє виправдання проекту, в якому розказано про цілі збору коштів, методику їх здійснення, особисту ідентичність автора та мотивацію створення проєкту. Для реалізації проєкту та отримання винагороди вказано більш занижену суму в 70$, ніж яка треба насправді, для того, щоб швидше її досягти. Датою публікації проєкту назначено 1 червня 2024 року, для того, щоб був час на вдосконалення проєкту, отримання рекомендацій від Kickstarter щодо дій, обробка яких може займати до декількох днів. В комірці «Тривалість кампанії» вказано мінімальну кількість днів – 1, так як проєкт вже завершений.

Наступним кроком був опис історії створення проєкту, розповідь про самого розробника, для чого він збирає кошти, як про це дбає та як планує це втілити в реальність. Після цього описано ризики та випробування з якими може зіткнутися проєкт, до яких входять наступні:

1. Технічна складність

2. Управління ресурсами

3. Ринкова конкуренція

Створено посилання на сторінку розробника на веб-сайті Kickstarter (https://www.kickstarter.com/profile/tzadorozhnyi). Після здійснення всіх кроків для публікації, проєкт було створено (рис. 3.1) та відправлено на перевірку, чи відповідає він правилам та вимогам платформи, яка триває від 1 до 3 робочих днів.

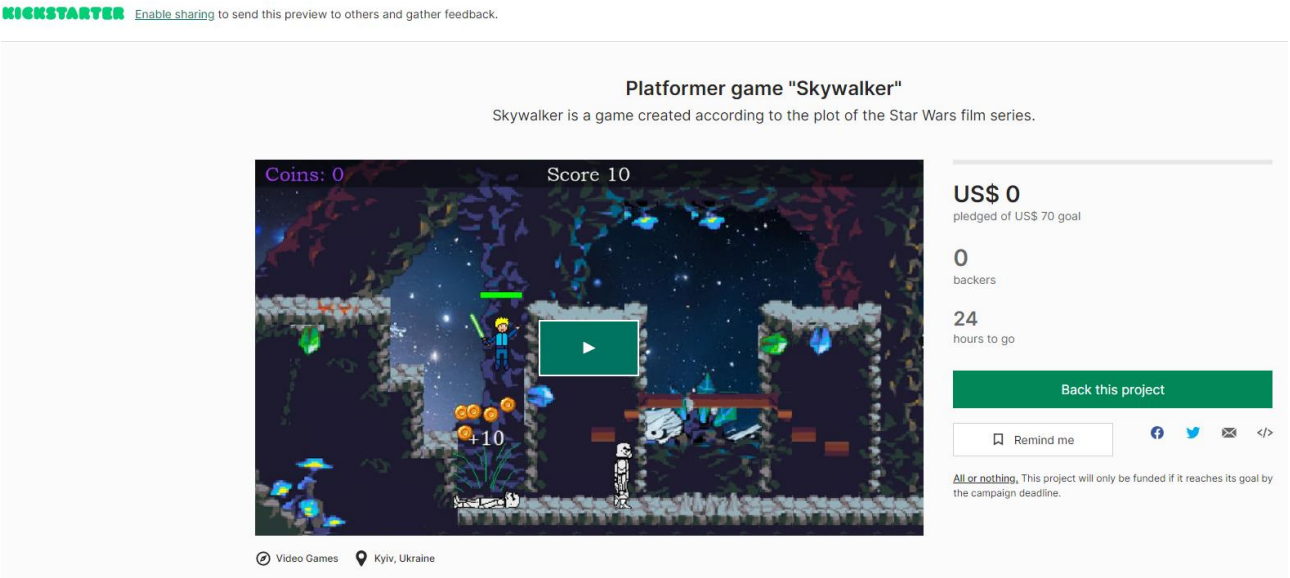



Рис. 3.1 Створений проєкт на платформі Kickstarter.

Для наступного кроку було виконано просування проєкту за допомогою реклами серед цільової аудиторії. Під час активної просування та реклами серед цільової аудиторії використовувалися різноманітні стратегії та інструменти для максимального привертання уваги до проєкту на Kickstarter. Спочатку була розроблена комплексна маркетингова стратегія. Далі, використовувалася таргетована реклама на платформах соціальних мереж та співпраця з інфлюенсерами та блогерами з великою аудиторією. Для персоналізованої комунікації використовувалася електронна пошта, спрямована на підписників зі спеціально підготовлених списків.

## 3.3 Вдосконалення і технічна підтримка проєкту

Під час розробки проєкту проводилося регулярне переглядання алгоритмів роботи гри з метою виявлення та виправлення можливих проблем з продуктивністю та ефективністю. Підходячи до кожного фрагмента коду з уважністю та систематичністю, аналізувалася його структура та робочий процес з метою ідентифікації можливих вузьких місць або ефективнішого способу реалізації функціональності. У разі виявлення проблем або можливих покращень, все ретельно документувалось з метою забезпечення належної зрозумілості та підтримки майбутнього розвитку проєкту.

Наразі планується створення скриптів для кращого розуміння коду, облегшення роботи з ним. Також, є думка замінити кодуванням всі сукупності командних блоків.

Під час розвитку проєкту використовувалися нові можливості та технології для поліпшення графічного візуалу, звукового супроводу та інших ресурсів гри. Планується розширення гри "Skywalker" шляхом додавання нових елементів геймплею, рівнів, персонажів та можливостей. Один із можливих напрямків розвитку включає у себе вдосконалення штучного інтелекту ворогів для більш складних та цікавих сценаріїв боїв. Крім того, планується додавання нових рівнів з унікальними викликами та підготовка нових зон для дослідження гравцями. Також розглядається можливість впровадження нових режимів гри, таких як мультиплеєр або режими виживання, що розширило б геймплей та збільшило б різноманіття взаємодії з грою. Вдосконалення графічної та звукової складових гри також входить у список планів.

Наразі, у користувачів гри є доступ до соціальних мереж та gmail-адреси розробника в його особистому кабінеті на платформі Kickstarter. У майбутньому планується підвищити якість технічної підтримки. Буде розвинено нові канали зв'язку для користувачів, зокрема, мобільні додатки та онлайн-чати, щоб забезпечити ще швидший та зручніший доступ до допомоги. Крім того, буде вдосконалено систему відстеження та вирішення проблем для більш ефективного виявлення та виправлення помилок. Крім того, буде враховано

фідбек користувачів та їхні побажання під час розробки оновлень з метою забезпечення максимального задоволення від гри для всіх гравців.

# ВИСНОВКИ

У рамках проведеного дослідження було виявлено, що створення комп'ютерних ігор з використанням рушія GameMaker: Studio відкриває широкі можливості для реалізації творчих ідей та концепцій. Проєкт "Skywalker" став ілюстративним прикладом успішного використання цього інструменту для розробки платформеру на основі популярної тематики кіносаги "Зоряні війни".

У роботі було детально розглянуто історію розвитку комп’ютерних ігор створених за допомогою різних рушіїв, які також було детально проаналізовано та обрано той, що найкраще підходить для розробки гри-платформера «Skywalker». Також, було описано сам процес проєктування та реалізації гри, включаючи вибір тематики, створення структури проєкту, реалізацію алгоритмів функціонування гри та технічну підтримку проєкту. Для створення гри були використані різноманітні інструменти та можливості, які надає рушій GameMaker: Studio, такі як вбудований редактор, події, змінні, анімація та багато іншого.

Проєкт було успішно запущено на платформі Kickstarter для залучення фінансування з метою подальшого розвитку та вдосконалення гри. Аналіз отриманих результатів дозволяє припустити, що продукт має потенціал для успішної реалізації та може здобути значну увагу та підтримку фанатів кіносаги та геймерської спільноти в цілому.

Попри успішність реалізації основної частини проєкту, було виявлено кілька недоліків. Серед них варто виділити:

1. **Обмеженість графічних можливостей**: Використання піксель-арту, хоч і відповідає ретро-стилістиці, може не задовольнити вимоги сучасних гравців до високоякісної графіки.
2. **Оптимізація продуктивності**: Деякі аспекти коду потребують подальшої оптимізації для забезпечення стабільної роботи гри на різних платформах і пристроях.

3. **Різноманітність контенту**: Наявність лише трьох рівнів може обмежити ігровий досвід користувачів, тому необхідно розширити кількість і різноманітність рівнів та ігрових ситуацій.

Перспективи усунення зазначених недоліків включають:

1. **Покращення графіки**: Використання більш сучасних графічних технологій та підвищення якості візуальних ефектів для задоволення потреб сучасних гравців.

2. **Оптимізація продуктивності**: Регулярний перегляд та оптимізація коду для покращення продуктивності та забезпечення плавного ігрового процесу на різних пристроях.

3. **Розширення контенту**: Створення додаткових рівнів та різноманітних ігрових ситуацій для забезпечення більш насиченого та тривалого ігрового досвіду.

Результати дослідженння були опубліковані в матеріалах Всеукраїнської конференції молодих вчених «Математичні методи та цифрові технології в економіці». У роботі проаналізовано можливості платформи GameMaker, виявлено його переваги та недоліки.

Загалом, проєкт "Skywalker" демонструє потенціал GameMaker: Studio як ефективного інструменту для розробки комп'ютерних ігор. Виявлені недоліки та шляхи їх усунення надають можливість подальшого вдосконалення гри та підвищення її конкурентоспроможності на ринку комп'ютерних ігор.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гембіцький М. Як запустити кампанію на Kickstarter для своєї гри. *Gamedev*. URL: https://gamedev.dou.ua/blogs/launching-a-kickstarter-campaign/. (date of access: 28.03.2024).

2. YoYo Games Ltd. Tutorials. *GameMaker*. URL: https://gamemaker.io/en/tutorials. (date of access: 12.03.2024).

3. Sarah Zulkiflee. The evolution of FIFA games [1993 – 2022] | esports.gg. *esports.gg*. URL: https://esports.gg/guides/gaming/the-evolution-of-fifa-games-1993-2022/. (date of access: 25.02.2024).

4. Contributors to Wikimedia projects. Commodore 64 - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Commodore_64. (date of access: 02.03.2024).

5. Shaun Spalding. GameMaker Studio 2: Complete Platformer Tutorial (Part 1: Basics), 2017. *YouTube*. URL: https://www.youtube.com/watch?v=izNXbMdu348. (date of access: 12.03.2024).

6. GameMaker. How To Create A Main Menu In GameMaker, 2022. *YouTube*. URL: https://www.youtube.com/watch?v=Us5GSddVedY. (date of access: 24.03.2024).

7. Shaun Spalding. How to play video files in a GameMaker game, 2022. *YouTube*. URL: https://www.youtube.com/watch?v=0S8WsAJvTjE. (date of access: 29.03.2024).

8. Вибір двигуна для гри. *AVADA-MEDIA*. URL: https://avada-media.ua/ua/services/vybor-dvizhka-dla-igry/. (date of access: 02.03.2024).

9. Авраменко О. М. Запорожець О. В. Іванов О. О. Проектування комп'ютерних ігор: теорія та практика. Київ: НТУУ "КПІ", 2018.

10. YoYo Games Ltd. Help Centre. *GameMaker*. URL: https://help.gamemaker.io/hc/en-us. (date of access: 10.03.2024).

11. Game Development Resources, Case Studies & Articles | Unity. *Unity*. URL: https://unity.com/resources?filters=game%20creation. (date of access: 02.03.2024).

12. We make the engine. You make it Unreal. *Unreal Engine*. URL: https://www.unrealengine.com/en-US. (date of access: 02.03.2024).

13. Godot Engine - Free and open source 2D and 3D game engine. *Godot Engine*. URL: https://godotengine.org/. (date of access: 04.03.2024).

14. Tutorialspoint. *Online Tutorials, Courses, and eBooks Library | Tutorialspoint*. URL: https://www.tutorialspoint.com/gamemaker/index.htm. (date of access: 15.03.2024).

15. Newest 'gamemaker' Questions. *Stack Overflow*. URL: https://stackoverflow.com/questions/tagged/gamemaker. (date of access: 16.03.2024).

16. Історія геймдеву XX століття: від Spacewar! до Sonic the Hedgehog. SKVOT / СКВОТ – онлайн-курси про рекламу, кіно та мистецтво | SKVOT. URL: https://skvot.io/uk/blog/istoriya-geymdevu-hh-stolittya-vid-spacewar-do-sonic-the-hedgehog. (date of access: 26.02.2024).

17. GameMaker Studio 2, створюйте власні відеоігри з Ubuntu. Ubunlog. URL: https://ubunlog.com/uk/gamemaker-studio-2-виробляє-власні-відеоігри/. (date of access: 21.03.2024).

18. Програми для створення ігор. Корисні поради та інструкції про операційні системи, програми та «залізну» начинку. URL: https://daad.org.ua/4741-programi-dlya-stvorennya-igor.htm. (date of access: 05.03.2024).

19. Полетаєва Г. Н. Етапи розвитку художньо-виразних складових комп'ютерних. Art and Design. 2020. № 4. С. 91–98. URL: https://doi.org/10.30857/2617-0272.2019.4.8. (date of access: 28.02.2024).

20. Tyers B. Platform Game. Practical GameMaker Projects. Berkeley, CA, 2018. P. 155–172. URL: https://doi.org/10.1007/978-1-4842-3745-8_7. (date of access: 21.03.2024).

21. PixelatedPope. GMS2 Cameras: As Simple as Possible, 2018. *YouTube*. URL: https://www.youtube.com/watch?v=_g1LQ6aIJFk. (date of access: 22.03.2024).

22. Heartbeast. GameMaker Studio 2 - TILE COLLISIONS, 2017. *YouTube*. URL: https://www.youtube.com/watch?v=YObbsg2Hbcw. (date of access: 22.03.2024).

23. Tom Francis. *Tom Francis*. URL: https://www.pentadact.com/category/games-writing/. (date of access: 23.03.2024).

24. Learn LLVM 12 | Packt. *Packt*. URL: https://www.packtpub.com/product/gamemaker-studio-2-cookbook-second-edition/9781839213502. (date of access: 23.03.2024).

25. Gamemaker. *Reddit*. URL: https://www.reddit.com/r/gamemaker/. (date of access: 13.03.2024).

26. Unreal Engine 5.4 Documentation. *Epic Developer Community*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-4-documentation. (date of access: 06.03.2024).

27. Making Text Appear And Fade. *GameMaker Community*. URL: https://forum.gamemaker.io/index.php?threads/making-text-appear-and-fade.111926/. (date of access: 24.03.2024).

28. Gamemaker!. *GameDev.net*. URL: https://www.gamedev.net/forums/topic/366516-gamemaker/. (date of access: 13.03.2024).

29. Blogs recent news | Game Developer. *Game Developer*. URL: https://www.gamasutra.com/blogs/NathanLovato/20190114/333123/10_Reasons_You_Should_Develop_Your_Next_Game_in_GameMaker.php. (date of access: 22.03.2024).

30. Doman M. Effect of GameMaker on Student Attitudes and Perceptions of Instructors. *International Journal of Modern Education and Computer Science*. 2015. Vol. 7, no. 9. P. 1–13. URL: https://doi.org/10.5815/ijmecs.2015.09.01. (date of access: 15.03.2024).

31. Vinciguerra D., Howell A. GameMaker Standard. Taylor & Francis Group, 2015. 288 p.

32. Video Game History - Timeline & Facts. *HISTORY*. URL: https://www.history.com/topics/inventions/history-of-video-games. (date of access: 27.02.2024).

33. How-To: Launch your product on Kickstarter. *Visual and Simple Online Project Management Tool | Casual*. URL: https://casual.pm/how-to/launch-kickstarter. (date of access: 06.04.2024).

34. wikiHow. How to Start a Project on Kickstarter: 9 Steps (with Pictures). *wikiHow Life - Trustworthy How To Instructions*. URL: https://www.wikihow.life/Start-a-Project-on-Kickstarter. (date of access: 08.04.2024).

35. Особливості тестування ігор на ПК | Онлайн-курси від компанії QATestLab. *Онлайн-курси від компанії QATestLab | Головна сторінка*. URL: https://training.qatestlab.com/blog/technical-articles/gaming-on-pc-features/. (date of access: 02.04.2024).

36. Комп'ютерні ігри як важливий фактор формування життєвих навичок. Реферат. *Освіта.UA*. URL: https://osvita.ua/vnz/reports/psychology/28614/.(date of access: 25.02.2024).

37. Бібічіков І. Як розробляють ігри?. *Lemon School*. URL: https://lemon.school/blog/yak-rozroblyayut-igry. (date of access: 06.03.2024).

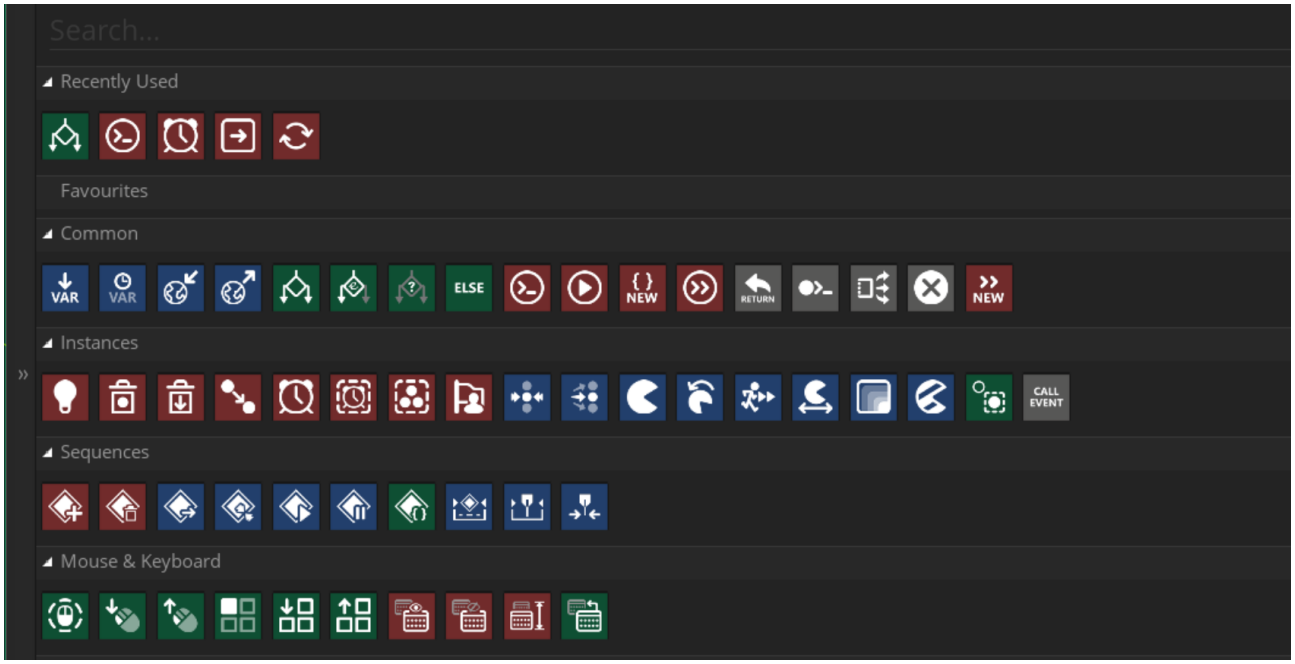
38. Розробка відеогри – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Розробка_відеогри. (date of access: 07.03.2024).

39. Алгоритми для розробників та їх застосування в проєктах. *FoxmindEd*. URL: https://foxminded.ua/alhorytmy-dlia-rozrobnykiv/. (date of access: 08.03.2024).

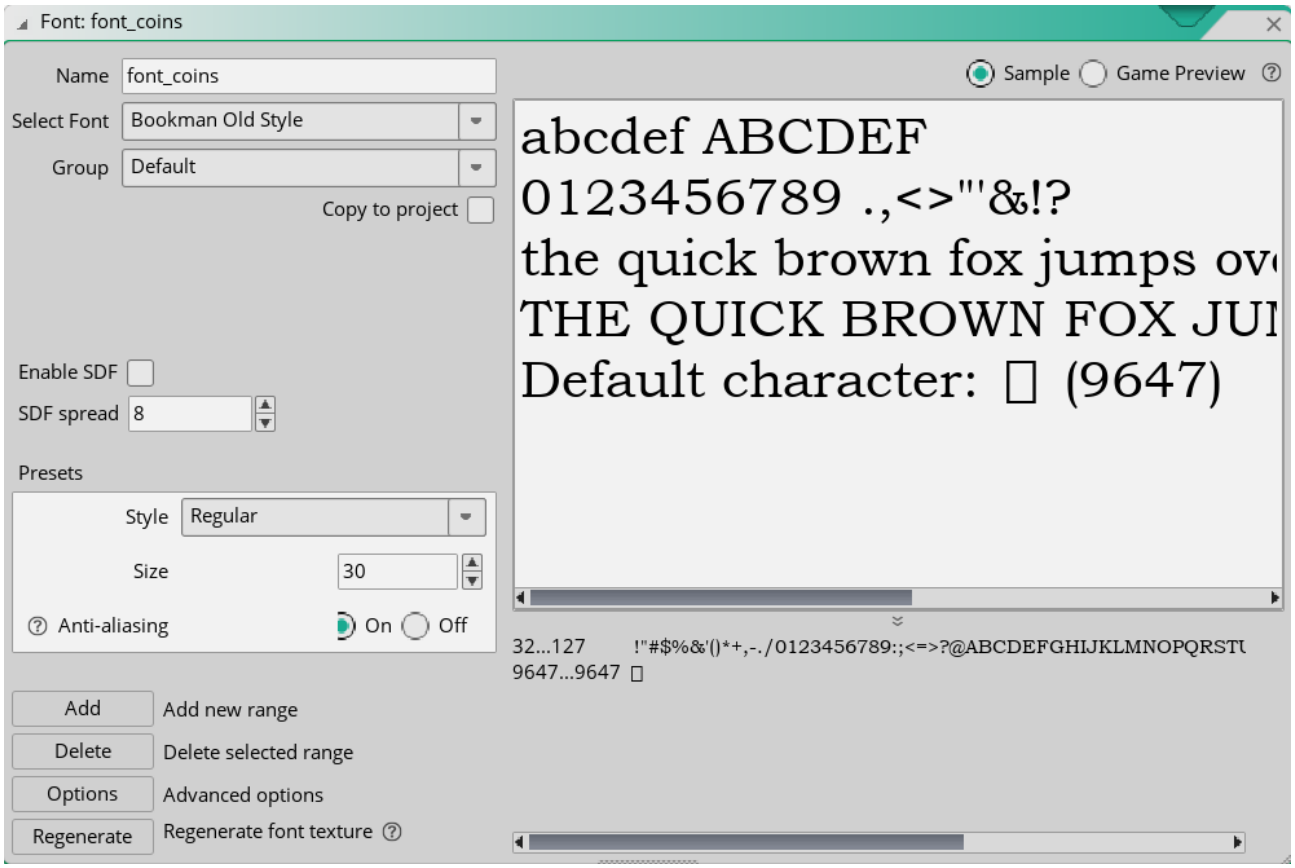
40. Бранець І. Опановуємо основи алгоритмів, або Як прискорити код з 15 до 1000 запитів за секунду. Dou. URL: https://dou.ua/lenta/articles/why-understanding-algorithms-is-important/. (date of access: 29.03.2024).

# ДОДАТОК А

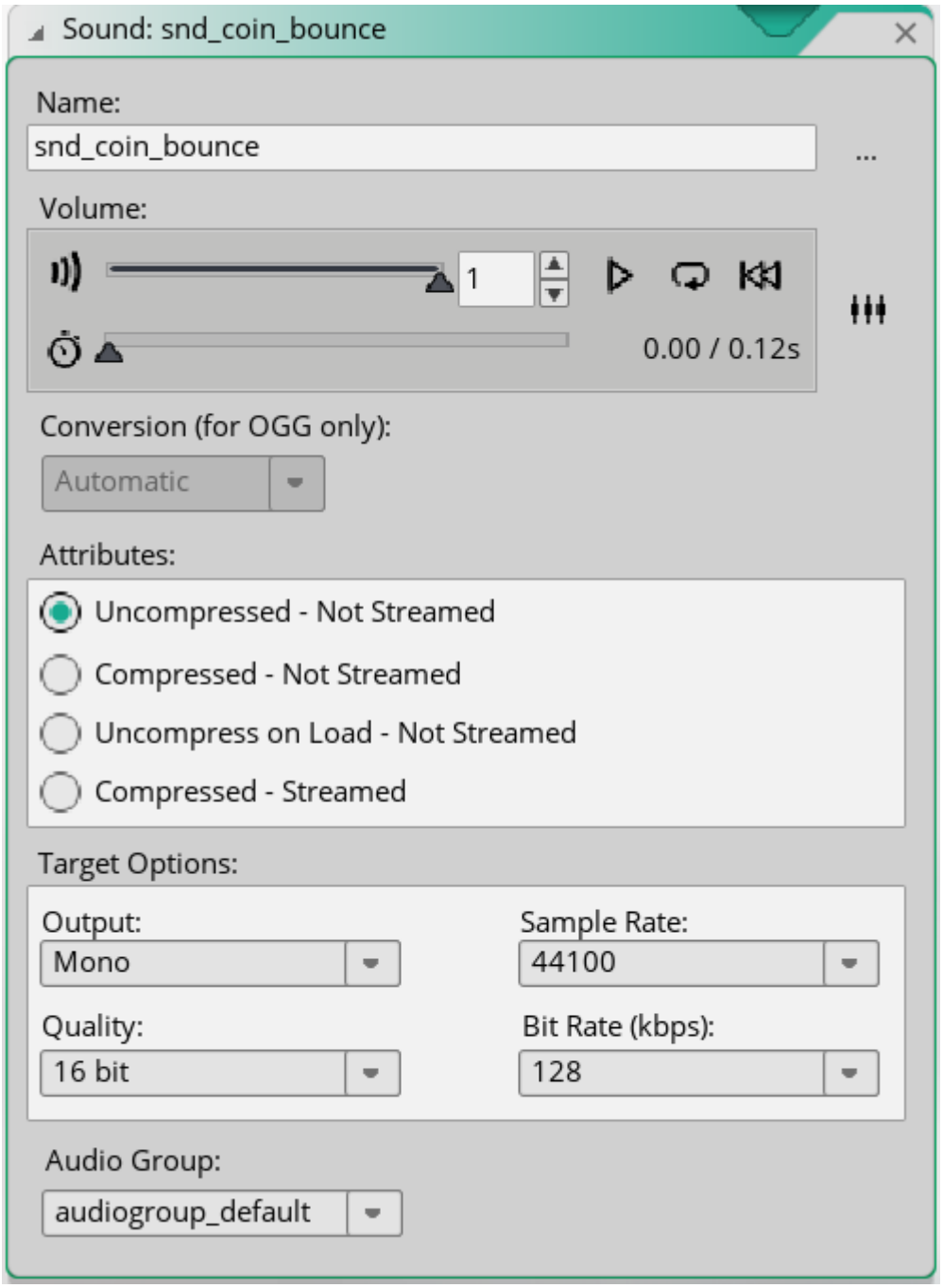
Бібліотека командних блоків.




Вікно створення шрифтів.

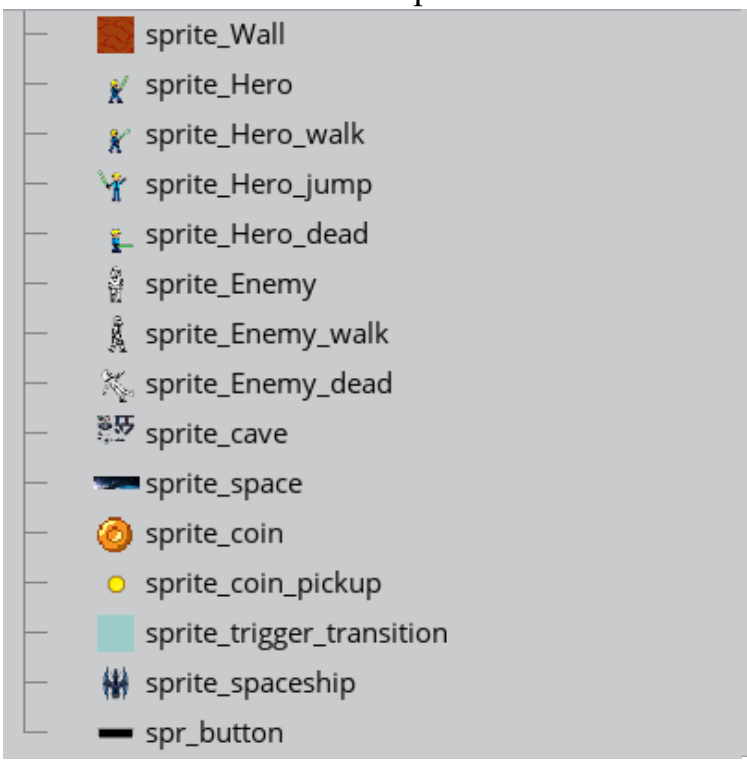


Вікно редагування аудіоконтенту



# ДОДАТОК Б

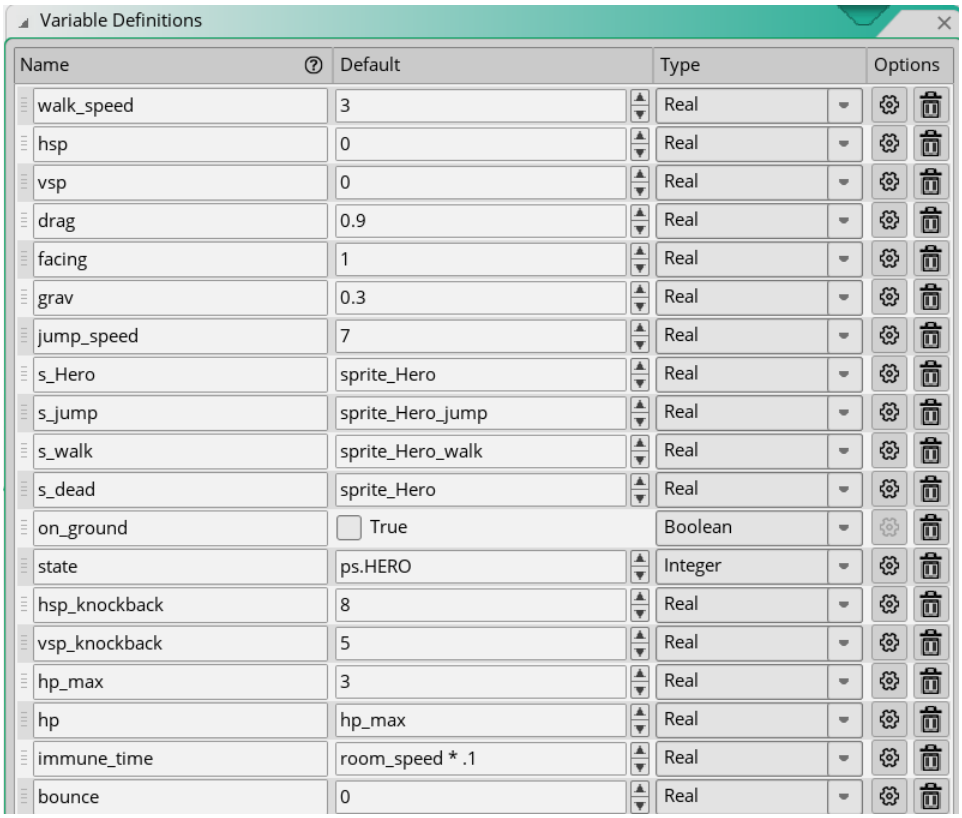
Список спрайтів



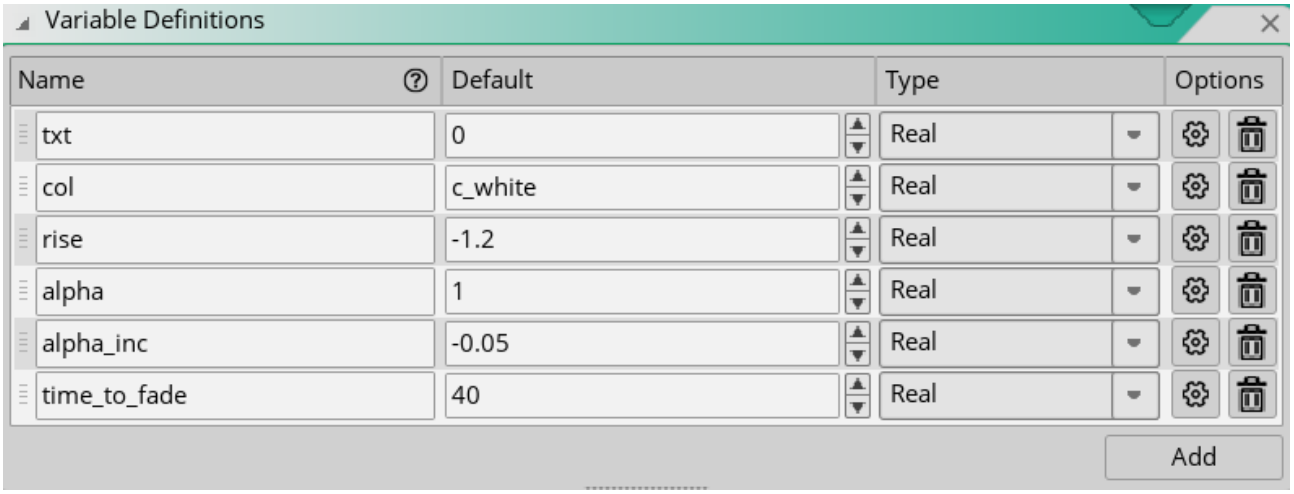

Список об’єктів



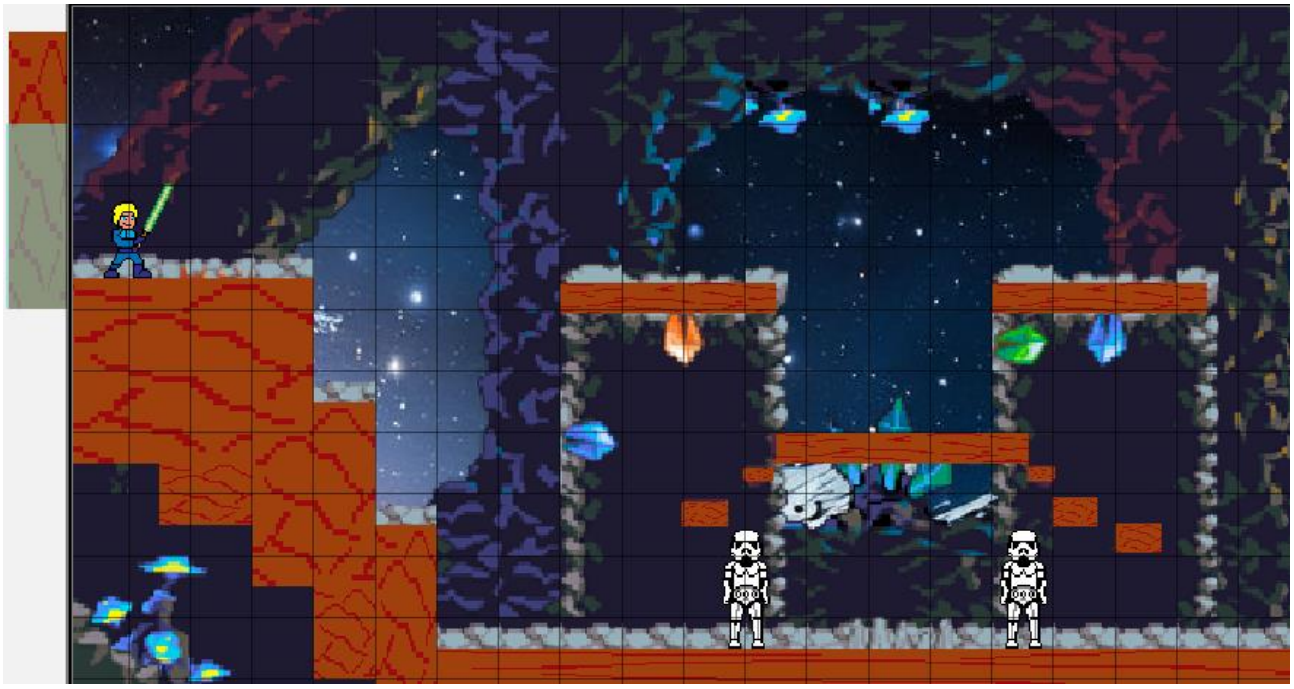

Створені змінні в об’єкті o_entity.



Створені змінні в об’єкті «o_floating_text»

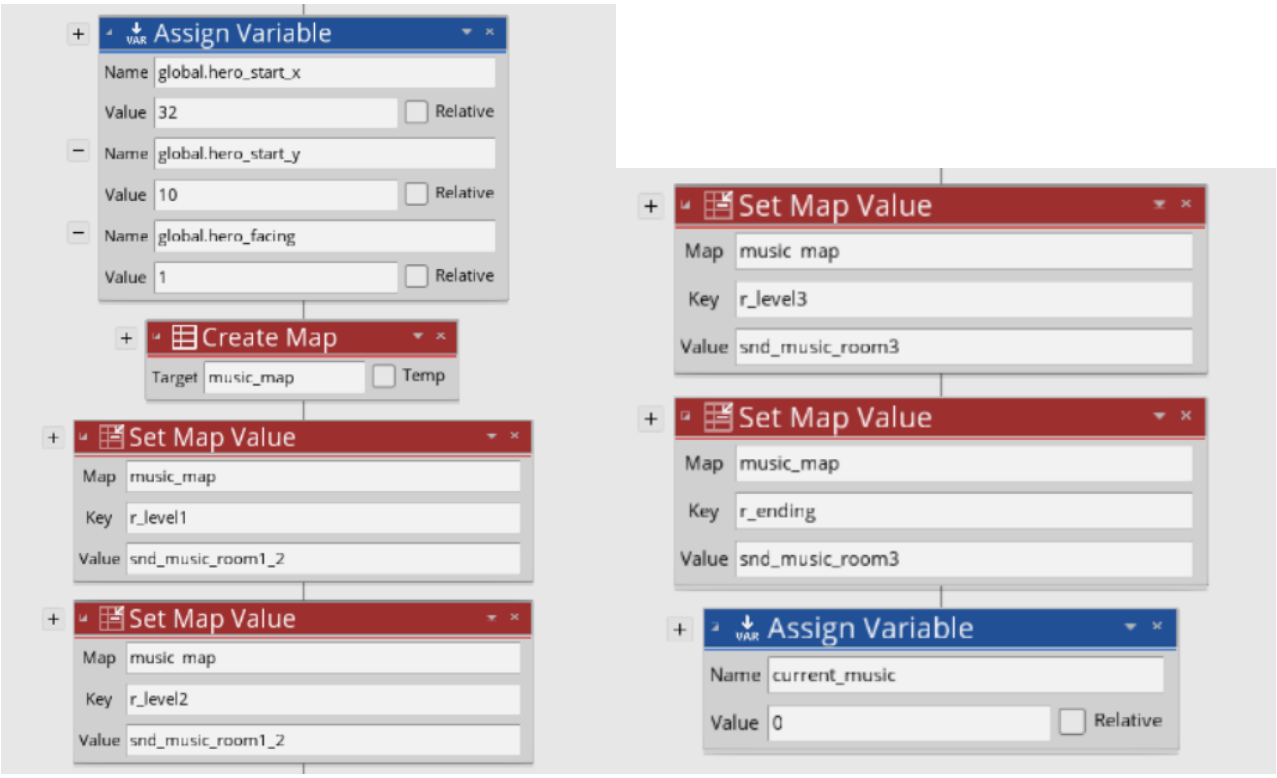



Вигляд першого руму з відображенням об’єктів “o_Wall” та “o_trigger_transition”.

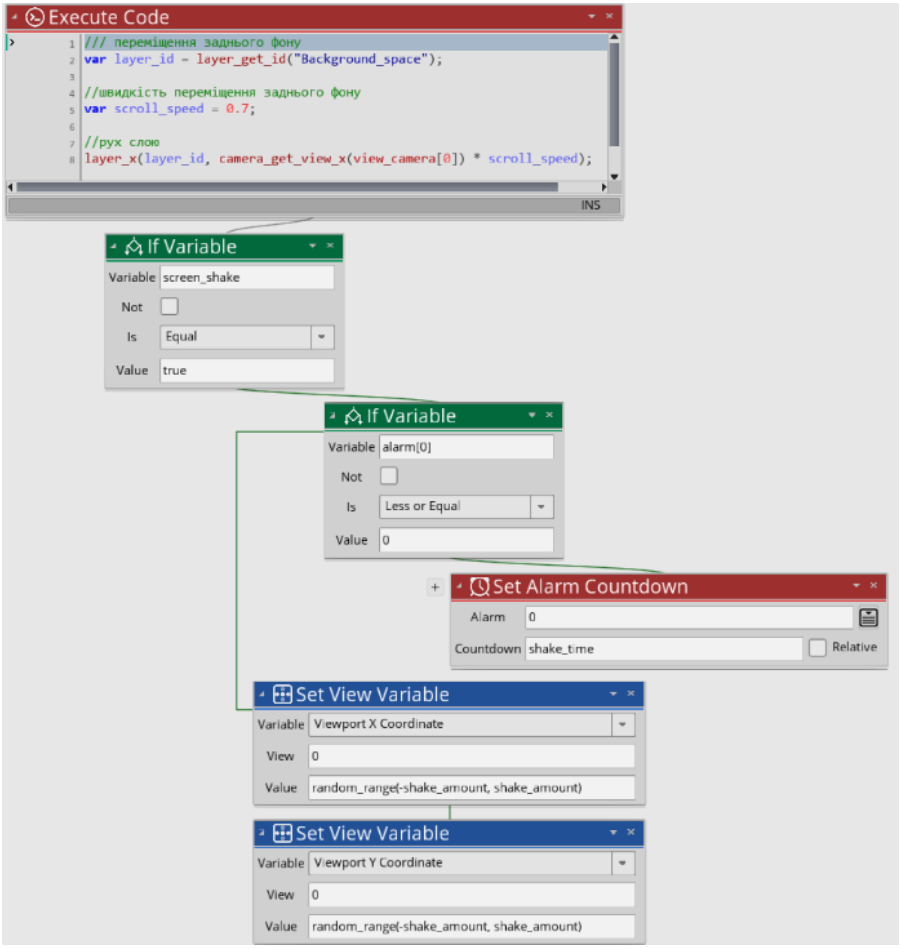


# ДОДАТОК В

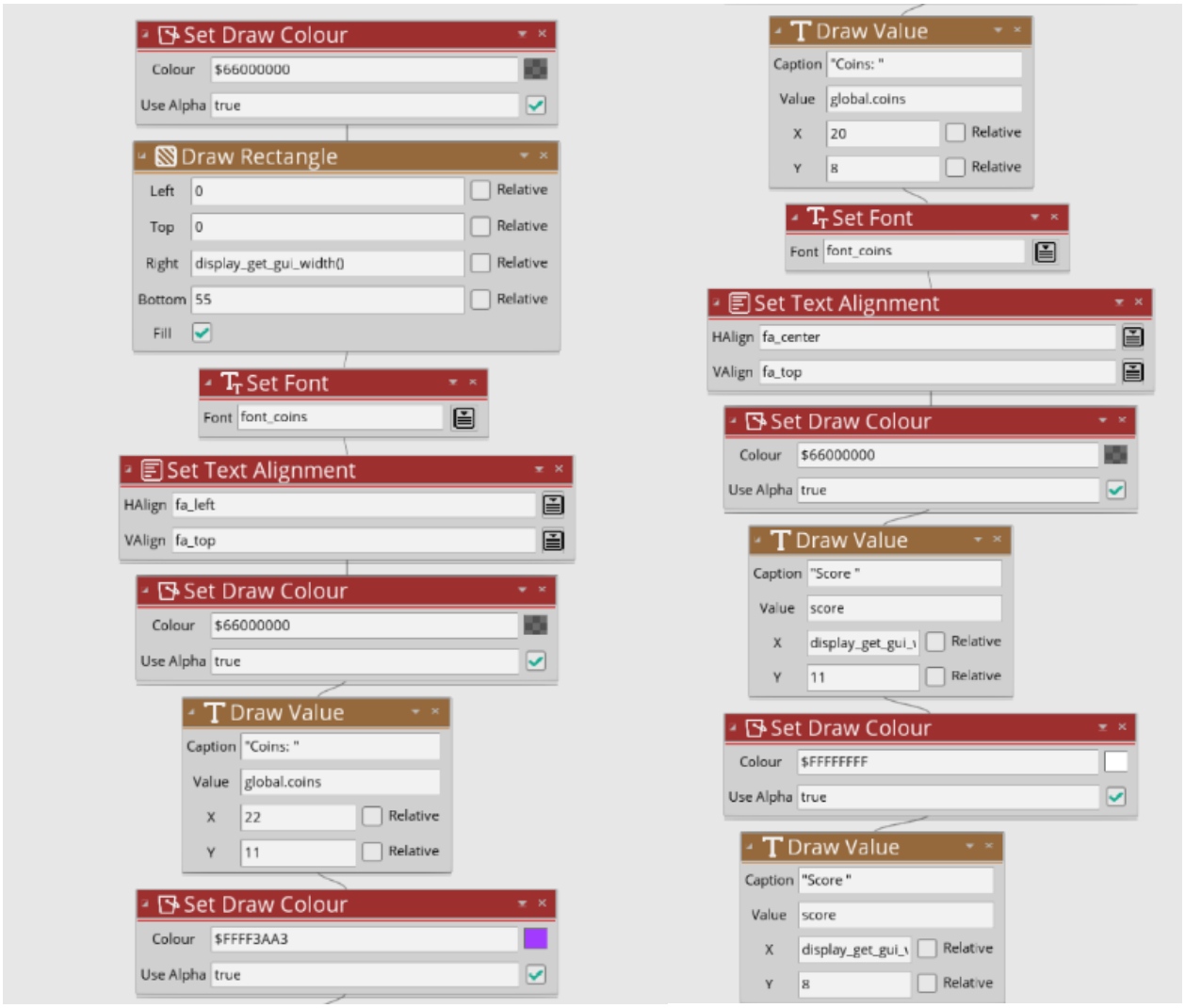
Подія «Create» об’єкту «o_game»:



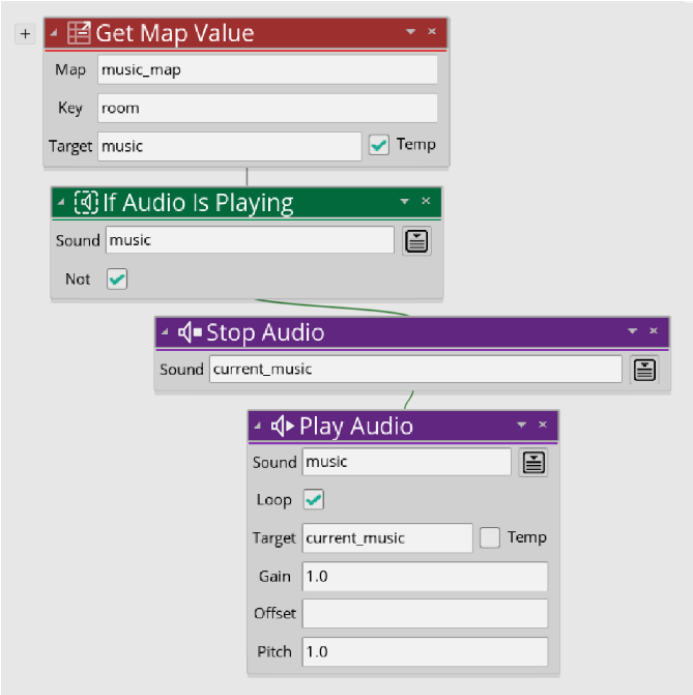

Подія «Step» об’єкту «o_game»:



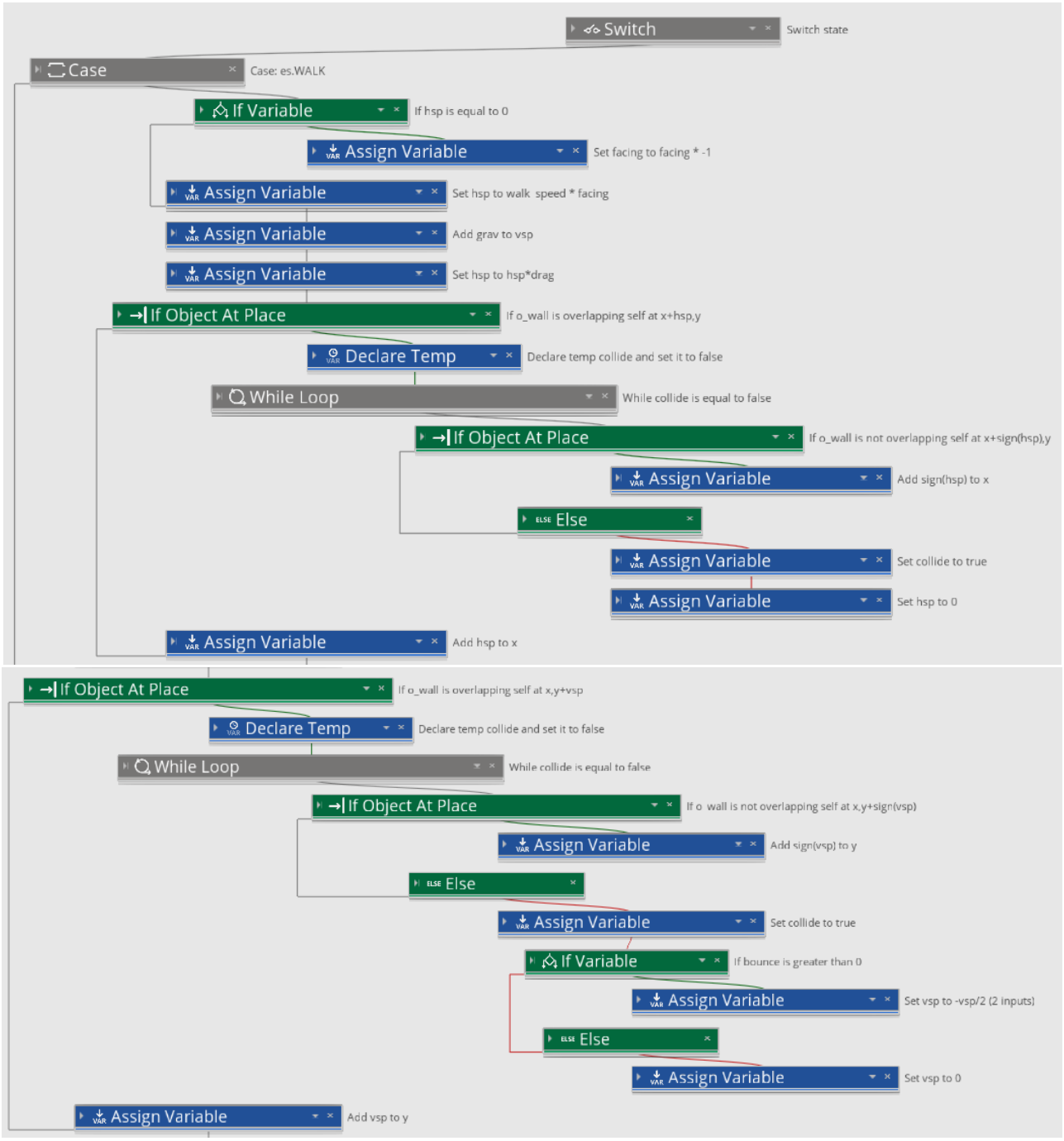

Подія «Draw» об’єкту «o_game»:

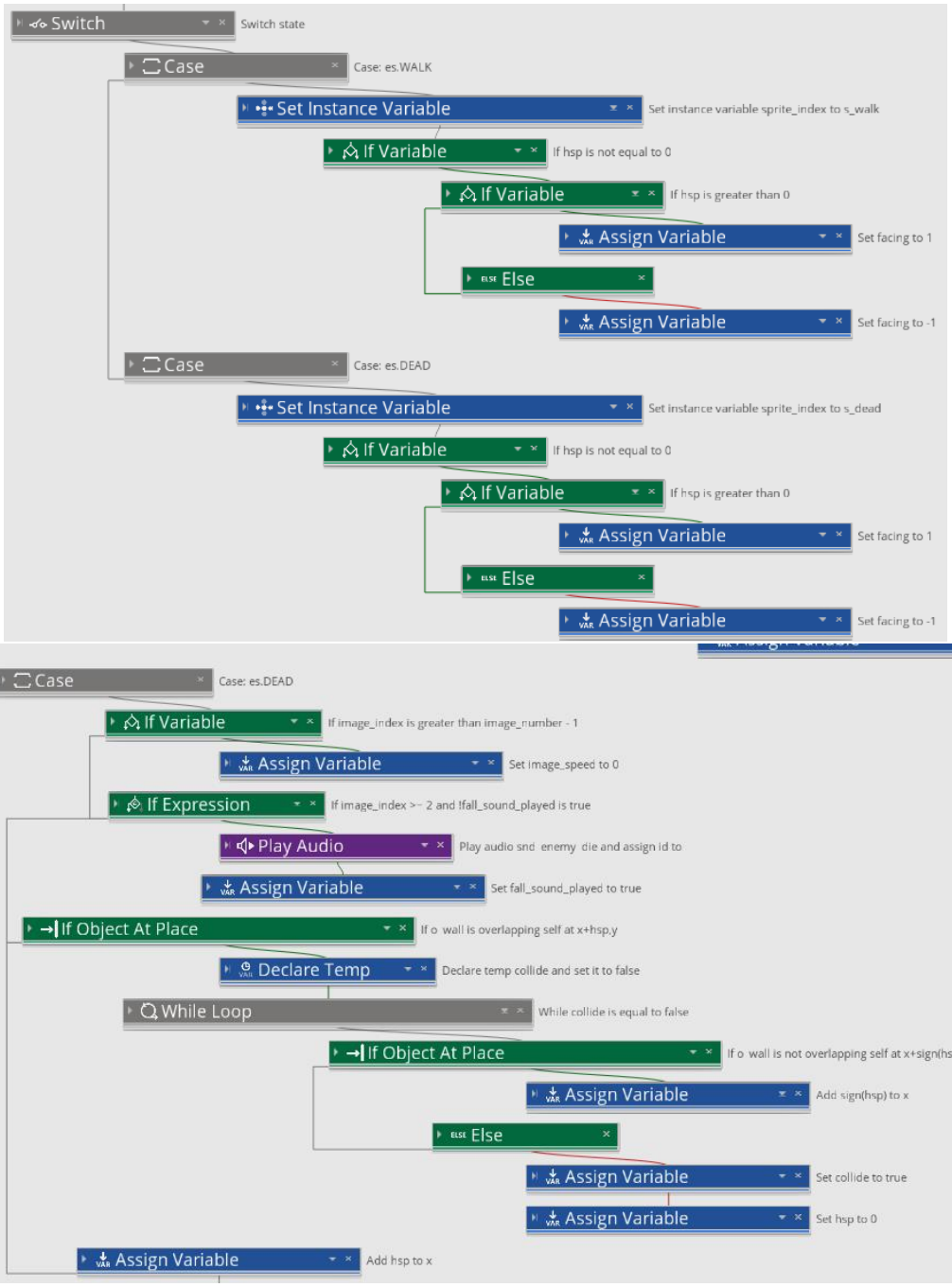


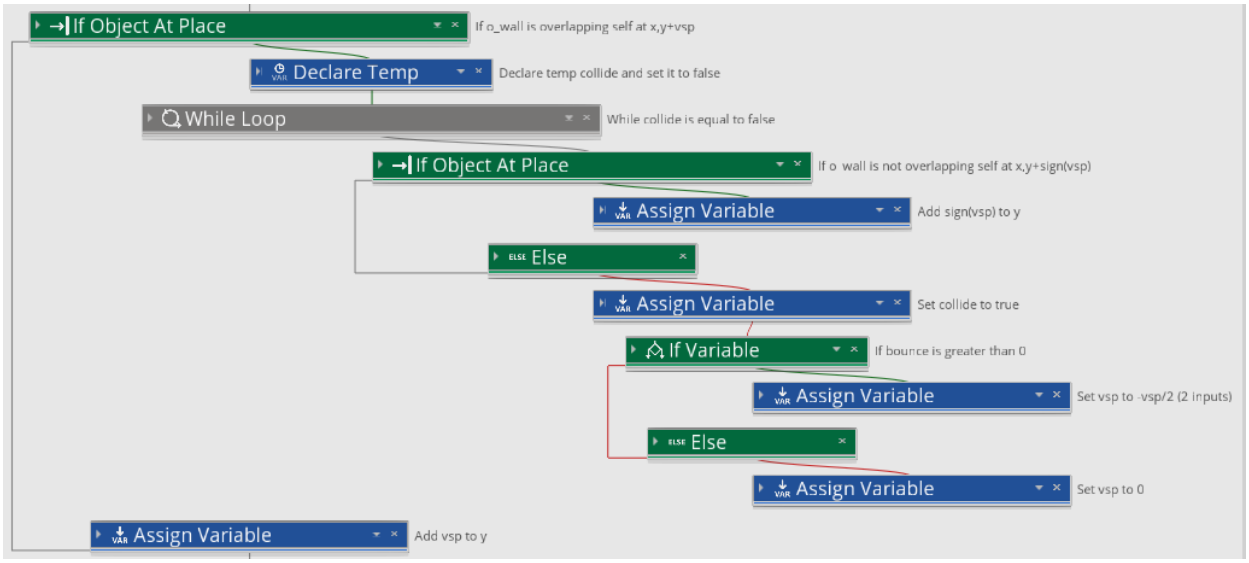

Подія «Room Start» об’єкту «o_game»:

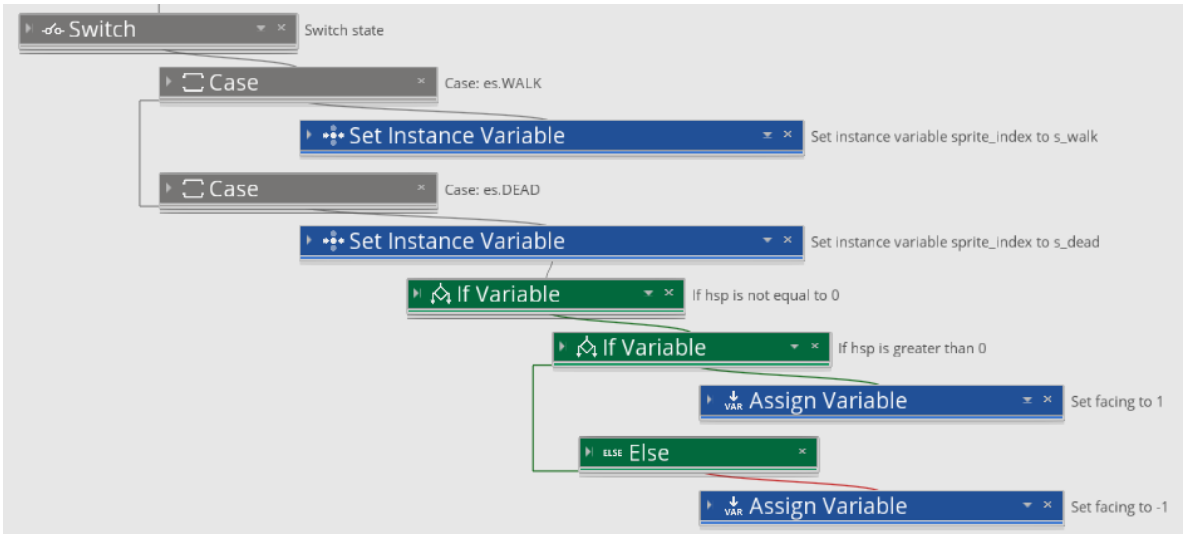



Подія «Step» об’єкту «o_Enemy»:

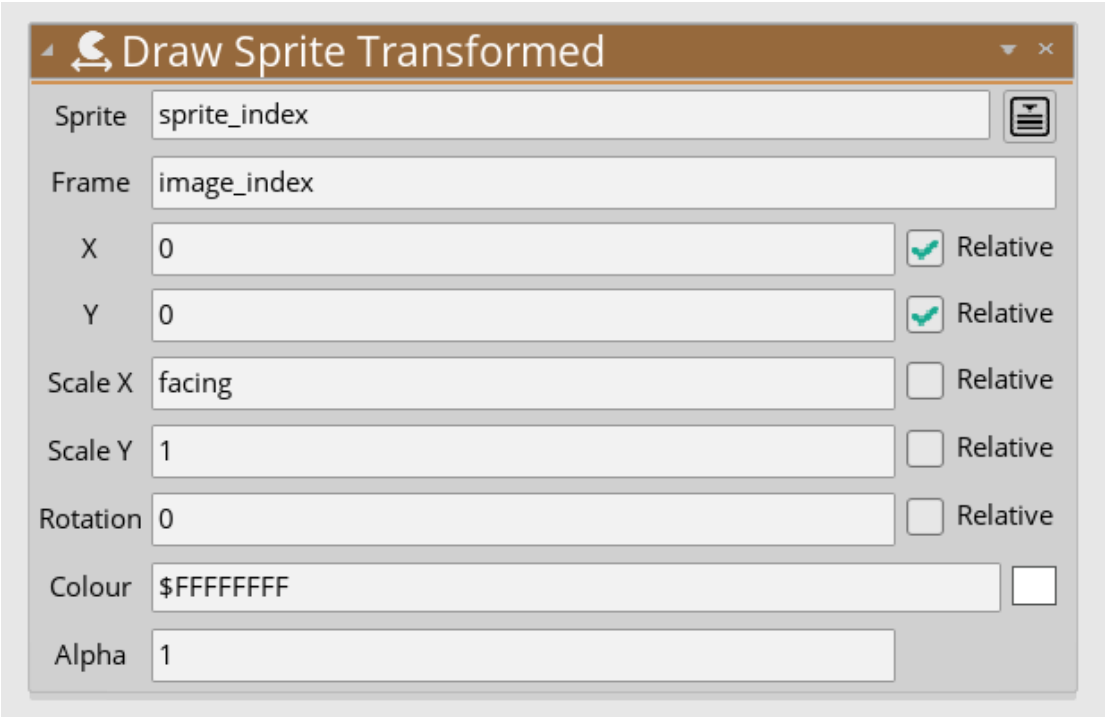

Switch
Switch state
Case
Case: es.WALK
If Variable
If hsp is equal to 0
Assign Variable
Set facing to facing * -1
Assign Variable
Set hsp to walk_speed * facing
Assign Variable
Add grav to vsp
Assign Variable
Set hsp to hsp*drag
If Object At Place
If o_wall is overlapping self at x+hsp,y
Declare Temp
Declare temp collide and set it to false
While Loop
While collide is equal to false
If Object At Place
If o_wall is not overlapping self at x+sign(hsp),y
Assign Variable
Add sign(hsp) to x
Else
Assign Variable
Set collide to true
Assign Variable
Set hsp to 0
Assign Variable
Add hsp to x
If Object At Place
If o_wall is overlapping self at x,y+vsp
Declare Temp
Declare temp collide and set it to false
While Loop
While collide is equal to false
If Object At Place
If o_wall is not overlapping self at x,y+sign(vsp)
Assign Variable
Add sign(vsp) to y
Else
Assign Variable
Set collide to true
If Variable
If bounce is greater than 0
Assign Variable
Set vsp to -vsp/2 (2 inputs)
Else
Assign Variable
Set vsp to 0
Assign Variable
Add vsp to y


Switch
Switch state
Case
Case: es.WALK
Set Instance Variable
Set instance variable sprite_index to s_walk
If Variable
If hsp is not equal to 0
If Variable
If hsp is greater than 0
Assign Variable
Set facing to 1
Else
Assign Variable
Set facing to -1
Case
Case: es.DEAD
Set Instance Variable
Set instance variable sprite_index to s_dead
If Variable
If hsp is not equal to 0
If Variable
If hsp is greater than 0
Assign Variable
Set facing to 1
Else
Assign Variable
Set facing to -1
Case
Case: es.DEAD
If Variable
If image_index is greater than image_number - 1
Assign Variable
Set image_speed to 0
If Expression
If image_index >= 2 and !fall_sound_played is true
Play Audio
Play audio snd_enemy_die and assign id to
Assign Variable
Set fall_sound_played to true
If Object At Place
If o_wall is overlapping self at x+hsp,y
Declare Temp
Declare temp collide and set it to false
While Loop
While collide is equal to false
If Object At Place
If o_wall is not overlapping self at x+sign(hsp),y
Assign Variable
Add sign(hsp) to x
Else
Assign Variable
Set collide to true
Assign Variable
Set hsp to 0
Assign Variable
Add hsp to x

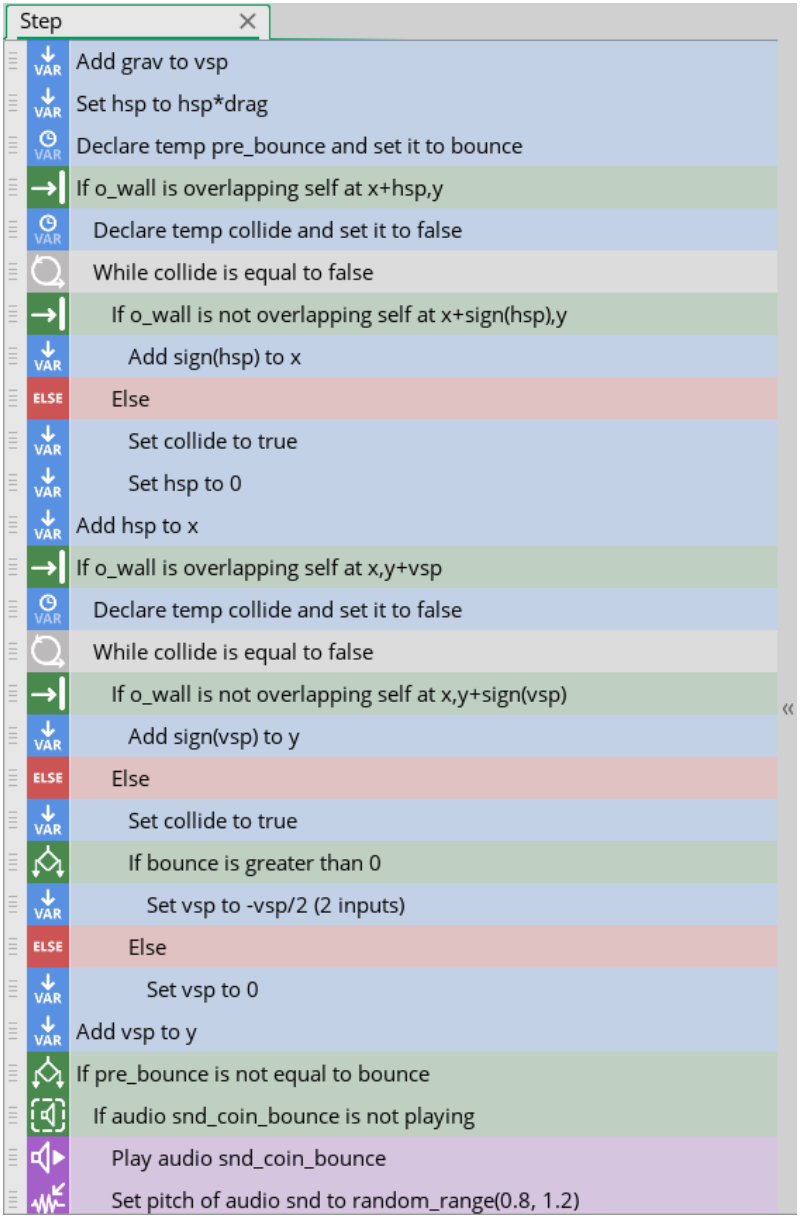


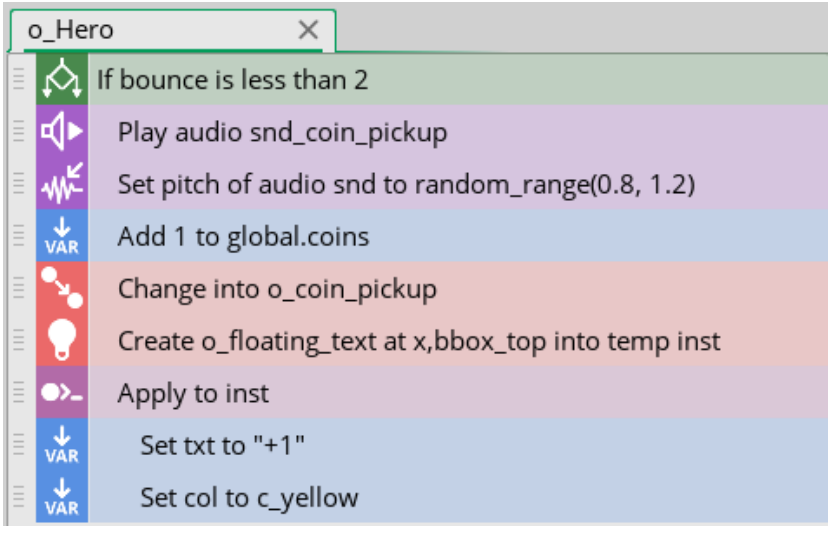

Подія «Create» об’єкту «o_Enemy»:



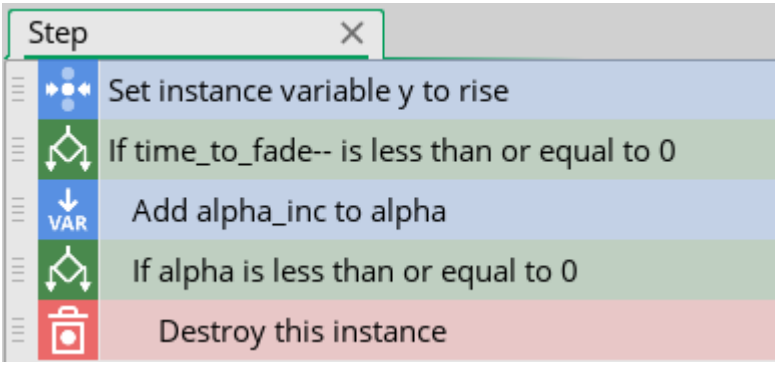
Подія «Step» об’єкту «o_item_par»:



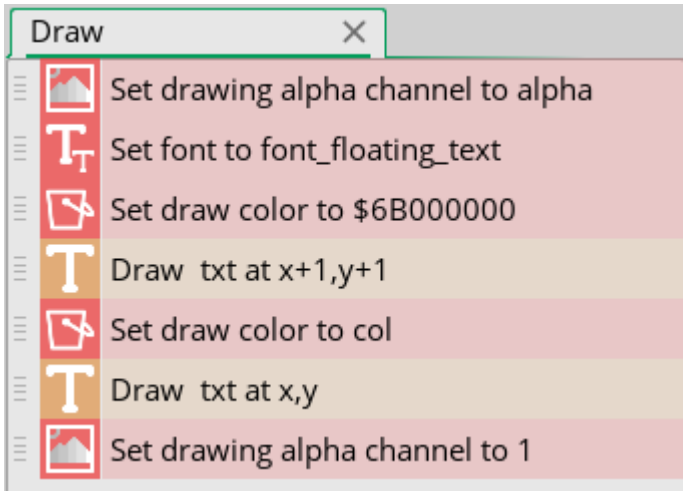

Подія «Collisions o_Hero» об’єкту «o_coin»:



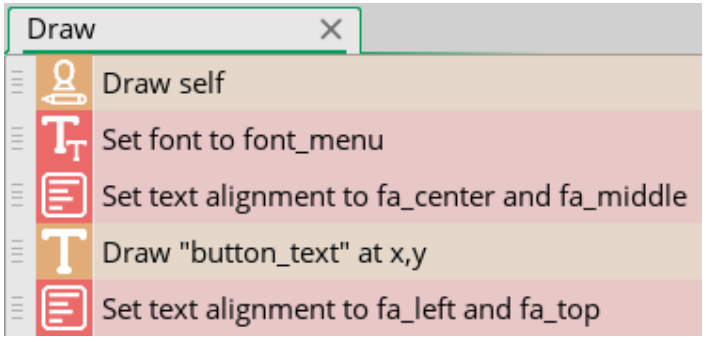
Подія «Step» об’єкту «o_floating_text»:



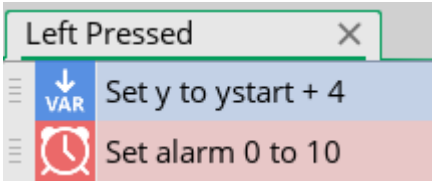

Подія «Draw» об’єкту «o_floating_text»:



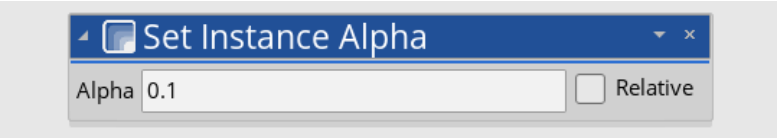

Подія «Draw» об’єкту «o_button_parent»:



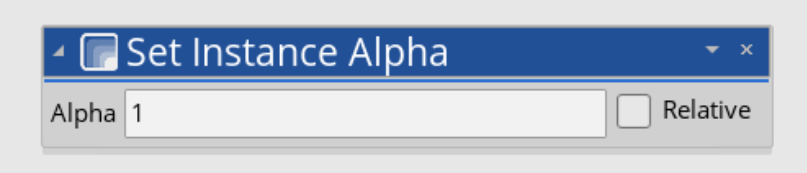

Подія «Left Pressed» об’єкту «o_button_parent»:



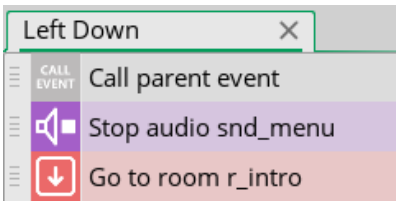

Подія «Mouse Enter» об’єкту «o_button_parent»:



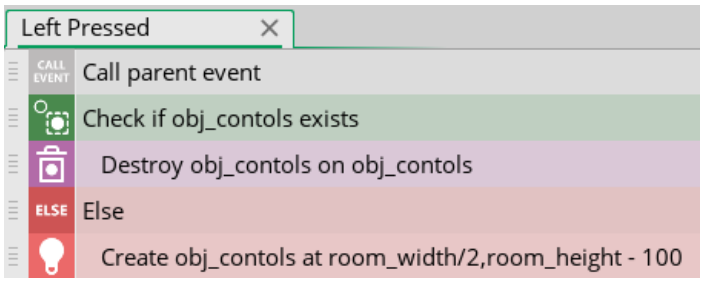

Подія «Mouse Leave» об’єкту «o_button_parent»:



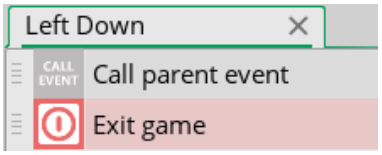

Подія «Left Down» об’єкту «o_button_play»:



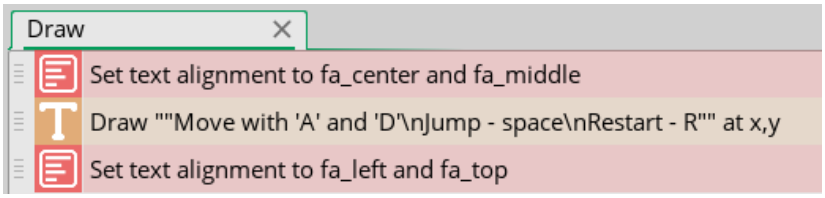

Подія «Left Pressed» об’єкту «o_button_options»:



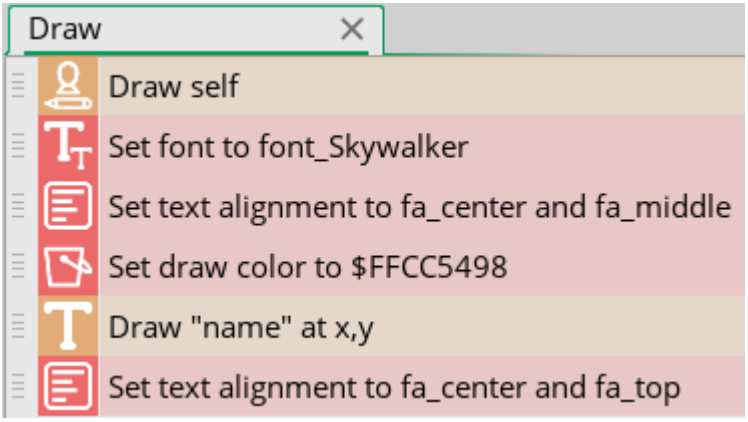

Подія «Left Down» об’єкту «o_button_quit»:



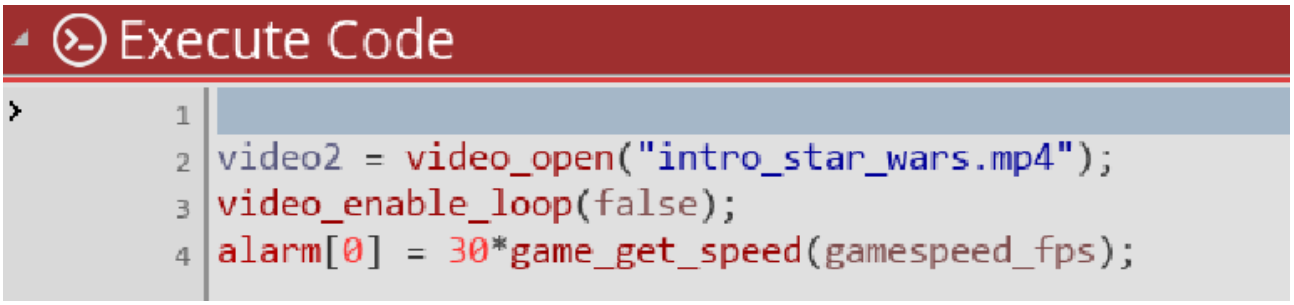

Подія «Draw» об’єкту «obj_controls»:




Подія «Draw» об’єкту «o_Skywalker»:




Подія «Create» об’єкту «o_video_intro»:




Подія «Destroy» об’єкту «o_video_intro»:




Подія «Alarm 0» об’єкту «o_video_intro»:



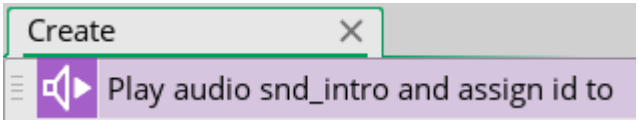

Подія «Draw» об’єкту «o_video_intro»:

Execute Code

```
var _videoData = video_draw();
var _videoStatus = _videoData[0];

if (_videoStatus == 0)
{
    draw_surface(_videoData[1], 50, 50);
}
if (video_get_status() == video_status_playing)
{
    draw_set_font(font_intro);
    draw_set_color(c_yellow);
    draw_text(185, 10, "Press Space to start the game");
    if keyboard_check_pressed(vk_space)
    {
        video_close();
        room_goto(r_level1);
    }
}
```

Подія «Create» об’єкту «o_intro_music»:

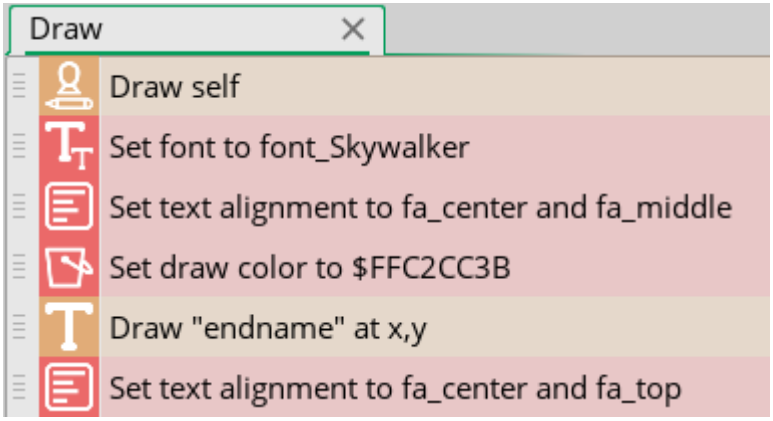



Подія «Create» об’єкту «o_ending»:

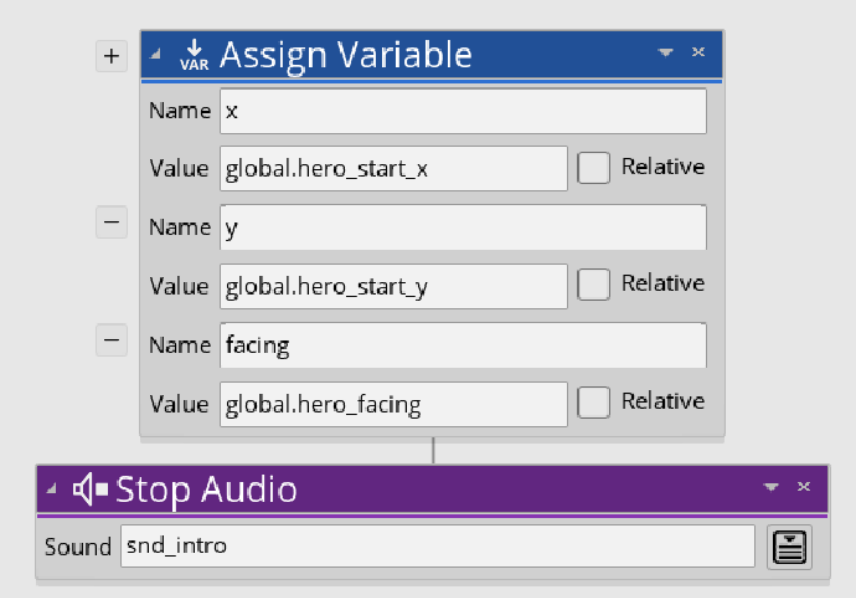
Execute Code

```
alarm[0] = 5*game_get_speed(gamespeed_fps);
```

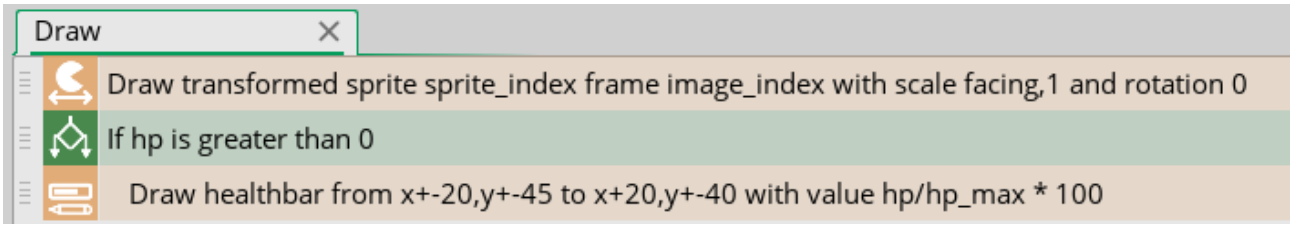
Подія «Draw» об’єкту «o_ending»:



Подія «Room Start» об’єкту «o_Hero»:




Подія «Draw» об’єкту «o_Hero»:




Подія «Collisions o_Enemy» об’єкту «o_Hero»:

o_Enemy

```
If state is not equal to ps.DEAD
    If other.state is not equal to es.DEAD
        If bbox_bottom < other.bbox_top + 10 is true (2 inputs)
            Play audio snd_enemy_land and assign id to
            Set vsp to -jump_speed
            Set alarm 0 to immune_time
            Apply to other
                Set instance variable image_index to 0
                Set hsp to 0
                Set state to es.DEAD
                Add points to score
                Create o_floating_text at x,bbox_top into temp inst
                Apply to inst
                    Set txt to "+" + string(other.points)
                Repeat irandom_range(4,6) times
                    Create o_coin at x+random_range(-10,10),bbox_top
                Apply to o_game
                    Set screen_shake to true
                    Set shake_amount to 6
        Else
            If state != ps.KNOCKBACK is true (2 inputs)
                Play audio snd_hero_damaged and assign id to
                Add -1 to hp
                Set hsp to sign(x - other.x) * hsp_knockback
                Set vsp to -vsp_knockback
                Set state to ps.KNOCKBACK
```

Подія «Step» об’єкту «o_Hero»:






```
If hp is less than or equal to 0
    Set state to ps.DEAD
    Play audio snd_hero_die and assign id to
If o_wall is overlapping self at x,y+grav
    Set on_ground to true
Else
    Set on_ground to false
If o_wall is overlapping self at x+hsp,y
    Declare temp collide and set it to false
    While collide is equal to false
        If o_wall is not overlapping self at x+sign(hsp),y
            Add sign(hsp) to x
        Else
            Set collide to true
            Set hsp to 0
    Add hsp to x
If o_wall is overlapping self at x,y+vsp
    Declare temp collide and set it to false
    While collide is equal to false
        If o_wall is not overlapping self at x,y+sign(vsp)
            Add sign(vsp) to y
        Else
            Set collide to true
            If bounce is greater than 0
                Set vsp to -vsp/2 (2 inputs)
            Else
                Set vsp to 0
    Add vsp to y
```

```
Switch state
    Case: ps.HERO
        Set instance variable sprite_index to s_Hero
        If hsp is not equal to 0
            If hsp is greater than 0
                Set facing to 1
            Else
                Set facing to -1
    Case: ps.WALK
        Set instance variable sprite_index to s_walk
        If hsp is not equal to 0
            If hsp is greater than 0
                Set facing to 1
            Else
                Set facing to -1
    Case: ps.JUMP
        Set instance variable sprite_index to s_jump
        If hsp is not equal to 0
            If hsp is greater than 0
                Set facing to 1
            Else
                Set facing to -1
        If vsp is less than 0
            Set instance variable image_index to 0
        Else
            Set instance variable image_index to 1
    Case: ps.KNOCKBACK
        Set instance variable sprite_index to s_jump
```

```
    If vsp is less than 0
        Set image_index to 0
    Else
        Set image_index to 1
    Case: ps.DEAD
        Set instance variable sprite_index to s_dead
Case: ps.WALK
    If ord("D") is down
        Set hsp to walk_speed
    If ord("A") is down
        Set hsp to -walk_speed
    If vk_space is pressed
        Set vsp to -jump_speed
    Add grav to vsp
    Set hsp to hsp*drag
    If abs(hsp) is less than 0.2
        Set hsp to 0
    If on_ground is equal to true
        If hsp is equal to 0
            Set state to ps.HERO
        Else
            Set state to ps.WALK
    Else
        Set state to ps.JUMP
    If hp is less than or equal to 0
        Set state to ps.DEAD
        Play audio snd_hero_die and assign id to
    If o_wall is overlapping self at x,y+grav
        Set on_ground to true
    Else
        Set on_ground to false
    Play audio snd_jump and assign id to
    If floor(image_index) == 1 is true
        Play audio snd_footsteps
        Set pitch of audio snd to random_range(0.7, 1.3)
    If o_wall is overlapping self at x+hsp,y
        Declare temp collide and set it to false
        While collide is equal to false
            If o_wall is not overlapping self at x+sign(hsp),y
                Add sign(hsp) to x
            Else
                Set collide to true
                Set hsp to 0
        Add hsp to x
    If o_wall is overlapping self at x,y+vsp
        Declare temp collide and set it to false
        While collide is equal to false
            If o_wall is not overlapping self at x,y+sign(vsp)
                Add sign(vsp) to y
            Else
                Set collide to true
                If bounce is greater than 0
                    Set vsp to -vsp/2 (2 inputs)
                Else
                    Set vsp to 0
        Add vsp to y
```

```
Switch state
  Case: ps.HERO
    Set instance variable sprite_index to s_Hero
    If hsp is not equal to 0
      If hsp is greater than 0
        Set facing to 1
      Else
        Set facing to -1
  Case: ps.WALK
    Set instance variable sprite_index to s_walk
    If hsp is not equal to 0
      If hsp is greater than 0
        Set facing to 1
      Else
        Set facing to -1
  Case: ps.JUMP
    Set instance variable sprite_index to s_jump
    If hsp is not equal to 0
      If hsp is greater than 0
        Set facing to 1
      Else
        Set facing to -1
    If vsp is less than 0
      Set instance variable image_index to 0
    Else
      Set instance variable image_index to 1
  Case: ps.KNOCKBACK
    Set instance variable sprite_index to s_jump
    If vsp is less than 0
      Set image_index to 0
    Else
      Set image_index to 1
  Case: ps.DEAD
    Set instance variable sprite_index to s_dead
  Case: ps.KNOCKBACK
    Add grav to vsp
    Set hsp to hsp*drag
    If abs(hsp) is less than 0.2
      Set hsp to 0
    If on_ground is equal to true
      If hsp is equal to 0
        Set state to ps.HERO
      Else
        Set state to ps.WALK
    Else
      Set state to ps.JUMP
    If vsp is greater than 0
      If on_ground is equal to true
        If vsp is greater than 0
          If on_ground is equal to true
            If o_wall is overlapping self at x,y+grav
              Set on_ground to true
            Else
              Set on_ground to false
    If o_wall is overlapping self at x+hsp,y
      Declare temp collide and set it to false
```

```
While collide is equal to false
  If o_wall is not overlapping self at x+sign(hsp),y
    Add sign(hsp) to x
  Else
    Set collide to true
    Set hsp to 0
If hp is less than or equal to 0
  Set state to ps.DEAD
  Play audio snd_hero_die and assign id to
Add hsp to x
If o_wall is overlapping self at x,y+vsp
  Declare temp collide and set it to false
  While collide is equal to false
    If o_wall is not overlapping self at x,y+sign(vsp)
      Add sign(vsp) to y
    Else
      Set collide to true
      If bounce is greater than 0
        Set vsp to -vsp/2 (2 inputs)
      Else
        Set vsp to 0
Add vsp to y
Switch state
  Case: ps.HERO
    Set instance variable sprite_index to s_Hero
    If hsp is not equal to 0
      If hsp is greater than 0
        Set facing to 1
      Else
        Set facing to -1
  Case: ps.WALK
    Set instance variable sprite_index to s_walk
    If hsp is not equal to 0
      If hsp is greater than 0
        Set facing to 1
      Else
        Set facing to -1
  Case: ps.JUMP
    Set instance variable sprite_index to s_jump
    If hsp is not equal to 0
      If hsp is greater than 0
        Set facing to 1
      Else
        Set facing to -1
    If vsp is less than 0
      Set image_index to 0
    Else
      Set image_index to 1
  Case: ps.KNOCKBACK
    Set instance variable sprite_index to s_jump
    If vsp is less than 0
      Set image_index to 0
    Else
      Set image_index to 1
  Case: ps.DEAD
    Set instance variable sprite_index to s_dead
```

```
Case: ps.DEAD
    Stop audio snd_music_room1_2
    Stop audio snd_menu
    Add grav to vsp
    Set hsp to hsp*drag
    If image_index is greater than image_number - 1
        Set image_speed to 0
    If o_wall is overlapping self at x+hsp,y
        Declare temp collide and set it to false
        While collide is equal to false
            If o_wall is not overlapping self at x+sign(hsp),y
                Add sign(hsp) to x
            Else
                Set collide to true
                Set hsp to 0
        Add hsp to x
    If o_wall is overlapping self at x,y+vsp
        Declare temp collide and set it to false
        While collide is equal to false
            If o_wall is not overlapping self at x,y+sign(vsp)
                Add sign(vsp) to y
            Else
                Set collide to true
                If bounce is greater than 0
                    Set vsp to -vsp/2 (2 inputs)
                Else
                    Set vsp to 0
        Add vsp to y
```

```
Switch state
    Case: ps.HERO
        Set instance variable sprite_index to s_Hero
        If hsp is not equal to 0
            If hsp is greater than 0
                Set facing to 1
            Else
                Set facing to -1
    Case: ps.WALK
        Set instance variable sprite_index to s_walk
        If hsp is not equal to 0
            If hsp is greater than 0
                Set facing to 1
            Else
                Set facing to -1
    Case: ps.JUMP
        Set instance variable sprite_index to s_jump
        If hsp is not equal to 0
            If hsp is greater than 0
                Set facing to 1
            Else
                Set facing to -1
        If vsp is less than 0
            Set image_index to 0
        Else
            Set image_index to 1
    Case: ps.KNOCKBACK
        Set instance variable sprite_index to s_jump
```

```
If vsp is less than 0
    Set image_index to 0
Else
    Set image_index to 1
Case: ps.DEAD
    Set instance variable sprite_index to s_dead
```