МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА РОЛЬОВОЇ ПРИГОДНИЦЬКОЇ 2D ВІДЕОГРИ ЗАСОБАМИ ІГРОВОГО РУШІЯ GAME MAKER»**

Виконав: студент 3 курсу, групи 6.1211-пі-с
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія (зі скороченим терміном навчання)
(назва освітньої програми)
А.Ю. Крицкалюк
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, доцент, к.т.н. Лимаренко Ю.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент професор кафедри комп'ютерних наук, доцент, д.т.н. Шило Г.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія (зі скороченим терміном навчання)

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

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Крицкалюку Антону Юрійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка рольової пригодницької 2D відеогри засобами ігрового рушія Game Maker

керівник роботи Лимаренко Юлія Олексіївна, к.т.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.
3. Документація по ігровому рушію Game Maker.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Реалізація 2D відеогри за допомогою ігрового рушія Game Maker.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 25.12.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	09.01.2024	
2.	Збір вихідних даних.	01.02.2024	
3.	Обробка методичних та теоретичних	01.03.2024	
	джерел.		
4.	Розробка першого та другого розділу.	15.04.2024	
5.	Розробка третього розділу.	20.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	21.06.2024	

Студент _____ А.Ю. Крицкалюк
(підпис) (ініціали та прізвище)

Керівник роботи _____ Ю.О. Лимаренко
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

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

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка рольової пригодницької 2D відеогри засобами ігрового рушія Game Maker»: 52 с., 34 рис., 11 джерел.

ВІДЕОГРА, ГЕЙМДИЗАЙН, ІГРОВИЙ ПРОЦЕС, ІГРОВИЙ РУШІЙ, ІНДІ-ПРОЄКТ, ПІКСЕЛЬАРТ.

Об’єкт дослідження – процес розробки ігрового застосунку.

Мета роботи: розробка ігрового застосунку.

Методи дослідження – методи об’єктно-орієнтованого програмування та геймдизайну.

У роботі розроблено ігровий застосунок на рушії Game Maker мовою програмування GML (Game Maker Language), що складається з одного рівня, котрий демонструє основні елементи ігрового процесу відеогри, створено та підготовано необхідні графічні матеріали, що використовуються у створенні застосунку, створено необхідний музичний супровід, продемонстровано процес аналізу предметної області та визначення вимог, проєктування та створення гри у візуальному стилі піксельарт.

SUMMARY

Bachelor’s qualifying paper “Development of a Role-Playing Adventure 2D Video Game Using the Game Maker Game Engine”: 52 pages, 34 figures, 11 references.

VIDEO GAME, GAMEDESIGN, GAME PROCESS, GAME ENGINE, INDIE PROJECT, PIXELART.

The object of the study is the process of developing a game application.

The aim of the study is development of a game application.

The methods of research are the methods of object-oriented programming and game design.

The work developed a game application on the Game Maker engine in the GML (Game Maker Language) programming language consisting of one level, which demonstrates the main elements of the gameplay of a video game, created and prepared the necessary graphic materials used in the creation of the application, created the necessary musical accompaniment, demonstrated the process of analyzing the subject area and defining requirements, designing and creating games in the visual style of pixel art.

ЗМІСТ

ВСТУП

Сучасний світ важко уявити без інформаційних технологій та Інтернету, які сьогодні пронизують усі сфери нашого життя, включаючи такі важливі аспекти, як комунікації, робота, навчання та, звичайно ж, розваги. Серед розваг особливе місце займають комп'ютерні та відеоігри, що стали невід'ємною частиною сучасної культури та дозвілля. Щодня мільйони людей по всьому світу витрачають свій вільний час на різноманітні ігрові проєкти, які забезпечують їм не лише розваги, але й можливість зануритися у віртуальні світи, пережити нові пригоди та відчути незабутні емоції. Завдяки цьому, створення ігор стало настільки ж актуальною і важливою роботою, як і розробка будь-яких інших інформаційних систем, що забезпечують наш повсякденний комфорт.

Метою цієї курсової роботи є створення ігрового застосунку в 2D стилістиці піксельарт. Піксельарт є популярним напрямом в ігровій індустрії завдяки своїй унікальній естетиці та можливості передати атмосферу класичних ігор, що додає проєктам особливого шарму та ретро-відчуття. Для досягнення поставленої мети необхідно виконати низку конкретних завдань, які забезпечать успішне втілення ідеї в життя.

Проаналізувати предметну область, для чого необхідно дослідити жанр гри, його особливості та вимоги до подібних проєктів. Важливо зрозуміти, які механіки, сюжети та візуальні стилі характерні для ігор в стилі піксельарт.

Проаналізувати приклади ігрових застосунків подібного жанру та стилю, для чого варто розглянути успішні проєкти в жанрі 2d піксельарт, вивчити їхні сильні та слабкі сторони, щоб отримати цінні уроки та ідеї для власного проєкту.

Проаналізувати наявні інструменти, для чого потрібно вивчити різноманітні програмні інструменти та платформи, доступні для створення 2d ігор, зокрема ті, що підтримують піксельарт графіку. Це допоможе обрати

найкращий інструмент для розробки гри.

Сформулювати вимоги до ігрового застосунку, для чого, на основі аналізу предметної області та існуючих прикладів, необхідно чітко визначити вимоги до ігрового застосунку. це включає як функціональні вимоги, так і вимоги до користувацького інтерфейсу, графіки, звуку та продуктивності.

Спроєктувати схему даних, для чого потрібно розробити структуру даних, що буде використовуватися в грі. Це включає визначення основних об’єктів гри, їх атрибутів та взаємозв’язків між ними.

Виконати ряд дій, спрямованих на створення візуальних та аудіо компонентів майбутньої гри, що передбачає створення графіки в стилі піксельарт, а також розробку аудіо-супроводу, що включає музику та можливі звукові ефекти, які будуть використовуватися в грі.

Програмно реалізувати та протестувати ігровий застосунок, для чого на заключному етапі потрібно здійснити програмну реалізацію всіх запланованих функцій та компонентів гри, а також провести тестування для виявлення та виправлення можливих помилок. Це забезпечить стабільну роботу гри та відповідність всім вимогам.

Таким чином, виконання всіх цих завдань дозволить створити якісний та цікавий ігровий застосунок у 2D стилістиці піксельарт.

1 ВИЗНАЧЕННЯ ВИМОГ

1.1 Аналіз предметної області

Сучасна ігрова індустрія являє собою величезний ринок, який складається з безлічі різноманітних жанрів і платформ. Умовно цей ринок можна розділити на дві основні частини: ААА-ігри та інді-ігри.

До перших відносяться проєкти великих студій, які розробляються із залученням середніх або великих видавців, що володіють значними фінансовими ресурсами, необхідними для розробки, маркетингу та реклами. Чітких критеріїв приналежності тієї чи іншої гри до категорії ААА немає, але зазвичай це клас просунутих високобюджетних ігор, розробка яких пов’язана зі значним фінансовим ризиком і необхідністю досягнення високих показників продажів для покриття витрат і отримання прибутку [1, 2]. До ігор класу ААА можна віднести такі відомі проєкти, як Grand Theft Auto V, Call of Duty, серія ігор Assassin’s Creed і багато інших. Всі ці ігри відрізняються складністю у виробництві, яка не під силу звичайному одиничному розробнику, оскільки вимагають великої команди фахівців, передових технологій та великих інвестицій.

Натомість до категорії інді-ігор відносяться проєкти незалежних розробників, які не залежать від підтримки великих видавців. Такі ігри часто не мають значного бюджету і можуть розповсюджуватися за низькою ціною або навіть безкоштовно. Зазвичай вони мають невеликий розмір і не мають масштабної рекламної кампанії, або ця реклама обмежена [2].

Прикладами таких ігор є Undertale, Foxhole, RimWorld, Among Us і багато інших. Проте існують інді-ігри, які з часом набувають популярності і навіть ознак ААА-ігор завдяки своїй інноваційності та залученню великої кількості гравців.

Яскравим прикладом такої трансформації є гра Minecraft, яка спочатку була інді-проєктом, але з часом перетворилася на один з найпопулярніших і найуспішніших ігрових проєктів у світі.

Окремо від усіх виділяється мобільний геймінг, який призначений для мобільних пристроїв, таких як телефони та планшети. Мобільні ігри часто мають короткі сеанси гри, зручний інтерфейс для сенсорного керування і орієнтовані на широку аудиторію, що робить їх надзвичайно популярними в сучасному світі.

Геймплейно ігри часто поділяють за положенням камери, через яку гравець спостерігає за ігровим світом. Основні типи камери – це вид від першої особи та вид від третьої особи.

Вид від першої особи означає, що камера знаходиться в голові персонажа, таким чином гравець бачить ігровий світ очима персонажа, яким керує. Це допомагає гравцю краще відчувати себе частиною світу гри, забезпечуючи глибше занурення та більш особисте сприйняття подій гри.

Вид від третьої особи може бути різним. Це може бути вид з фіксованим кутом камери, коли гравець може спостерігати за персонажем і світом лише з однієї позиції, або з динамічним кутом, коли гравець може змінювати кут зору. Існують ігри, де камера надає вид тільки зверху, тільки збоку, або де гравець може вільно оглядати ігровий світ з різних ракурсів [3]. Це дозволяє створювати різні геймплейні стилі та механіки, забезпечуючи унікальний досвід для гравців.

Виходячи з цієї інформації, ігровий застосунок, що розробляється в рамках даної кваліфікаційної роботи, буде представляти собою рівень інді-гри у візуальному стилі піксель-арт з фіксованим видом від третьої особи. Цей вибір обґрунтований бажанням створити гру, яка поєднує ретро-естетику піксель-арту з сучасними ігровими механіками, що дозволить досягти цікавого та привабливого кінцевого продукту для широкої аудиторії.

1.2 Вимоги до ігрового застосунку

Перед кожною грою постає ряд вимог, виконання яких є фактором її успішності. Для ігрового застосунку, що розробляється в рамках цієї кваліфікаційної роботи, вимоги спрощені. Це пояснюється тим, що створення повністю готового комерційного продукту займає надто багато часу, зусиль і, на фінальному етапі, ще й фінансових вкладень. До спрощених вимог відносяться:

– базові елементи ігрового оточення та дизайну;
– базові елементи управління ігровим персонажем;
– мінімальний штучний інтелект для неігрових персонажів.

Розглянемо ці вимоги більш детально.

Кожну гру можна розглядати як витвір візуального мистецтва. Це означає, що гравець одразу отримує багато інформації та формує перше враження про неї. Задля кращого розуміння специфіки обраної теми, розглянемо приклади подібних ігор.

Почнемо з візуальної складової. Гра Undertale використовує мінімалістичний стиль піксельного мистецтва, який викликає почуття ностальгії, одночасно ефективно передаючи свою історію та персонажів. Простота графіки забезпечує тісний зв’язок між гравцем і персонажами, що робить світ більш близьким. Використання кольору та дизайну в грі гарантує, що кожен персонаж є відмінним і таким, що запам’ятовується. Стиль піксельного мистецтва – це не просто повернення до ретро, а свідомий вибір для посилення оповіді та емоційного впливу гри [4, 5].

Deltarune, як і попередня гра Toby Fox, Undertale, використовує стиль піксель-арту. Візуально гра сповнена яскравих кольорів та унікальних персонажів. Головні події відбуваються в «Темному світі», який наповнений барвистими неоновими кольорами та елементами, що нагадують ранній інтернет, створюючи ностальгічну атмосферу. Візуальні ефекти та анімація персонажів є плавними та детальними, що дозволяє повністю зануритися в

ігровий світ. Всі персонажі мають унікальні костюми, що підкреслюють їхні особистості та ролі в грі [6, 7].

Отже, кольорова палітра повинна підбиратися так, щоб око гравця не втомлювалося, а єдиний унікальний стиль зберігався. Гравець повинен розрізняти рухи персонажа, постійно тримаючи його в своєму полі зору. Наприклад, у старих піксельних іграх, де технічні обмеження не дозволяли деталізації, проблему вирішували окремими деталями, як вуса у персонажа Маріо або червоні кросівки у Соніка з однойменних серій ігор [8, 9]. У ігровому застосунку цієї кваліфікаційної роботи повинні використовуватися прості спрайти персонажів та оточення, що враховують зазначене.

Далі звук. Саундтрек до гри Undertale, створений Тобі Фоксом, є важливим аспектом її чарівності та привабливості. Музика варіюється від привабливої та оптимістичної до неймовірно красивої, ідеально доповнюючи різні настрої та сценарії гри. Кожен персонаж має особливий лейтмотив, що допомагає підкреслити їх особистість і роль в історії. Такі треки, як «Hopes and Dreams» і «Megalovania», стали культовими, демонструючи силу звукового дизайну гри, щоб викликати емоції та покращити досвід гравця [10, 5].

Звуковий супровід Deltarune є однією з найсильніших сторін гри. Саундтрек, створений Toby Fox, складається з запам’ятовуваних та емоційно насичених композицій. Музика у грі варіюється від синтезаторних треків до електронних мелодій, що гармонійно доповнюють різні ігрові сцени. Саундтрек вражає своєю якістю і вмінням передати атмосферу кожної локації та події у грі [6, 7].

Отже, звуки доповнюють візуальну складову. Вони допомагають гравцю розрізняти ігрові дії та дають додаткову інформацію про ігрові локації [11]. Наприклад, локація Лавандове Місто у грі Pokemon Red and Blue стала легендарною завдяки своєму саундтреку, який створював атмосферу загадковості. Для ігрового застосунку цієї кваліфікаційної роботи повинна використовуватися проста музична композиція, створена власноруч.

Далі на черзі геймплей. Бойова система Undertale є інноваційною, вона

поєднує традиційну покрокову рольову механіку з елементами «кульового пекла». Гравці можуть вибрати, чи битися з ворогами, чи пощадити їх, при цьому кожен ворог потребує унікального підходу, щоб його пощадити. Ця механіка не тільки забезпечує різноманітність, але й поглиблює зв'язок гравця зі світом і персонажами гри. ШІ неігрових персонажів (NPC) розроблено для динамічної реакції на дії гравця, створюючи відчуття ефекту та інтерактивності. Бої з босами, зокрема, розширюють межі бойової системи гри, пропонуючи незабутні та складні зіткнення [4, 5].

Штучний інтелект (ШІ) у грі Deltarune зосереджується на різноманітних NPC (неігрових персонажах), які взаємодіють з гравцем через діалоги, битви та інші дії. NPC у грі мають добре прописані характери та унікальні діалоги, що додають глибини історії. Вороги в боях мають унікальні патерни атак, що поєднують елементи класичних RPG з bullet-hell механіками, що змушує гравців розробляти стратегії для перемоги. Однак, повторні сутички можуть іноді бути трохи нудними, оскільки відсутня можливість втечі з битви після виконання необхідних дій для «мирного» проходження [6, 7].

Управління персонажем повинно бути інтуїтивним і зручним для гравця, оскільки це є одним з найважливіших аспектів ігрового процесу. Зазвичай, гравець має такі функції управління, як пересування по рівню, взаємодія з об'єктами оточення або іншими персонажами, лікування персонажа, керування інвентарем та предметами. Ці дії повинні бути легко доступні та зрозумілі, тому їх слід закріпити на ті ж клавіші, що й у інших популярних іграх. Наприклад, пересування персонажа можна здійснювати за допомогою стрілок клавіатури або клавіш WASD, що дозволить гравцю одразу зрозуміти, які кнопки потрібно натискати.

У цьому ігровому застосунку варто обмежитися базовими діями, закріпленими на відповідні базові клавіші, щоб зробити управління простішим та інтуїтивним. Це також дозволить зменшити використання комп'ютерної миші та клавіш NumPad, що спрощує можливе портування гри на інші пристрої, окрім ПК, такі як консолі або мобільні телефони. Крім того,

враховуючи можливі технічні обмеження клавіатур користувачів (деякі клавіатури не мають блоку NumPad), таке рішення зробить гру доступнішою для більшої кількості гравців.

Штучний інтелект (ШІ) в іграх дозволяє неігровим персонажам (NPC) виконувати певні дії, щоб гравцю було цікавіше грати. Це може включати переміщення по рівню, взаємодію з ігровим персонажем через битви, діалоги та інші дії. Важливо, щоб ШІ був достатньо розумним, щоб створювати цікаві ситуації для гравця, але водночас не надто складним, щоб не перевантажувати систему та не створювати несправедливі умови для гравця.

У даному випадку ШІ повинен бути простим та відповідати за базові дії. Наприклад, NPC можуть пересуватися по заздалегідь визначеним маршрутам, реагувати на присутність ігрового персонажа, вступати з ним у діалоги або битви. Взаємодія NPC з ігровим персонажем повинна бути зрозумілою і передбачуваною, щоб гравець міг планувати свої дії і розуміти, як будуть реагувати персонажі на його дії.

Таким чином, управління персонажем та штучний інтелект NPC є ключовими елементами, що впливають на загальне враження від гри та її ігровий процес. Правильно налаштоване управління робить гру доступною та зручною для гравця, а продуманий ШІ забезпечує цікаві та захоплюючі взаємодії, роблячи гру більш динамічною та захоплюючою.

1.3 Вибір інструментів розробки

Для досягнення поставлених цілей і виконання вимог, визначених у розділі 1.2, потрібно ретельно підібрати відповідний інструментарій. Для створення ігор використовуються різноманітні ігрові рушії, кожен з яких має свої унікальні особливості, що дозволяють вибрати найбільш підходящий для конкретного проєкту. Вибір правильного рушія є критичним етапом, оскільки він визначає ефективність розробки, можливості графічного та аудіо

оформлення, а також загальну якість кінцевого продукту.

Задачею цієї кваліфікаційної роботи є створення ігрового застосунку у візуальному стилі піксель-арт. Це вимагає використання рушія, спеціалізованого на 2D графіці. Після аналізу різних варіантів було обрано рушій GameMaker, який має кілька значних переваг перед іншими.

По-перше, цей рушій повністю спеціалізується на 2D графіці і має численні приклади відомих ігор, створених на його базі, що демонструє його широкі можливості. Серед таких прикладів – проаналізовані в пункті 1.2 цієї роботи ігри Undertale та Deltarune. Ці ігри стали популярними завдяки своїй унікальній графіці, захоплюючому геймплею та емоційній глибині, що підтверджує потужність та універсальність рушія GameMaker для створення якісних проєктів.

По-друге, зручність та інтуїтивність інтерфейсу GameMaker є його важливою перевагою. Він пропонує елементи візуального програмування, що значно спрощує процес розробки, особливо для новачків. Крім того, власна мова програмування GameMaker, відома як GML (GameMaker Language), містить риси інших популярних мов, що значно прискорює адаптацію до неї, особливо в умовах обмеженого часу.

По-третє, рушій має вбудовані редактори для різних завдань, що дозволяє редагувати код і створювати графічні моделі без необхідності використання стороннього програмного забезпечення. Це включає редактори спрайтів, рівнів та анімацій, що робить процес розробки більш інтегрованим та зручним.

По-четверте, наявність детальної документації та безлічі уроків і ресурсів в інтернеті є ще однією значною перевагою GameMaker. Це допомагає розробникам швидко освоїти функціонал рушія і використовувати вже існуючі оптимальні рішення, що значно скорочує час на вивчення та впровадження нових технологій.

Додатково до рушія GameMaker, для виконання поставленої задачі будуть використовуватися програми Adobe Photoshop та FL Studio. Adobe

Photoshop необхідний для створення більш складних моделей, таких як фони локацій, оскільки він надає ширший функціонал для редагування зображень, ніж вбудований редактор рушія. Це дозволяє створювати високоякісні та деталізовані графічні елементи, що значно покращує загальне візуальне враження від гри.

FL Studio є музичним редактором, необхідним для створення саундтреку та іншого звукового супроводу. Ця програма дозволяє використовувати різноманітні VST-плагіни, які є модулями музичних інструментів і звукових ефектів. Використання FL Studio забезпечує високу якість звукового оформлення, що є важливим аспектом у створенні захоплюючої атмосфери гри.

Таким чином, комбінація рушія GameMaker та додаткових програмних інструментів Adobe Photoshop і FL Studio забезпечує повний набір засобів для створення якісного та цікавого ігрового застосунку у стилі піксель-арт. Використання цих інструментів дозволяє досягти високого рівня деталізації графіки та звукового супроводу, що є ключовими компонентами успішної гри.

2 ПРОЄКТУВАННЯ ІГРОВОГО ЗАСТОСУНКУ

2.1 Діаграма прецедентів

Перед початком розробки потрібно зрозуміти, які актори задіяні в ігровому процесі застосунку та які дії вони можуть виконувати.

Згідно з діаграмою прецедентів (рис. 2.1), в нас є 2 актори: гравець, що керує ігровим персонажем і штучний інтелект, що керує NPC.

Рисунок 2.1 – Діаграма прецедентів

Також є 3 основні базові дії:

– переміщення персонажа по рівню;

– атака на іншого персонажа;

– діалог з іншим персонажем.

Перші 2 дії можуть виконувати як гравець, так і штучний інтелект. Остання ж поки буде доступна лише гравцю, адже саме він активуватиме діалоги.

2.2 Діаграма класів

Згідно з принципами ООП, головними елементами програмної реалізації будуть об’єкти. Ігровий застосунок матиме загалом 5 основних об’єктів, з них

1 материнський, 2 дочірніх і 2 звичайних. Властивості кожного об’єкта, а також взаємодія між кожним з них вказані на UML діаграмі класів (рис. 2.2).

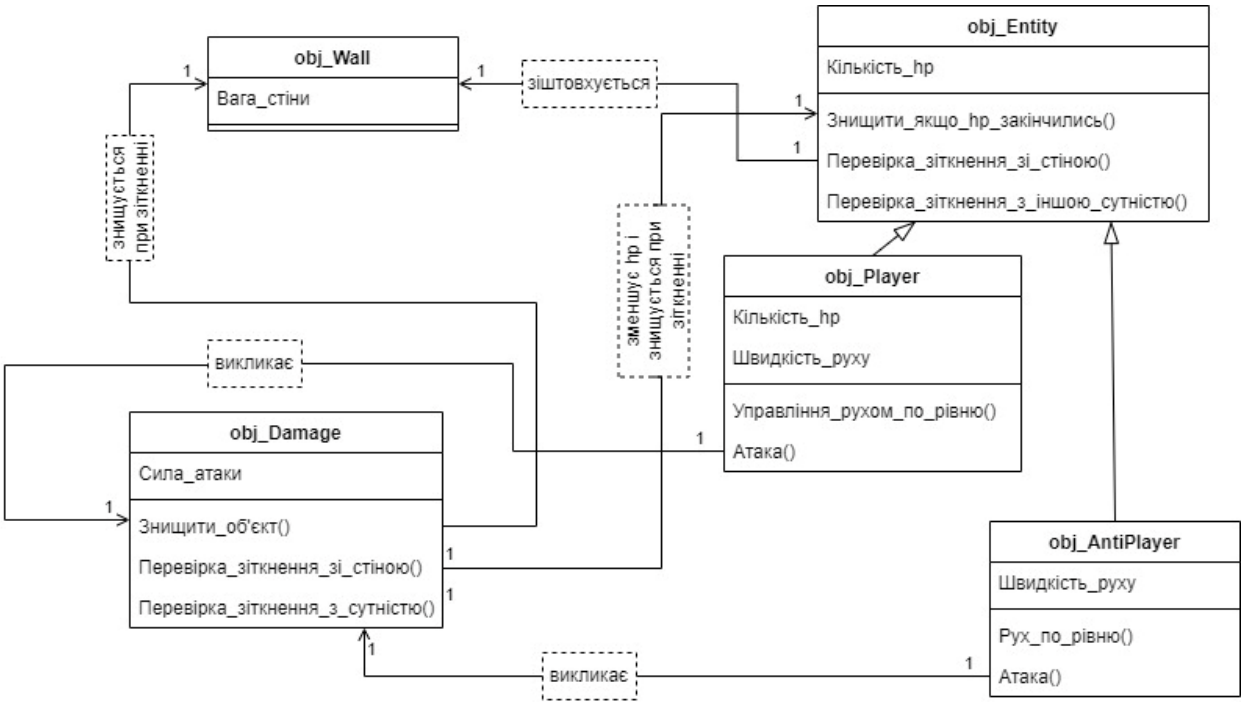

Рисунок 2.2 – UML діаграма класів

З огляду на особливості роботи в GameMaker, в програмній реалізації кожен об’єкт може мати різні параметри (окрім вказаних), які не враховуються в діаграмі під час проєктування.

Об’єкт obj_CaveWall є стіною. В собі він має лише один параметр, що відповідає за її вагу. Завдяки цьому параметру можна регулювати, чи можуть якісь сутності штовхати цей об’єкт.

Об’єкт obj_Entity є материнським об’єктом для всіх об’єктів-сутностей, що будуть з’являтися у грі. В собі він містить параметр, який відповідає за кількість hp (англ. hit points – очки здоров’я) кожної сутності, метод, який знищує об’єкт у випадку, якщо hp <= 0, методи, які перевіряють зіткнення зі стінами (obj_CaveWall) та іншими сутностями (obj_Entity). Цей об’єкт наслідують об’єкти obj_Player та obj_AntiPlayer.

Об’єкт obj_Player є ігровим персонажем. В ньому перевизначений параметр hp, адже гравець повинен мати більше здоров’я в порівнянні зі

звичайними сутностями. Також додано параметр, який відповідає за швидкість руху персонажа по ігровому рівню. В даному об’єкті також містяться метод для управління рухом персонажа з клавіатури, метод для атак проти інших сутностей, який викликає об’єкт obj_Damage.

Об’єкт obj_AntiPlayer аналогічний до obj_Player за винятком метода для руху по рівню, який не включає в себе управління з клавіатури, а також відсутності перевизначення параметра hp.

Об’єкт obj_Damage відповідає за нанесення шкоди сутностям. Містить у собі параметр, що відповідає за силу атаки, має 3 методи, які відповідають за знищення даного об’єкта, перевірку зіткнення зі стіною (obj_CaveWall), після чого викликається його знищення і перевірку зіткнення з сутністю (obj_Entity), після чого у сутності зменшується параметр hp, а obj_Damage знищує сам себе.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Реалізація ігрового застосунку

3.1.1 Підбір спрайтових листів

Перш, ніж приступити до програмної реалізації ігрового застосунку, потрібно підібрати спрайти, які будуть використовуватися в таких об’єктах, як obj_AntiPlayer, obj_CaveWall і так далі. Для цього в мережі інтернет було знайдено ряд безкоштовних спрайтових листів, автори котрих дозволяють вільне використання їхнього контенту.

Це можливо завдяки принципу “free to use”. Це означає, що певні матеріали або контент можуть бути використані безкоштовно, часто з незначними обмеженнями або взагалі без них. Це може стосуватися різних типів ліцензій, таких як Public Domain, Creative Commons або інших форм відкритих ліцензій.

Найчастіше такий контент розповсюджується без будь-яких умов, або ж з умовою вказання авторства. У випадку з такими продуктами, як ігри, це можна зробити в титрах. Існують також випадки, коли авторство позначається прямо в самій грі.

Рішення використовувати такі матеріали було прийнято одразу з кількох причин. По-перше, відсутність достатньої кількості навичок для відмалювання всіх елементів ігрового застосунку. По-друге, необхідність збереження часу задля того, аби вкластися у встановлені терміни.

Завдяки цьому рішенню було знято величезний пласт роботи зберігаючи якість майбутнього ігрового застосунку.

Знайдені спрайтові листи було відкореговано відповідно до потреб у програмі Adobe Photoshop. Вони будуть використані для obj_AntiPlayer (рис. 3.1) та obj_Damage (рис. 3.2).

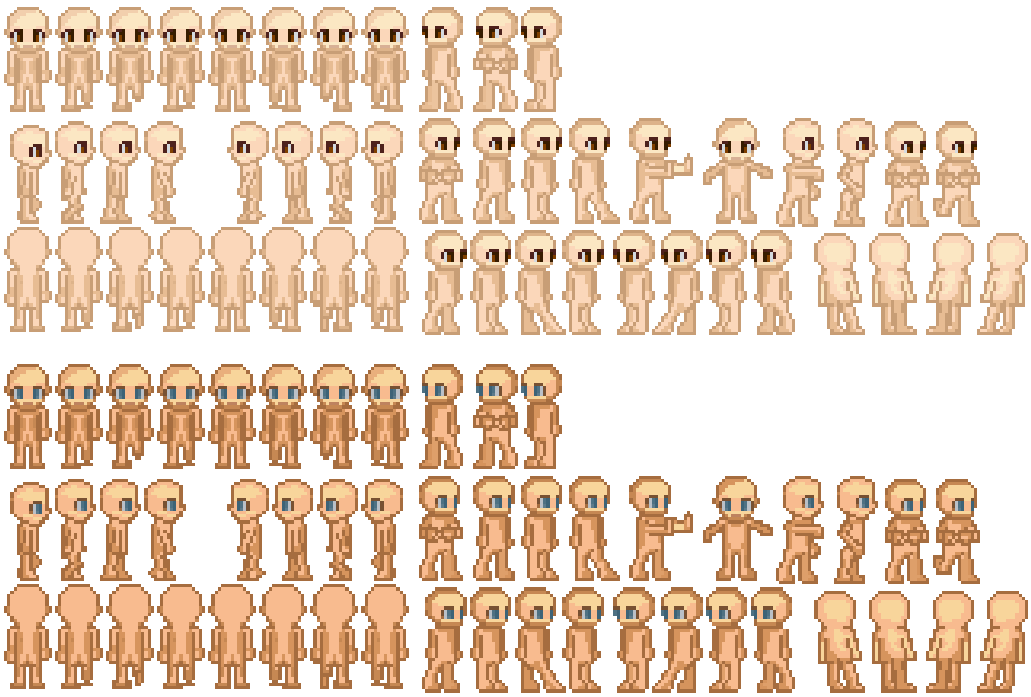

Рисунок 3.1 – Спрайтовий лист для obj_AntiPlayer

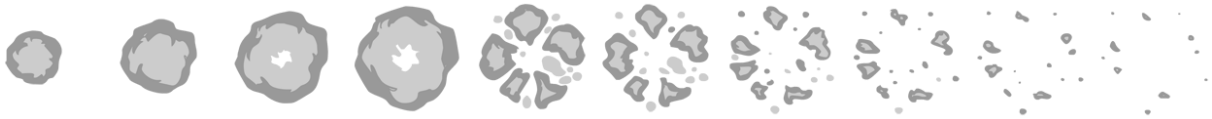

Рисунок 3.2 – Спрайтовий лист для obj_Damage

Спрайтові листи необхідні для створення анімацій, які відповідатимуть таким діям, як, наприклад, пересування по рівню.

Принцип дії цих анімацій аналогічний до мультиплікації. Створюється кілька кадрів, котрі відповідають за різні фази рухів. Після цього ці кадри розташовуються в необхідній послідовності і програються.

Спрайтові листи за своєю суттю призначені для того, аби великі об’єми спрайтів не губилися серед файлів і їх можна було швидко знайти за потреби. Крім того, це спрощує їх редагування, адже немає необхідності відкривати окремі файлики однієї анімації.

Зазвичай на одному спрайтовому листі розміщують всі спрайти, котрі стосуються певного персонажа або набору рухів, котрі він виконує.

Вбудований в GameMaker редактор зображень дозволяє вилучити з цих листів окремі спрайти для створення анімації (рис. 3.3).

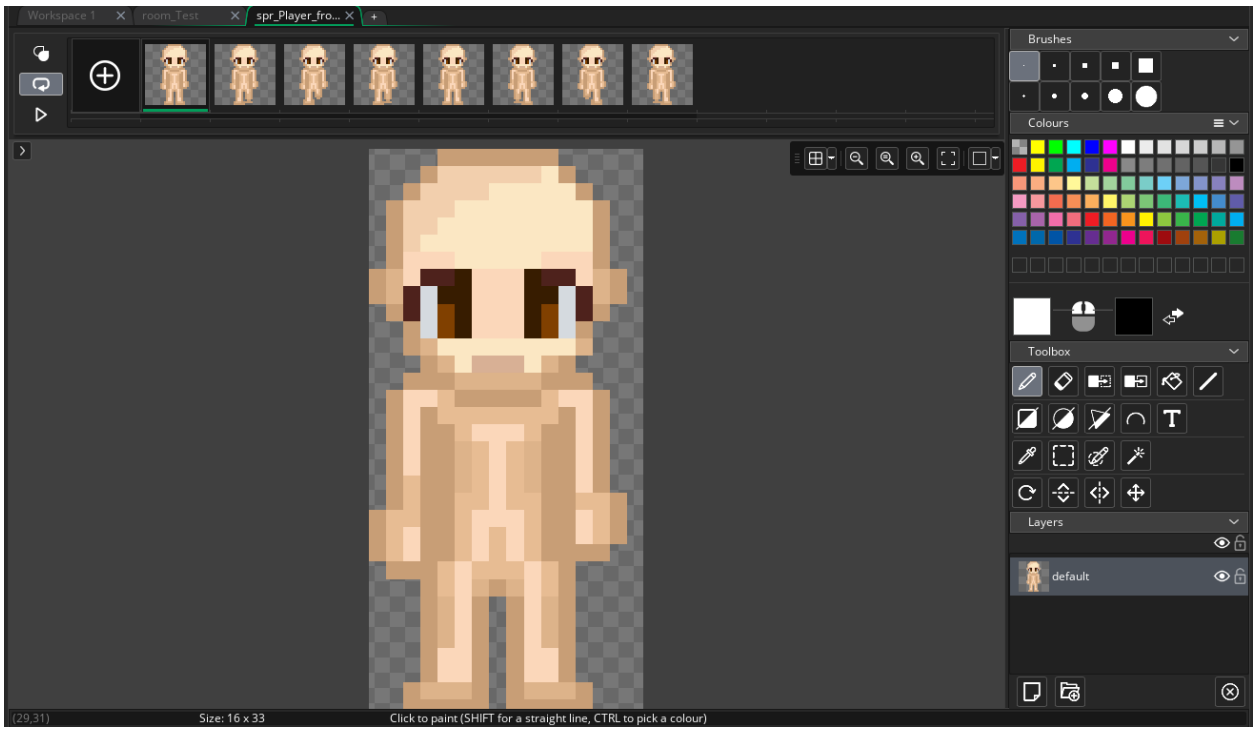

Рисунок 3.3 – Створення анімації для obj_AntiPlayer

Можливі ситуації, коли вилучені зі спрайтового листа анімації будуть не коректними. Це може відбуватися з кількох причин:

– спрайтовий лист оформлено без дотримання однакових відступів між кожним спрайтом, що призводить до некоректного імпорту окремих кадрів анімації;
– під час імпорту спрайтового листа, було неправильно встановлено межі спрайтів, що призводить до аналогічної з першим пунктом проблеми;
– в спрайтовому листі не вистачає кадрів для анімації, або ж вони розташовані в неправильному порядку.

Остання проблема зазвичай пов’язана з тим, що в спрайтових листах не прийнято дублювати однакові кадри задля зменшення ваги файлу і більш компактного розташування спрайтів на листі. Це, наприклад, може бути кадр, де персонаж просто стоїть.

Як результат, виходить анімація, де персонаж під час ходіння перестрибує з однієї ноги на іншу, пропускаючи момент, коли вони сходяться.

Це вирішується дублюванням необхідного кадру в редакторі спрайтів GameMaker і переміщенням його в необхідне місце.

3.1.2 Розробка власних персонажів

В ході роботи було розроблено та відмальовано унікального персонажа для obj_Player (рис. 3.4).

Рисунок 3.4 – Спрайтовий лист власноруч розробленого персонажу для obj_Player

Під час розробки власного персонажу було враховано інформацію, отриману під час аналізу вимог в 1 розділі цієї кваліфікаційної роботи.

Для цього були визначені найбільш помітні елементи образу, котрі відрізнятимуть головного персонажа зпоміж інших персонажів гри, а також споміж персонажів інших ігор.

Такими елементами є:

– зачіска, а саме високий пишний хвіст, чубчик, що роздвоюється на більшу частину та меншу та червоний колір волосся;
– червоний фракоподібний костюм з білими вставками в районі грудей, котрий гармоніює з кольоровим тоном, заданим зачіскою, але при цьому не зливається з нею в суцільну масу;
– коричневий пояс з невеликою жовтою пряжкою, котрий доповнює й урізноманітнює верхню частину образу;
– чорні штані та черевики;
– коричневі, під колір аналогічних в реальності шкіряних виробів, рукавички з відкритими пальцями.

Відмальовка відбувалась в кілька етапів. Спочатку формувався загальний силует тіла персонажа, що дає розуміння форми, меж різних частин і анатомії.

На цьому ж етапі формувалося обличчя. Задля кращого візуального сприйняття, обличчя було спрощене. Було намальовано виразні великі очі, котрі в подальшому з розвитком персонажа можуть служити в якості елемента, котрий передаватиме його настрій. У фронтальній проекції ніс було виділено завдяки затемненню, котре позначає тіні. В профілі ніс відмальований, як виступ у силуеті обличчя. Рот відсутній, адже, як вже було вказано раніше, емоції можна передати очима, а вільне місце надто обмежене для його виразного відображення. Риси обличчя здавалися б нерівними та відштовхуючими.

На наступному етапі відбувалася відмальовка волосся, як найбільш помітного і визначного елементу. Зачіска має бути впізнаваною, задавати кольоровий тон і певні риси характеру персонажа. При цьому вона не має бути

надто громіздкою і нереалістичною. Також вона має легко піддаватися перемальовуванню на різних кадрах.

Вибір пав на високий пишний хвіст і яскравий червоний колір. Таку зачіску легко малювати не дуже досвідченому художнику. Червоний колір, за своїми властивостями завжди привертає увагу. Завдяки цьому гравець не загубить персонажа на екрані і завжди триматиме його в полі зору, де б той не знаходився.

Розділений чубчик був доданий для урізноманітнення зачіски у фронтальній та профільних проекціях.

Наступним етапом стало відмалювання одягу. Загальний стиль надихався стилем одягу жанру стімпанк, завдяки чому вдалося створити цільний впізнаваний образ, котрий не загубиться на екрані гравця серед безлічі інших елементів, запам’ятовується завдяки нечисленним, але виразним деталям, передає певну нетиповість ігрового світу відносно нашого, підкреслює стать і можливий характер та можливості персонажа, що важливо для сприйняття гравцем майбутнього сюжету гри і ролі, котру він в ній відіграє.

Окремо можна виділити нижню частину тіла. Чорний колір, котрий було обрано для штанів та взуття, гарно поєднується з червоним. В той же час він урізноманітнює кольорову гамму. У випадку, якщо в якійсь з майбутніх локацій будуть домінувати червоні кольори, чорні штани можуть зіграти таку саму роль маркера, як і волосся на інших локаціях.

Втім, на майбутнє варто уникати активного використання такої самої палітри кольорів, як у ігрового персонажа. Також можлива зміна маркерних елементів моделі залежно від локації. Однак це ускладнює роботу і може спричинити те, що гравець сприйматиме оновлені моделі, як різних персонажів.

Аби персонаж не виглядав пласким, на всіх етапах було відмальоване затінення або засвітлення окремих частин моделі персонажа. Це досягається

завдяки затемненню чи освітленню основного кольору. в тих місцях, де у справжнього об’єкту були б згини та викривлення.

Весь процес відбувався в програмі Adobe Photoshop.

Ще одним розробленим власноруч персонажем є вигадане створіння, котре має пластичну структуру тіла (рис. 3.5).

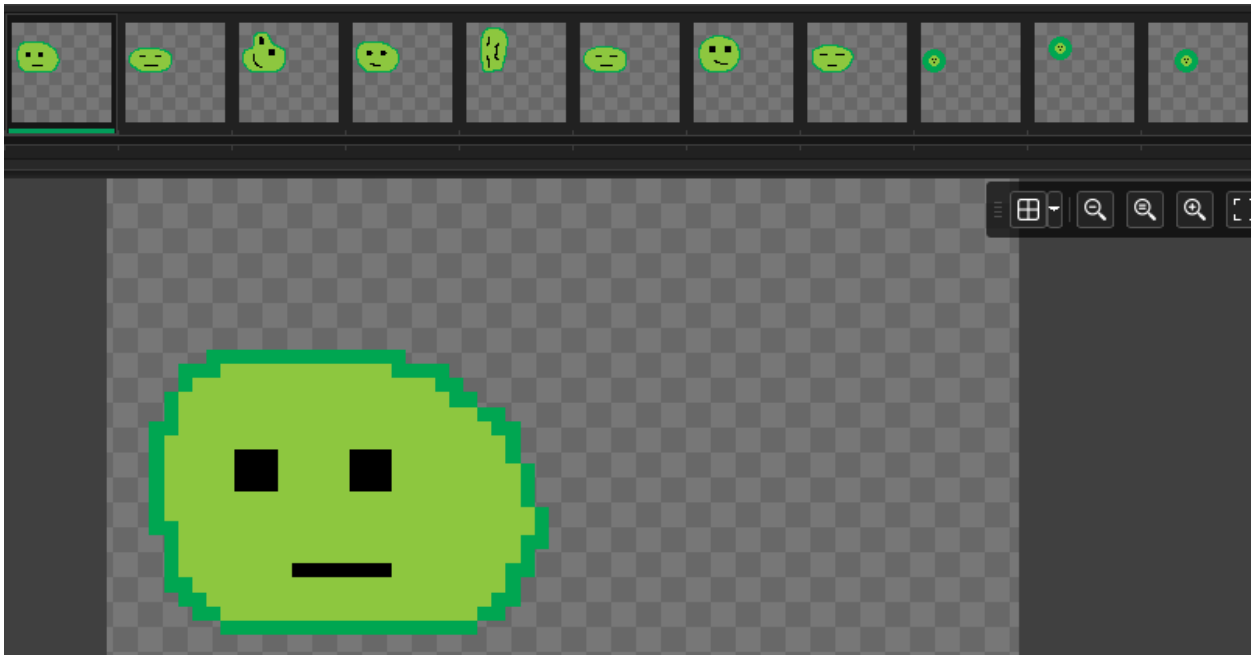

Рисунок 3.5 – Другий розроблений власноруч персонаж

У зв’язку з особливостями тіла і анатомії персонажа, необхідності використовувати спрайтовий лист для нього не виявилось. Він має бути створінням, котре постійно змінює свою форму і виглядає дивакувато. Тож для відмальовки було достатньо функціоналу редактору спрайтів GameMaker.

За рядом ознак він нагадує слаймів із фентезійних ігор та аніме. Тож і колір його було обрано відповідний, зелений.

Нерівний контур, відсутність тіней і примітивне обличчя підкреслюють незвичність цього персонажа і вводить гравця в певну невизначеність очікувань і подальших дій.

Завдяки таким ознакам, він впадає в око і запам’ятовується. Кожна його поява визначатиме певну значимість цього персонажа і створить підґрунтя до

можливих майбутніх сюжетних теорій та здогадок. Або ж просто трохи повеселить гравця.

3.1.3 Елементи оточення

Невід'ємною частиною кожної гри є оточення, в якому і відбуватимуться всі ігрові дії. Задля економії часу і ресурсів було вирішено створювати локацію в стилістиці підземелля.

Підземелля є одним з найбільш впізнаваних і часто використовуваних локацій в іграх, пов'язаних з тематикою фентезі. У свідомості гравців підземелля завжди асоціюються з небезпеками у вигляді підземних мешканців та монстрів і нагород у вигляді скринь зі скарбами, печер з цінними ігровими артефактами, чи спорядженням.

Гравець може швидше сконцентруватися на ігровому процесі, адже вже приблизно знає, чого чекати від цієї локації. Він не розгубиться на старті гри від відчуття нерозуміння, що саме тут відбувається і навіщо він тут. Кожне ігрове підземелля завершується виходом, тож першочергова задача гравця, котра одразу приходить на думку – знайти цей вихід.

Завдяки особливостям таких локацій, можна не перейматися різного роду мілкими деталями оточення і їх промальовкою, як це могло бути, наприклад, на локації з містом. У разі нестачі художнього досвіду це може відіграти ключову роль у можливості реалізувати ігрову локацію.

Простота також проявляється у підборі необхідних кольорів. Сірий колір є домінуючим, адже він найкраще асоціюється з кам'яними поверхнями. Аби це не виглядало, як однотонна бетонна стіна, необхідно лише відмалювати ребристу хвилясту поверхню, котра має виступи, що відкидають тінь.

Об'єкти освітлення (лампи, смолоскипи і т.д.) в локації можна не відмальовувати. Скоріш за все, гравець не задаватиметься питанням їх

наявності. А якщо такі думки виникнуть, вже відмальовані тіні на стінах можуть наштовхнути його на здогадку, що джерело освітлення знаходиться десь зверху. Цьому також сприятиме розташування тіней на модельці персонажа.

Ще однією вимогою для таких спрайтів є безшовність. Цієї вимоги треба дотримуватися через те, що один і той самий спрайт буде дублюватися, формуючи суцільну площу. Якщо в спрайті буде відмальовано якісь деталі, котрі якось позначають її межі, його дублювання буде надто помітним. Це буде виглядати, в кращому випадку, як кам'яні блоки, розташовані один поруч з іншим, а в гіршому – взагалі вибиватиме гравця з атмосфери локації. Її стіни так і кричатимуть, що тут все нереальне і штучне.

Таким чином, для об'єкта obj_CaveWall за допомогою вбудованого в GameMaker редактора було намальовано простий спрайт (рис. 3.6), котрий має обриси стіни печери.

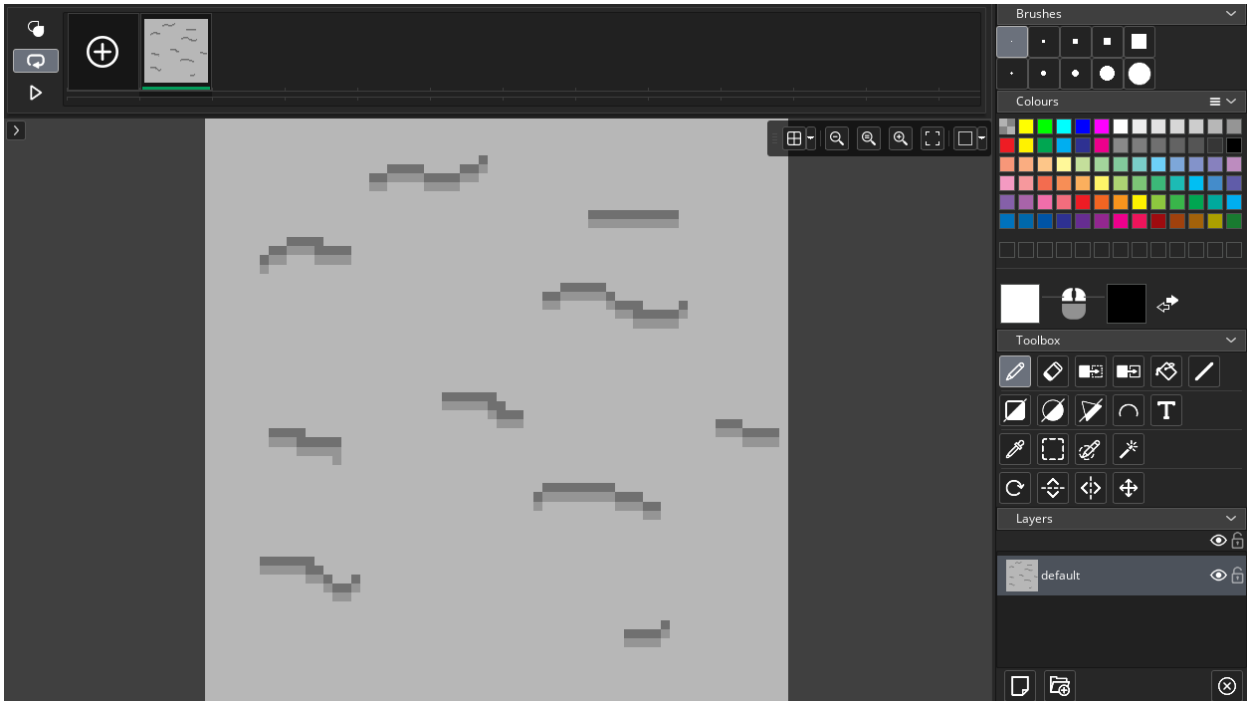

Рисунок 3.6 – Спрайт для obj_CaveWall

Аналогічно було створено спрайт для підлоги (рис. 3.7). Аби було видно межу, де закінчується стіна, а де вже починається підлога, вони мають бути

різних відтінків. Тому для підлоги було обрано більш темний сірий колір.

Виступи зображено менш контрастними, ніж на стінах. Це зумовлено тим, що по підлозі постійно хтось ходить, а значить, полірує її поверхню.

Як вже було написано раніше, маємо на думці, що джерело освітлення знаходиться десь зверху. З огляду на це, виступи не відкидають тіні.

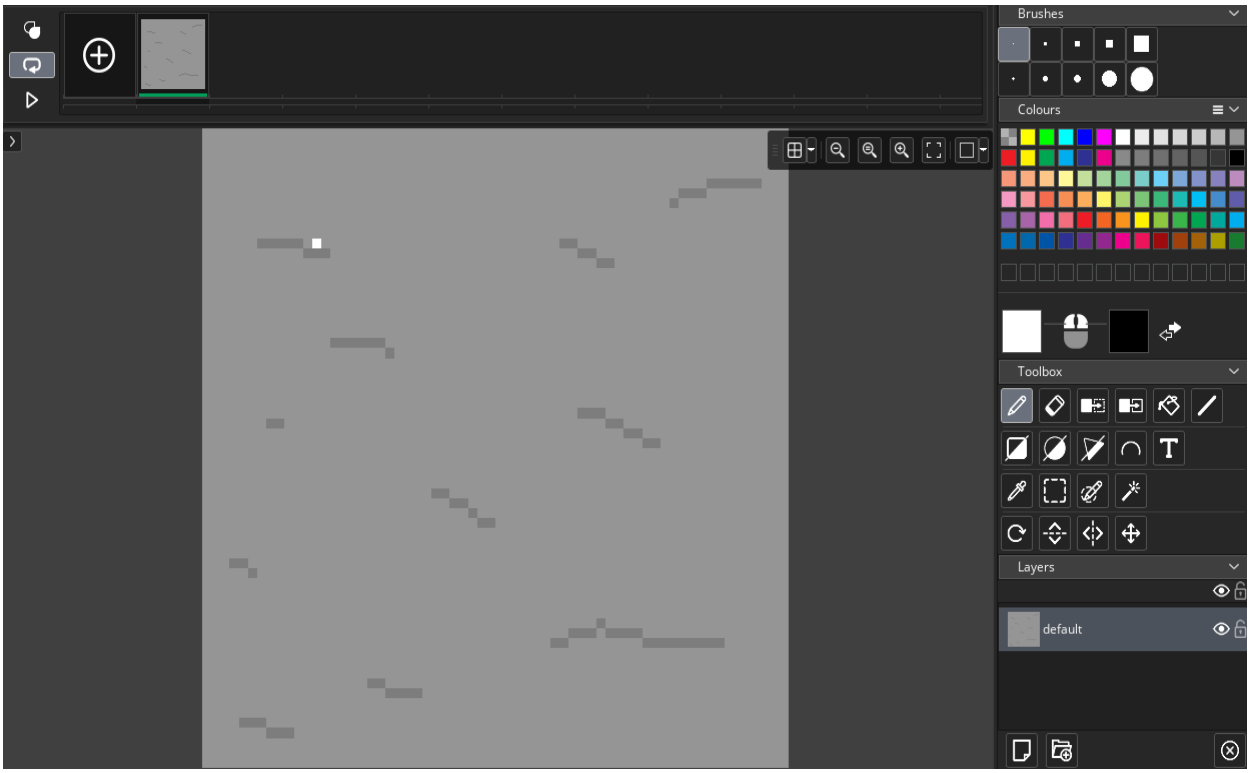

Рисунок 3.7 – Спрайт для підлоги (Tiles_1)

Таким чином було створено необхідні спрайти оточення, після чого вже можна приступити до наступних етапів розробки.

3.1.4 Створення об’єктів

Для створення нових об’єктів ігровий рушій GameMaker надає відповідний інтерфейс, куди винесені всі основні параметри (рис. 3.8).

Тут об’єкту призначається назва, відповідний спрайт (якщо він

необхідний), налаштовується колізія та фізика. Також в цій панелі можна встановити дочірні об’єкти і провести інші важливі налаштування.

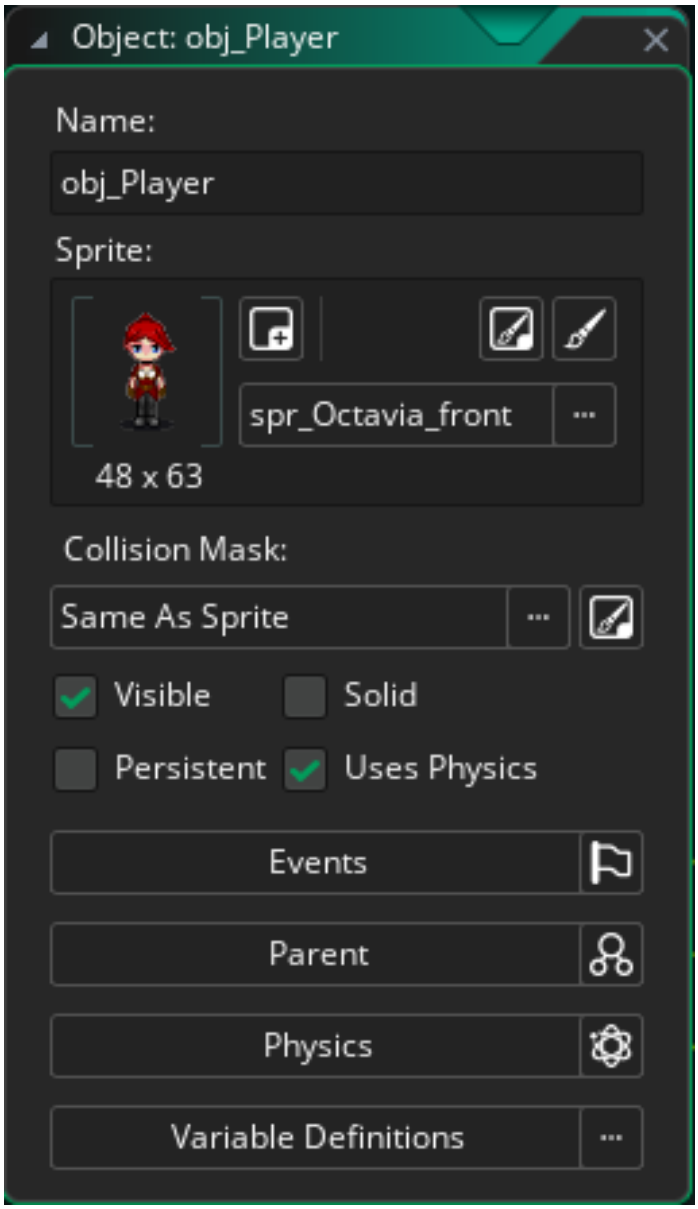

Рисунок 3.8 – Панель редагування об’єкта

В першу чергу було створено об’єкт obj_Entity. Згідно логіці цей об’єкт має бути батьківським для всіх сутностей гри (гравець, NPC, монстри і т.д.). Він не має визначеного спрайту та фізики.

В ньому визначаються методи, спільні для всіх сутностей. Такими методами є поведінка при зіткненні з різними об’єктами накшталт стін, зокрема, аби персонаж впирався в стіну і не змінював при цьому свою

орієнтацію у просторі (не обертався). також тут визначаються базові параметри для здоров’я сутностей і того, аби ті знищувалися, коли це здоров’я закінчується.

Далі створено об’єкт ігрового персонажа. Йому було призначено спрайт spr_Octavia_front, котрий складається з першого рядка розробленого раніше власноруч спрайтового листа.

Колізія була відредагована чітко під розміри кожного кадру анімації (рис. 3.9). Якщо не приділити увагу цьому моменту, в подальшому можуть виникнути проблеми, коли персонаж якоюсь частиною свого тіла входить в інші об’єкти, або ж атака, чи пастка, котра знаходиться в боці від персонажа, спрацьовуватиме на ньому.

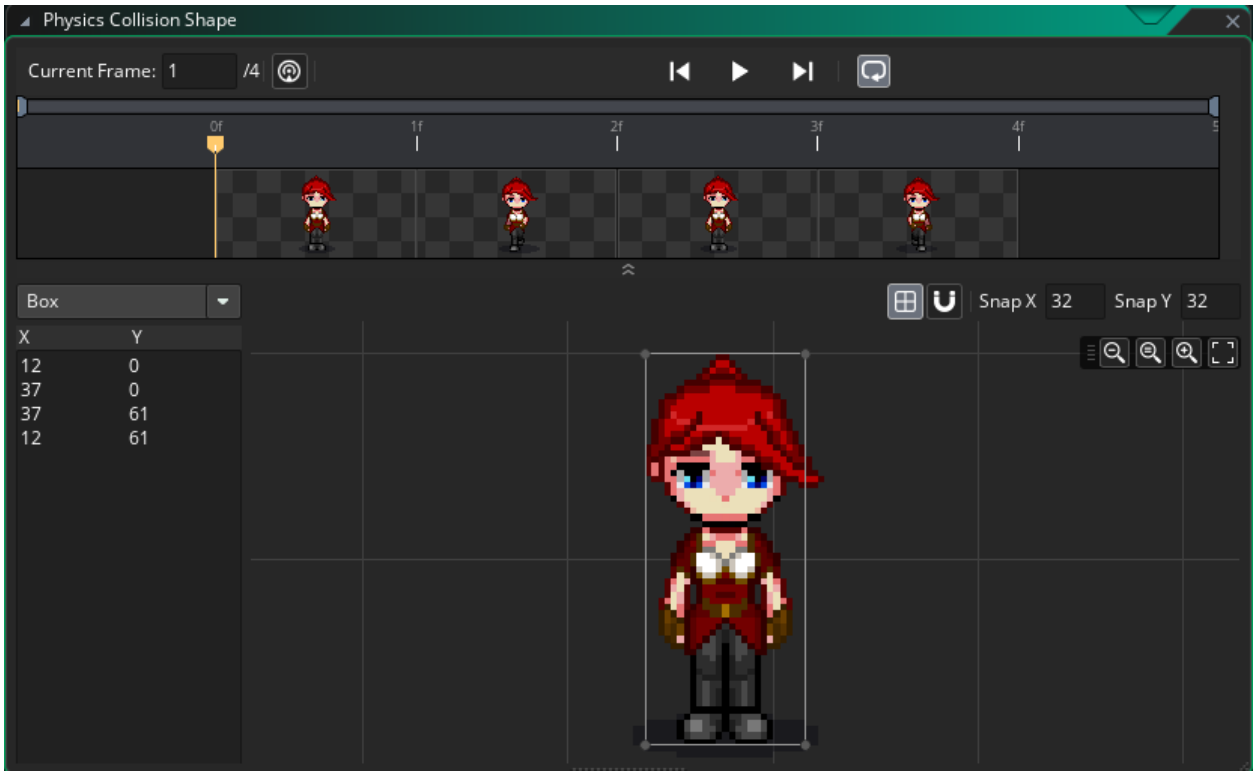

Рисунок 3.9 – Редагування колізії для obj_Player

Наступним було створено об’єкт obj_AntiPlayer, котрий є ворогом для гравця. Його було налаштовано аналогічно до obj_Player. Цьому об’єкту було присвоєно спрайт spr_AntuPlayer (рис. 3.10).

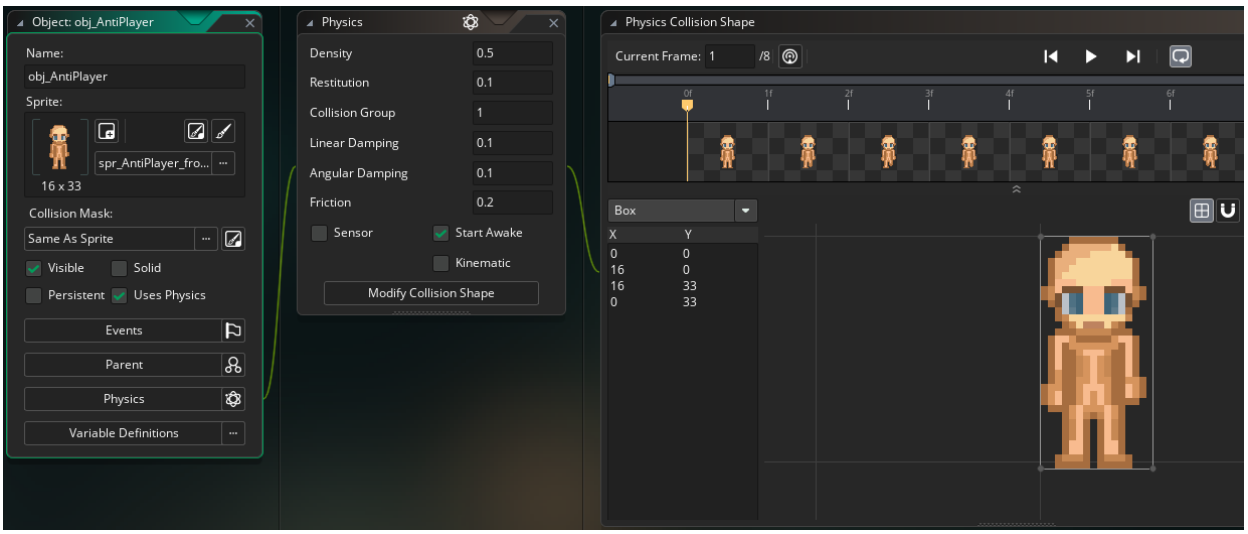

Рисунок 3.10 – Об’єкт obj_AntiPlayer з призначеним йому спрайтом і проведеними налаштуваннями

Також було створено об’єкт obj_Damage з присвоєнням йому спрайту spr_Damage. Він необхідний для реалізації бойової системи і її візуалізації (рис. 3.11).

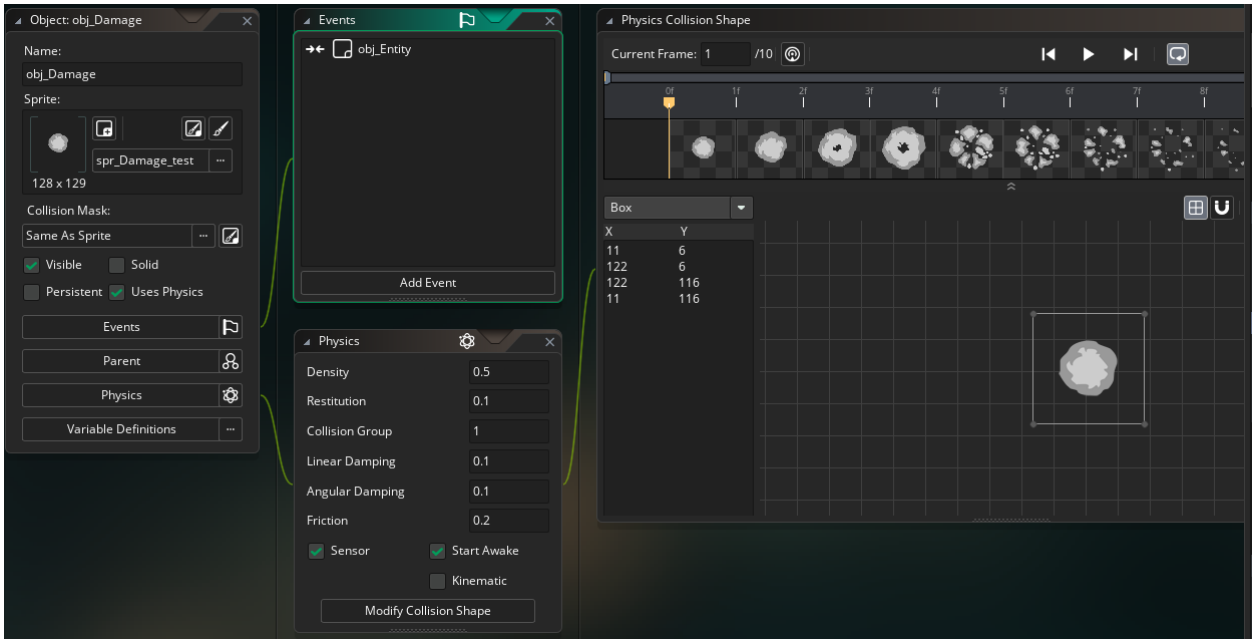

Рисунок 3.11 – Об’єкт obj_Damage і його налаштування

Задля того, аби персонаж не впирався головою у стіну зверху локації, а візуально наче підходив до неї, було реалізовано 2 різних об’єкти для стін:

obj_CaveWall_side та obj_CaveWall_up.

Перший – для вертикальних коридорів. Він має колізію на весь об’єм спрайту.

Другий – для горизонтальних. В ньому колізію було піднято знизу до середини спрайту. Завдяки цій дії вдалося досягти ефекту, як на рисунку 3.12.

В цих двох об’єктах параметр ваги було встановлено на 0. За логікою рушія це значить, що об’єкт має нескінченну вагу і його неможливо зрушити з місця. При зміні цього параметру на більші значення, персонаж зміг би штовхати стіни, що дозволило б виходити за межі визначених коридорів і повністю б ламало логіку ігрового процесу.

Рисунок 3.12 – Демонстрація ефекту від зміни колізії стіни

Ще одним об’єктом став анімований персонаж, описаний в розділі 3.1.2, котрий за своєю реалізацією не є сутністю, тож і не успадковує характеристики об’єкту obj_Entity. Єдиною його задачею буде діалог з користувачем, коли той наблизить до нього ігрового персонажа і натисне кнопку взаємодії (Enter).

Аби гравець розумів, що йому треба тиснути, над персонажем розташована підказка у вигляді діалогової хмарки з назвою клавіші.

Згідно діаграмі класів, описаній у розділі 2.2 цієї кваліфікаційної роботи, було створено наступні об’єкти:

– obj_CaveWall_side;

– obj_CaveWall_up;

– obj_Entity;

– obj_Player;

– obj_AntiPlayer;

– obj_Damage.

3.1.5 Збір ігрового рівня

Наступним етапом став збір ігрового рівня, котрий і буде майданчиком для тестування та гри (рис. 3.13). Для цього була створена кімната room_Test.

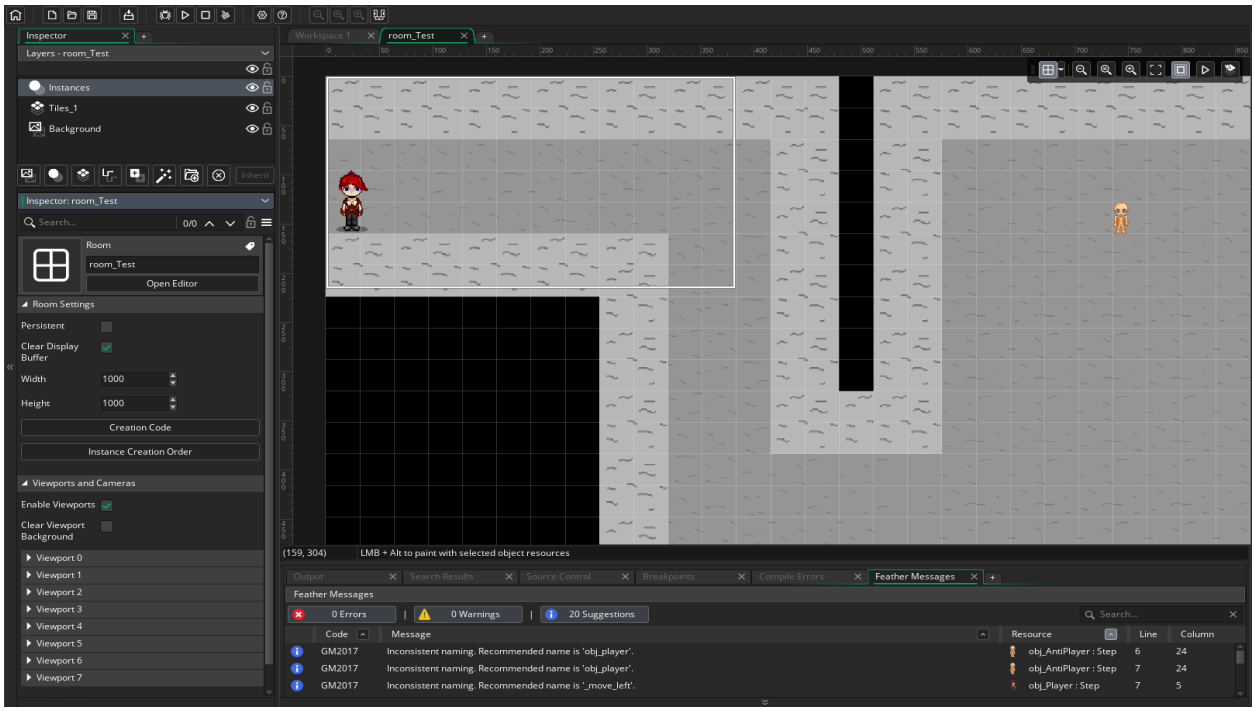

Рисунок 3.13 – Збір рівня room_Test

Для цього зі створених раніше об’єктів було зібрано локацію, котра складається з коридорів та великою кімнатою, де буде відбуватися основна

битва.

При створенні локації важливо дотриматися правильного порядку різних об’єктів і розмірів коридорів, аби персонаж не застрягав в занадто вузьких місцях, або прогалинах колізії.

Коридори не мають бути занадто довгими, чи заплутаними. Через це гравець може занудитись їх проходити.

Підлога локації створена за допомогою шару Tiles_1. Йому було призначено набір клітин ts_CaveGround, створений на основі спрайту spr_CaveGround.

За допомогою кисті, було накладено текстуру підлоги там, де це необхідно (рис. 3.14).

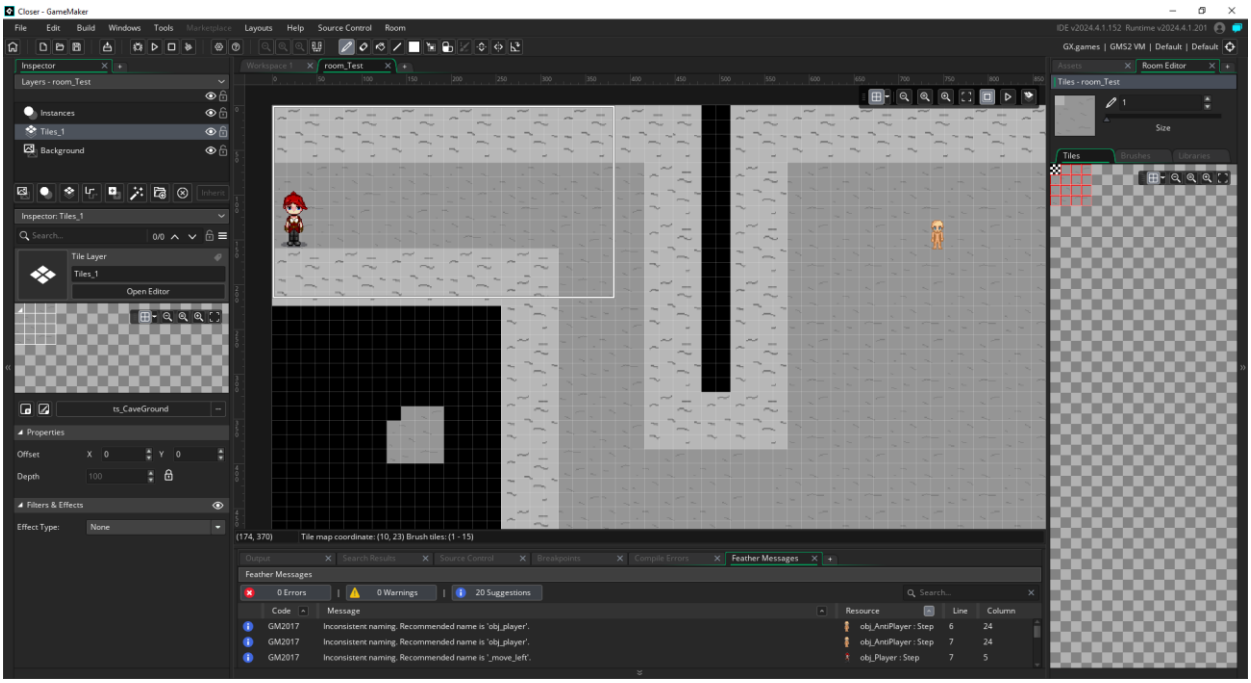

Рисунок 3.14 – Накладання текстури на підлогу локації

Клітини за межами стін були видалені, аби гравець розумів, що там нічого немає.

Для того, аби все це відображалося на екрані, необхідно створити камеру. Як і у випадку з фільмами, камера надає зображення того, що автор хоче показати глядачу. Вона визначає межі поля видимості і, як наслідок, того,

що може гравець одночасно побачити на своєму моніторі.

В рушії GameMaker камера є одним з розділів налаштування кімнати. Необхідно задати розміри камери, її видимість, а також інші параметри (рис. 3.15). Межі, котрі покриває камера, видно на робочому просторі.

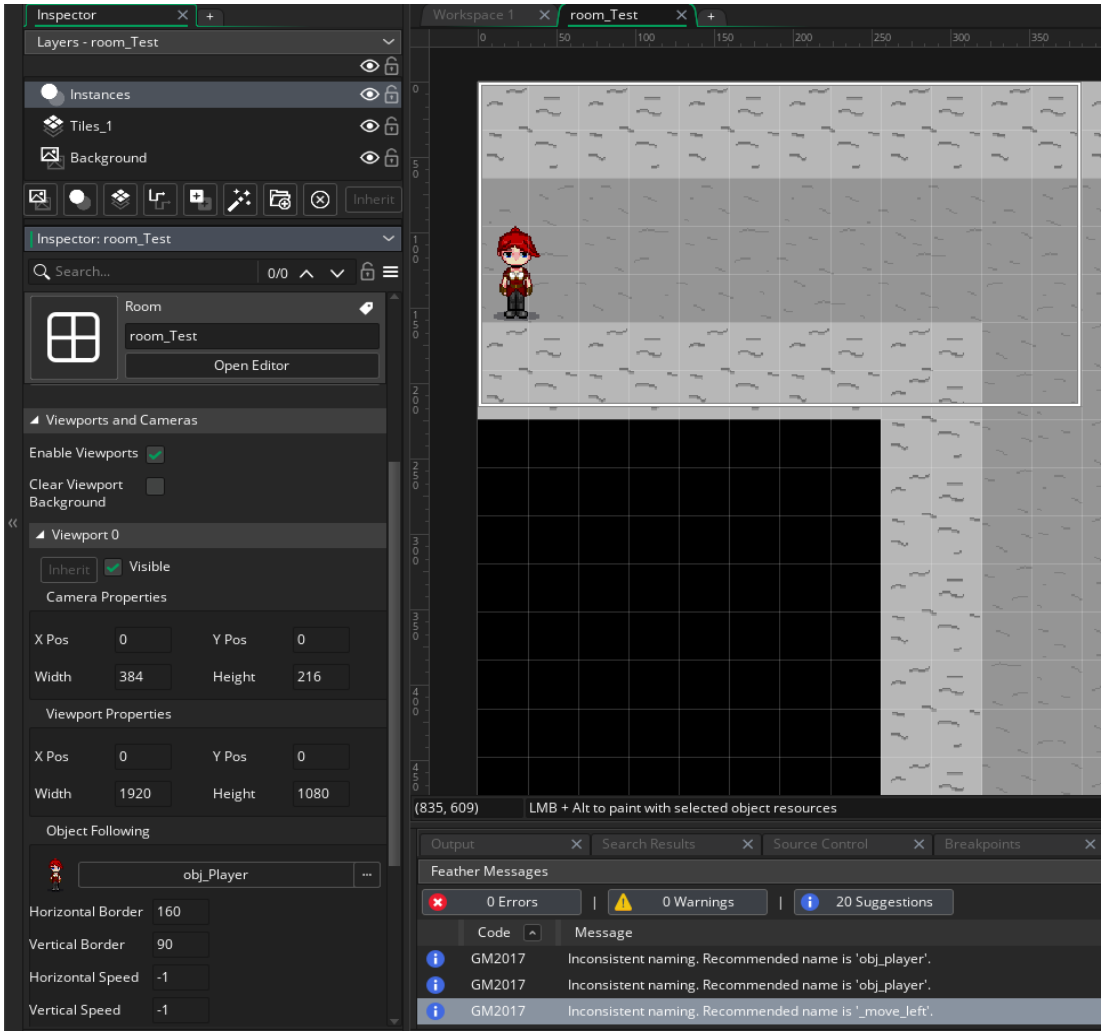

Рисунок 3.15 – Налаштування камери та відображення області, котру вона передає

Аби при переміщенні ігровий персонаж не виходив за межі поля зору камери, необхідно, аби камера слідувала за ним. Для цього в розділі Object Following було визначено, за яким об’єктом треба слідкувати, межі, в котрих камера повинна починати свій рух і швидкість цього руху.

В межах однієї кімнати можна створити одночасно кілька камер для різних цілей. Однак для задач цієї кваліфікаційної роботи вистачить і однієї.

3.1.6 Створення музичного супроводу

Аби ігровий застосунок не був мовчазним і аби дотриматися описаним в попередніх розділах вимог, необхідно було розробити власну музичну композицію, котра б грала під час запуску рівня. У якості такого музичного супроводу в програмі FL Studio було створено аудіокомпозицію (рис. 3.16).

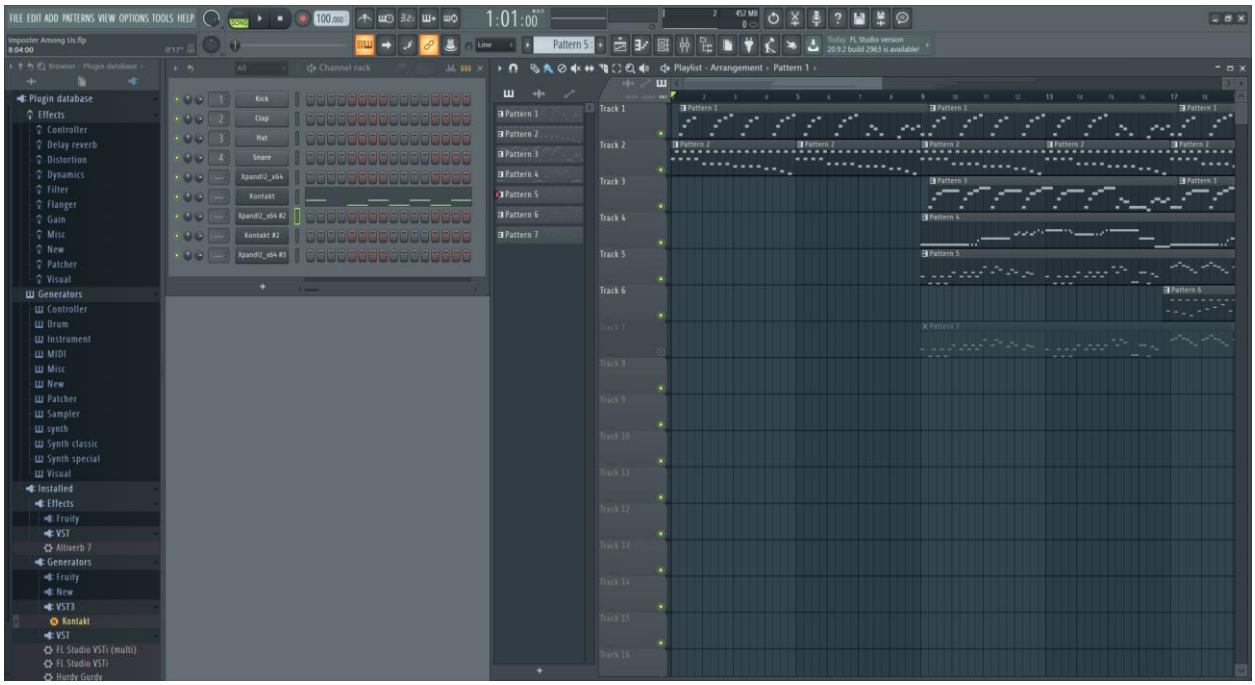

Рисунок 3.16 – Відкритий в FL Studio аудіопроєкт

Печери зазвичай є загадковими просторими локаціями з гарною акустикою. Тож і мелодія повинна бути спокійною, але не надто повільною. Частенько в печерах чутно, як крапає вода, десь вделечині скочується каміння через рухи якихось підземних створінь, або інші незрозумілі звуки.

Спочатку було задано основний ритм, котрий визначає загальний настрій композиції. Це простий повторюваний ритм, не навантажений надмірною кількістю нот.

Після цього, на основі нього було додано ще один ритм, котрий доповнює попередній. Він додає звуку фарб та урізноманітнює звучання.

Для того, аби мелодія не набридала під час бродіння по рівню, до неї

додано ще кілька простих ритмів.

Коли патерни нот створено, вони розташовані в правильній послідовності, настав момент підбору інструментів. З огляду на визначені раніше вимоги, вибір пав на піаніно, контрабас і синтетичний бас.

Завдяки такому вибору мелодія почала відповідати задуму і формувати в уяві необхідні образи, що безумовно допомвнює візуальну частину рівня.

3.1.7 Написання коду

Аби створені об’єкти правильно взаємодіяли між собою, музика відігравалась, користувач міг керувати персонажем, необхідно приступити до написання коду.

Написання коду відбувається для кожного об’єкту у вкладці Events (рис. 3.17).

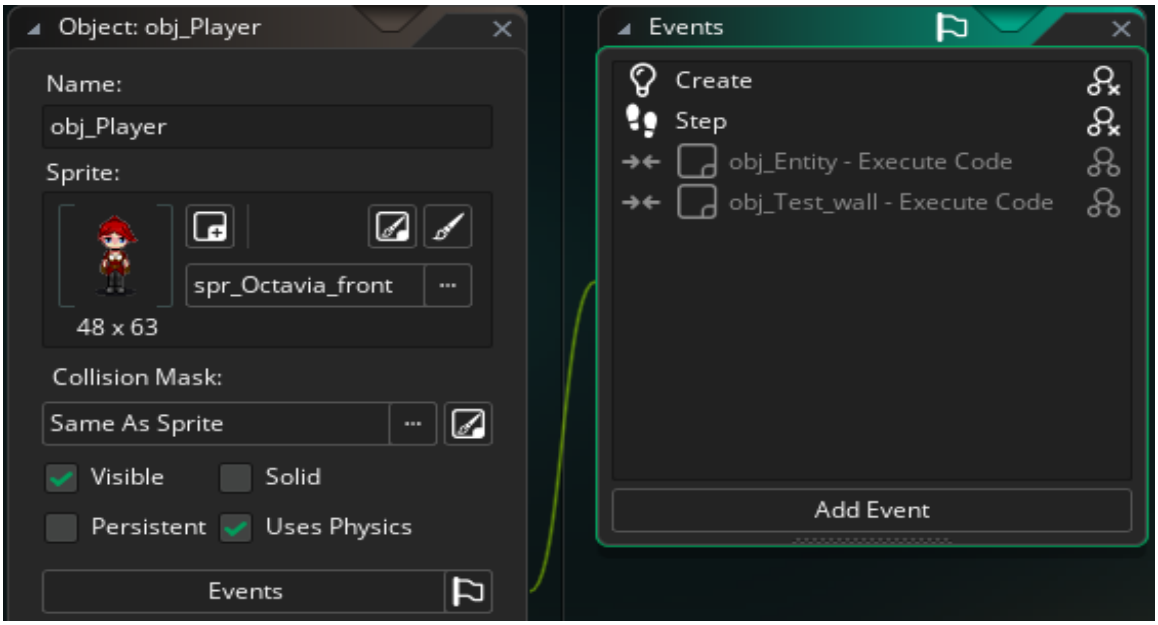

Рисунок 3.17 – Вкладка Events

Особливості рушія GameMaker дозволяють створювати код як візуальними засобами (рис. 3.18), так і прописувати його напряму.

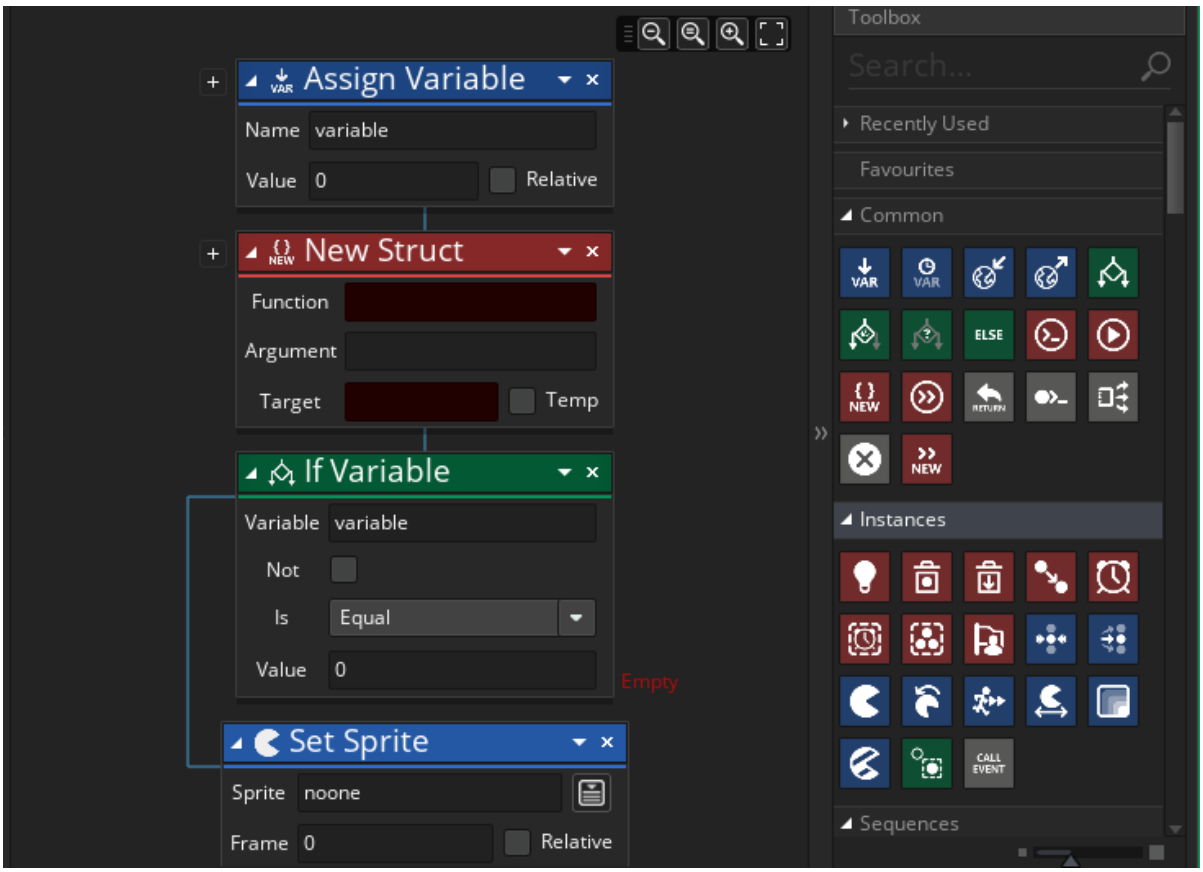

Рисунок 3.18 – Приклад графічного способу створення коду гри

Першим об’єктом, котрому почалося прописування коду, став об’єкт obj_Player.

В розділі Events було створено подію Create. Вона відповідає за дії, котрі програма повинна виконати під час створення цього об’єкту.

В редакторі цієї події було створено блок редактору коду, де вручну буде прописуватися необхідний код (рис. 3.19).

Такий вибір пов’язаний з тим, що так можна швидше і з власними коментарями внести необхідні змінні та функції.

```
1 event_inherited();
2 hp = 10;
3 //необхідні для проекту користувацькі змінні
4 player_speed = 2;
5
6 //інші важливості
7 image_speed = 0; //швидкість анімацій за замовчуванням
```

Рисунок 3.19 – Код події Create об’єкта obj_Player

Функція event_inherited(); у GameMaker є важливою частиною обробки подій у системі об’єктно-орієнтованого програмування, яку використовує цей рушій. Ця функція дозволяє викликати код події з базового (батьківського) об’єкта в спадковому (дочірньому) об’єкті. Це особливо корисно, коли дочірній об’єкт повинен виконати специфічні дії у відповідь на подію, але також зберегти поведінку, визначену в батьківському об’єкті.

В даному випадку ми викликаємо зміст об’єкта obj_Entity, котрий є батьківським для всіх сутностей.

У другому рядку коду перевизначається змінна hp, котрій в батьківському об’єкті призначене значення за замовчуванням. таким чином було збільшено кількость очків здоров’я у гравця.

Змінна player_speed відповідає за швидкість руху персонажу, котра вимірюється в умовних одиницях рушія.

Змінна image_speed відповідає за те, як швидко відігрується анімація. В попередніх розділах цієї кваліфікаційної роботи було вказано, що об’єкту obj_Player призначено спрайт spt_Octavia_front, котрий складається з 4 кадрів. Призначивши цій змінній значення 0, анімація була зупинена. Завдяки цьому персонаж не ходить на місці.

Далі було створено подію step, котра відповідає за переміщення персонажа (рис. 3.20).

Як і в попередньому коді, в цьому було додано функцію event_inherited();.

Після цього було визначено 4 змінних, котрі відповідають за відстеження натискання клавіш руху завдяки функції рушія keyboard_check(). Кожна змінна відповідає за конкретну стрілку на клавіатурі користувача.

```
1  event_inherited();
2  //рухи по рівню
3  var move_left = keyboard_check(vk_left);
4  var move_right = keyboard_check(vk_right);
5  var move_up = keyboard_check(vk_up);
6  var move_down = keyboard_check(vk_down);
```

Рисунок 3.20 – Змінні в події Step

Після змінних було прописано код, що відповідає за обробку натискань визначених клавіш.

```
if (move_left){
    phy_position_x -= player_speed;
    sprite_index = spr_Octavia_sideLeft;
    image_speed = 1;
}
if (move_right){
    phy_position_x += player_speed;
    sprite_index = spr_Octavia_sideRight;
    image_speed = 1;
}
if (move_up){
    phy_position_y -= player_speed;
    sprite_index = spr_Octavia_back;
    image_speed = 1;
}
if (move_down){
    phy_position_y += player_speed;
    sprite_index = spr_Octavia_front;
    image_speed = 1;
}
if (!move_left && !move_right && !move_up && !move_down){
    image_speed = 0;
    image_index = 0;
}
```

Рисунок 3.21 – Код, прописаний в події Step об’єкта obj_Player

Згідно коду, якщо затиснута клавіша «вліво», від поточної позиції гравця по осі x (phy_position_x) віднімається стільки одиниць, скільки вказано у змінній player_speed, котра відповідає за швидкість персонажа.

При цьому об’єкту призначається спрайт spr_Octavia_sideLeft, котрий відповідає за анімацію руху персонажа вліво.

Параметр image_speed запускає саму анімацію.

Аналогічні дії прописано для руху вверх, вправо та прямо.

Якщо жодна з визначених клавіш на поточний момент не натиснута, анімація зупиняється і встановляється на перший (image_index = 0; де 0 – індекс масиву) кадр.

Наступна частина коду відповідає за атаки головного персонажа на ворогів (рис. 3.22).

```
//атаки
if (keyboard_check_pressed(ord(" "))){
    instance_create_depth(x+5, y+20, 0, obj_Damage);
}
```

Рисунок 3.22 – Код, що відповідає за атаки

Відповідно до нього, якщо гравець натискає клавішу «пробіл», що обробляється за допомогою функції рушія keyboard_check_pressed(), на певній відстані від нього створюється об’єкт obj_Damage, котрий і наноситиме шкоду ворогам.

Тепер розглянемо об’єкт obj_AntiPlayer. В події Create ввнесено код (рис. 3.23), що наслідує код батьківського об’єкта та встановлює швидкість персонажа на 0,25.

```
event_inherited();
antiplayer_speed = 0.25;
```

Рисунок 3.23 – Код події Create обйєкта obj_AntiPlayer

В події Step додано код (рис. 3.24), котрий відповідає за рухи цього NPC. На відміну від гравця, де об’єкт відслідковує натискання клавіш, тут об’єкт відслідковує положення гравця і переміщується в його напрямку зі встановленою швидкістю.

```
event_inherited();
phy_position_x += sign(obj_Player.x - x) * antiplayer_speed;
phy_position_y += sign(obj_Player.y - y) * antiplayer_speed;
```

Рисунок 3.24 – Код в події Step об’єкта obj_AntiPlayer

В об’єкті obj_Entity в події Create прописаний код (див. рис. 3.25), що визначає спільні для всіх сутностей параметри.

Рядок phy_fixed_rotation = true; запобігає явищу, коли, під час зіткнення з кутом іншого об’єкту, даний об’єкт обертається навколо умовної осі z. Наприклад, ігровий персонаж зачіпає головою кут стінки, продовжуючи

рухатись, перевертається на спину і продовжує ходити саме в цій позиції (рис. 3.26).

```
///щоб не крутився під час торкання стінок
phy_fixed_rotation = true;
hp = 3;

if (hp <= 0){
    instance_destroy();
}
```

Рисунок 3.25 – Код події Create об’єкту obj_Entity

Змінна hp відповідає за лічильник здоров’я сутностей. Якщо цей лічильник падає до нуля або менше, згідно подальшому коду, сутність помирає, а отже об’єкт знищується за допомогою функції рушія instance_destroy().

Рисунок 3.26 – Демонстрація відсутності рядка коду phy_fixed_rotation = true;

В події Step об’єкту obj_Entity прописано лише 1 рядок коду (рис. 3.27).

```
depth = -y; //гравець поверх інших текстур
```

Рисунок 3.27 – Код події Step об’єкту obj_Entity

Це зроблено для того, аби об’єкти сутностей відображалися поверх об’єктів оточення, а не під ними.

Останнім розглянемо об’єкт obj_Damage. В розділі Events цього об’єкту було додано подію, котра відповідає за взаємодію з сутностями (рис. 3.28).

Рисунок 3.28 – Подія для взаємодії об’єкту obj_Damage з сутностями

В цій події прописано код, котрий наносить шкоду ігровим ворогам (рис. 3.29). Перед зміною вказано параметр other, котрий значить, що вказана змінна змінюється для всіх сутностей, котрі взаємодіють з об’єктом obj_Damage.

```
1 other.hp -= 1;
```

Рисунок 3.29 – Код в події об’єкта obj_Damage

Таким чином і було реалізовано всі необхідні функції ігрового застосунку.

3.2 Тестування ігрового застосунку

Коли дії з реалізації ігрового застосунку було проведено, можна приступити до його тестування. Це стає можливим завдяки вбудованому в ігровий рушій функціоналу, що дозволяє швидко провести необхідні тести та оперативно виправити виявлені помилки. Тестування є важливим етапом розробки, який забезпечує стабільну роботу гри та відповідність її

функціональних можливостей до визначених вимог.

Першим проводиться тестування керування ігровим персонажем (рис. 3.30).

Рисунок 3.30 – Перевірка рухів персонажа

Цей етап включає в себе наступні тести:

– перевірка реакції на натискання клавіш, що визначає, чи правильно персонаж реагує на команди гравця;
– перевірка напрямків і швидкості руху персонажа, що допомагає визначити, чи рухається він у правильному напрямку та з правильною швидкістю;
– перевірка правильності запуску та зупинки анімацій і їх відповідності до дій, для яких вони призначені, що гарантує, що анімації відображаються коректно та відповідають діям персонажа.

Наступним етапом є перевірка колізій об’єктів, тобто правильність їх реакцій один на одного. Сюди входять такі перевірки:

– чи не проходить персонаж через стіну, що гарантує правильне визначення меж об’єктів;
– чи не застрягає персонаж в стіні, що забезпечує плавність руху

персонажа;

– чи може він зрушити стіну з місця, що дозволяє перевірити взаємодію з рухомими об’єктами;
– чи не змінюється його положення (чи не крутиться) при зіткненні з іншими об’єктами, що гарантує стабільну позицію персонажа під час колізій;
– чи відповідає колізія розмірам спрайтів, що забезпечує точність візуального відображення колізій;
– що відбувається при зіткненні з NPC, що дозволяє перевірити правильність взаємодії з іншими персонажами;
– що відбувається при зіткненні NPC зі стінами, що перевіряє поведінку інших персонажів при взаємодії з навколишнім середовищем.

Наступним етапом було тестування штучного інтелекту NPC (рис. 3.31).

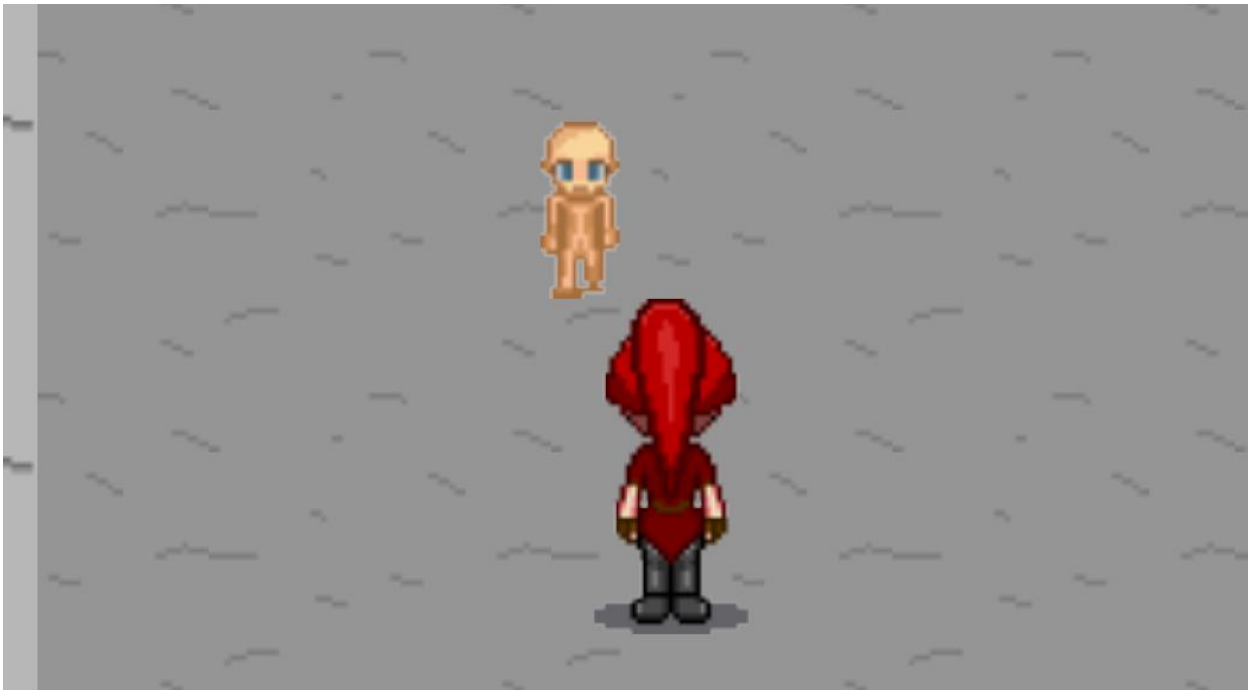

Рисунок 3.31 – Тестування штучного інтелекту

Цей етап включає такі перевірки:

– чи правильно NPC знаходить напрямок свого руху, що визначає коректність алгоритму навігації;

– чи правильна у NPC швидкість руху, що забезпечує реалістичність поведінки;
– чи не застрягає NPC у перешкодах, що гарантує плавність руху та уникнення блокування.

Далі була проведена перевірка бойової системи (рис. 3.32).

Рисунок 3.32 – Перевірка бойової системи

Сюди увійшли наступні перевірки:

– чи викликається атакуючий об’єкт, що перевіряє коректність ініціації бойових дій;
– чи зменшується параметр hp у NPC, що гарантує правильність обчислень шкоди;
– чи сила атаки відповідає необхідній, що перевіряє балансування бойових механік;
– чи діє атака на ігрового персонажа, що забезпечує правильність бойових взаємодій.

Останнім етапом була протестована діалогова система. Сюди входили такі перевірки:

– чи працює кнопка взаємодії, що визначає коректність початку

діалогів;

– чи правильно відображається діалогове вікно, що забезпечує зручність інтерфейсу;
– чи немає помилок в самому діалозі, що гарантує точність та чіткість спілкування.

Після завершення всіх тестувань були виявлені і виправлені знайдені помилки, а також враховано певні нюанси реалізації різних деталей гри. Наприклад, було підкореговано колізії деяких об’єктів та порядок спрайтів в анімаціях ігрового персонажа. Ці кроки дозволили досягти більш високої якості кінцевого продукту та забезпечити його стабільну роботу.

ВИСНОВКИ

У даній кваліфікаційній роботі детально розглянуто основні принципи роботи з ігровим рушієм GameMaker для створення ігрового застосунку, що представляє собою початковий рівень майбутньої гри. У процесі виконання роботи було вирішено низку важливих завдань.

Аналіз існуючого інструментарію, де було ретельно проаналізовано доступні засоби та можливості gamemaker, що дозволило визначити оптимальні інструменти для реалізації задуму.

Аналіз предметної області, де проведено глибоке дослідження предметної області, що охоплює пригодницькі ігри та їх особливості. Це дало змогу краще зрозуміти вимоги до гри, її структуру та основні елементи.

Формулювання вимог до ігрового застосунку, де визначено ключові вимоги до ігрового застосунку, включаючи функціональні та нефункціональні аспекти. Це забезпечило чітке бачення кінцевого продукту.

Проєктування схеми даних, де розроблено детальну схему даних, яка описує взаємодію між різними об’єктами гри, їхні властивості та методи. Це дозволило створити чітку структуру даних для гри.

Створення візуальної та аудіо складових, де виконано комплекс дій, спрямованих на створення візуальних і аудіо елементів гри. Враховано стиль піксельарт, який підкреслює унікальність і естетику майбутньої гри.

Програмна реалізація та тестування ігрового застосунку, де реалізовано програмну частину гри, включаючи основний функціонал та ключові елементи геймплею. Проведено тестування для виявлення та виправлення можливих помилок.

У теоретичній частині роботи було досліджено предметну область, визначено особливості Проєктування ігрових застосунків у жанрі пригодницької гри з піксельарт стилем. Виділено основні об’єкти гри, їх параметри та методи, а також зв’язки між ними.

Під час практичної частини роботи було безпосередньо створено ігровий застосунок, у якому реалізовано основний функціонал та базові елементи гри. Також детально розглянуто особливості роботи з ігровим рушієм GameMaker, що дозволило краще зрозуміти його можливості та обмеження.

Таким чином, у роботі не лише створено прототип ігрового рівня, але й проведено всебічний аналіз та дослідження, які забезпечили високу якість кінцевого продукту. Результати роботи можуть бути використані для подальшого розвитку гри та створення більш складних і цікавих рівнів.

ПЕРЕЛІК ПОСИЛАНЬ

1. What Is A Triple-A Game (AAA)? URL: https://www.gamingscan.com/what-is-a-triple-a-game/ (дата звернення: 15.03.2024).

2. The major differences between ‘indie’ and ‘AAA’ video games. URL: https://www.windowscentral.com/indie-vs-aaa-which-type-game-you (дата звернення: 15.03.2024).

3. Popular camera angles used in video games. URL: https://behind-the-scenes.net/popular-camera-angles-used-in-video-games/#Third-person_view (дата звернення: 20.03.2024).

4. Undertale PC Review. URL: https://rpgamer.com/review/undertale-review/ (дата звернення: 01.04.2024).

5. Undertale Review. URL: https://techraptor.net/gaming/reviews/undertale-review-no-bones-about-it (дата звернення: 01.04.2024).

6. Deltarune Chapter 2 Review (Pc): Right At Cyber Home. URL: https://www.theclick.gg/gaming/reviews/deltarune-chapter-2-review-right-at-cyber-home/ (дата звернення: 03.04.2024).

7. Deltarune is a game about playing, and it’s bloody brilliant. URL: https://www.eurogamer.net/deltarune-is-a-game-about-playing-and-its-bloody-brilliant (дата звернення: 03.04.2024).

8. A visual history of video games from retro pixels to lifelike graphics. URL: https://mobidictum.com/a-visual-history-of-video-games-from-retro-pixels-to-lifelike-graphics/ (дата звернення: 05.04.2024).

9. Як створити персонажа гри: принципи гейм-арту та легендарні герої. URL: https://vokigames.com/ua/yak-stvoryty-personazha-gry-prynczypy-gejm-artu-ta-legendarni-geroyi/ (дата звернення: 05.04.2024).

10. Undertale Review. URL:

https://www.nintendolife.com/reviews/switch-eshop/undertale (дата звернення: 05.04.2024).

11. Важливість звуку в відеоіграх. URL: https://versum.ua/vazhnost-zvuka-v-videoigrah (дата звернення: 07.04.2024).