МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

“До захисту допущена”
Зав. кафедри комп’ютерних наук та прикладної математики

«_____» _____20__р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **Застосування візуальної платформи Miro для командної розробки 2D гри**

Виконав:
студент навчально-наукового Інституту автоматики, кібернетики
та обчислювальної техніки
Група КН-41
Брикса Вадим Володимирович
Керівник: к. е .н., доцент каф. комп’ютерних наук та прикладної математики
Бачишина Лариса Дмитрівна
Рецензент

Рівне – 2023

Національний університет водного господарства та природокористування

Навчально-науковий інститут автоматики, кібернетики та обчислювальної техніки
Кафедра **комп'ютерних наук та прикладної математики**
Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки та інформаційні технології»**

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____
_____
"____" _____20__ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Брикси Вадима Володимировича

1. Тема проекту **Застосування візуальної платформи miro для командної розробки 2D гри**

керівник проекту *к.е.н., доцент Бачишина Лариса Дмитрівна*,

затверджені наказом вищого навчального закладу від " 19 *" квітня 2023* року С-№ 449

2. Термін здачі студентом закінченої роботи *26.05.2023 р.*

3. Вихідні дані до проекту: *інформація про існуючі методи та засоби командної розробки, та засоби розробки 2D ігор*

4. Зміст розрахунково-пояснювальної записки *у першому розділі роботи розглянуто особливості командної розробки, платформи і методи, які використовуються при командній розробці, платформи і мови програмування для створення ігор. У другому розділі розглядаються основні принципи роботи з Unity, C#, Miro. У третьому розділі описано командну розробку 2D гри за допомогою Unity з використання сервісу Miro.*
_____

5. Перелік графічного матеріалу _____

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	*к. е. н, доцент, Бачишина Л. Д.*	*11.10.2022 р*	*11.10.2022 р*
2	*к. е. н, доцент, Бачишина Л. Д.*	*24.02.2023*	*24.02.2023*
3	*к. е. н, доцент, Бачишина Л. Д.*	*24.03.2023*	*24.03.2023*

7. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	*Отримання завдання і розробка плану роботи*	*11.10.2022*	*Виконано*
2	*Дослідження літератури*	*11.10.2022-11.01.2023*	*Виконано*
3	*Аналіз і підбір засобів для виконання завдань*	*11.01.2023-24.02.2023*	*Виконано*
4	*Створення концепції і функціоналу для роботи на платформі Miro*	*24.02.2023-24.03.2023*	*Виконано*
5	*Напрацювання механік і функціоналу для розробки проекту на Unity*	*24.03.2023-24.04.2023*	*Виконано*
6	*Збірка готового проекту із напрацьованих механік на Unity*	*24.04.2023-22.05.2023*	*Виконано*
7	*Написання документації*	*22.05.2023-26.05.2023*	*Виконано*

Студент _____ (*Брикса В.В.*)

Керівник кваліфікаційної роботи _____ (*Бачишина Л.Д.*)

РЕЗЮМЕ

Кваліфікаційна робота на тему « Застосування візуальної платформи Miro для командної розробки 2D гри» на здобуття освітньо-кваліфікаційного рівня «Бакалавр» зі спеціальності 122 «Комп'ютерні науки та інформаційні технології» написана обсягом 62 сторінки і містить 7 рисунків та 19 джерел за переліком посилань.

Метою роботи є оцінка та аналіз ефективності використання візуальної платформи Miro в процесі командної розробки 2D гри.

Методи досліджень. Для досягнення поставленої мети у кваліфікаційній роботі використовуються наступні методи: аналіз та синтез, оптимізація параметрів, опис, спостереження, моделювання.

Результати дослідження: проведено оцінку ефективності використання візуальної платформи Miro в процесі командної розробки 2D гри.

Орієнтовні напрямки розвитку досліджень: обмеження та проблеми, що виникають при використанні Miro, надають можливості для подальших досліджень та вдосконалення.

КЛЮЧОВІ СЛОВА: 2D ГРА, ВІЗУАЛЬНА ПЛАТФОРМА MIRO, КОМАНДНА РОБОТА, UNITY, C#.

SUMMARY

Qualification work on the topic "Application of the Miro visual platform for team development of a 2D game" for the bachelor's degree in the specialty 122 "Computer Science and Information Technology" is written in 62 pages and contains 7 figures and 19 sources in the list of references.

The purpose of the work is to evaluate and analyse the effectiveness of using the Miro visual platform in the process of team development of a 2D game.

Research methods. To achieve this goal, the following methods are used in the qualification work: analysis and synthesis, parameter optimisation, description, observation, modelling.

Results of the study: the effectiveness of using the Miro visual platform in the process of team development of a 2D game was evaluated.

Indicative directions for further research: the limitations and problems that arise when using Miro provide opportunities for further research and improvement.

KEYWORDS: 2D GAME, MIRO VISUAL PLATFORM, TEAMWORK, UNITY, C#.

ЗМІСТ

ВСТУП

Актуальність теми. Застосування візуальних платформ, таких як Miro, для командної розробки 2D-гри є актуальною темою дослідження, особливо в сучасному контексті, коли дедлайни стають все коротшими, а команди розробників стають все більш розосередженими географічно.

Перше, що робить цю тему актуальною, - це зростання сектору розробки ігор [1],[2],[3]. Галузь розвивається швидко, а з ростом цієї галузі збільшується і потреба в ефективних інструментах для командної роботи. Miro - це такий інструмент, який дозволяє командам спілкуватися та працювати разом незалежно від їхнього місцезнаходження. Це особливо важливо в контексті постійного зростання ринку ігор, який став одним з найбільш прибуткових сегментів розважальної індустрії. Варто відзначити і швидке зростання індустрії ігор у близьких географічно до України країнах ЄС, таких зокрема, як Польща [4] та Чехія [5].

Крім того, з'являється все більше і більше віддалених команд розробників. Це стає нормою, особливо після пандемії COVID-19, яка змусила багато компаній перейти на віддалену роботу. Візуальні платформи, такі як Miro [6], дозволяють цим командам співпрацювати ефективно, незважаючи на відстань. Ці платформи дозволяють командам спільно працювати над проектами, спілкуватися та ділитися ідеями в режимі реального часу.

Серед всього іншого, Miro є потужним інструментом для візуалізації, який може допомогти в процесі розробки ігор. Платформа Miro може бути використанf для створення діаграм, що допомагають у візуальному плануванні та розробці ігор. Візуальне представлення може бути корисним для розуміння процесу розробки та сприяти ефективнішій комунікації між членами команди.

Окрім цього, Miro має вбудовані інструменти для організації та керування проектами, що можуть допомогти командам більш ефективно керувати своїми проектами розробки. Платформа надає можливість створювати дошки для різних аспектів проекту, таких як планування,

проектування, тестування тощо. Це може допомогти командам краще організувати свою роботу і забезпечити більш ефективне виконання проекту.

Отже, враховуючи швидкий розвиток галузі розробки ігор, зростання числа віддалених команд та важливість ефективних інструментів для командної роботи та візуалізації, тема "застосування візуальної платформи Miro для командної розробки 2D гри" безсумнівно є актуальною для сучасного дослідження.

Мета дослідження полягає в оцінці та аналізі ефективності використання візуальної платформи Miro в процесі командної розробки 2D гри.

Виходячи з мети дослідження, випливають такі **основні завдання**:

1) визначити особливості командної розробки, платформи і методи, які використовуються при командній розробці, платформи і мови програмування для створення ігор;

2) розглянути основні принципи роботи з Unity, C#, Miro;

3) описати командну розробку 2D гри за допомогою Unity з використання сервісу Miro та оцінити ефективність застосування візуальної платформи Miro в процесі командної розробки 2D гри.

Об’єктом дослідження є платформи та рішення для оптимізації та підвищення ефективності командної розробки 2D ігор.

Предметом дослідження є застосування візуальної платформи Miro для командної розробки 2D гри.

Методи досліджень. Для досягнення поставленої мети у кваліфікаційній роботі використовуються наступні методи: аналіз та синтез, оптимізація параметрів, опис, спостереження, моделювання.

Практичне значення роботи полягає у тому, що дослідження доводить доцільність використання методів організації командної роботи з використанням необхідних для цього платформ і сервісів. Робота має перспективи розвинутися до повноцінного продукту. Аналіз використання платформ, приклад розробки конкретної гри може бути використано як кейс в

проєктно-орієнтованому навчанні в навчальному процесі для підготовки та/або перепідготовки фахівців з компютерних наук.

Структура роботи: кваліфікаційна робота складається з трьох розділів, обсягом 62 сторінок; кількість позицій списку використаних джерел – 19, рисунків – 7. У першому розділі роботи розглянуто особливості командної розробки, платформи і методи, які використовуються при командній розробці, платформи і мови програмування для створення ігор У другому розділі розглядаються основні принципи роботи з Unity, C#, Miro. У третьому розділі описано командну розробку 2D гри за допомогою Unity з використання сервісу Miro.

РОЗДІЛ 1. ОГЛЯД ОСНОВНИХ МЕТОДІВ І ЗАСОБІВ ДЛЯ КОМАНДНОЇ РОЗРОБКИ ІГОР

1.1 Огляд розробки ігор та співпраці в команді

1.1.1 Вступ до розробки 2D ігор

Розробка 2D ігор - це процес створення відеоігор, які працюють у двовимірному середовищі. На відміну від 3D-ігор, 2D-ігри мають висоту і ширину, але не мають глибини. До цього жанру належать класичні аркадні ігри, такі як "Pac-Man" і "Space Invaders", а також новіші ігри, такі як "Stardew Valley" і "Terraria". Ключові підходи дло розробки ігор описані в сучасній літературі, зокрема [7], [8], [9] [10], [11], [12], [13].

Розробка 2D-гри складається з кількох ключових етапів:

Концептуалізація: Це етап, на якому народжується ідея гри. Він включає в себе мозковий штурм і прийняття рішень щодо основних ігрових механік, сюжетної лінії, персонажів, естетичного стилю та інших фундаментальних аспектів гри.

Дизайн: Після того, як концепція створена, фаза дизайну включає в себе створення правил, структури та макету гри. Цей етап часто включає малювання або створення рівнів, персонажів та інших візуальних елементів гри в цифровому форматі.

Програмування: Цей етап передбачає написання коду для реалізації ігрових механік, елементів керування та логіки. Зазвичай це робиться за допомогою ігрового рушія та мови програмування, таких як Unity та C# відповідно.

Тестування: На етапі тестування у гру грають різними способами, щоб виявити та виправити будь-які помилки або збої. Відгуки тестувальників використовуються для вдосконалення ігрового процесу та забезпечення того, щоб гра була приємною та придатною для гри.

Реліз: Після того, як гру ретельно протестовано та доопрацьовано, вона готова до релізу. Це може включати публікацію гри на таких платформах, як Steam або App Store, маркетинг гри та надання підтримки після релізу.

1.1.2 Роль командної співпраці у розробці ігор

Зважаючи на складність розробки ігор, це рідко буває одиночна робота. Командна співпраця часто необхідна для ефективного та результативного створення гри. Типова команда розробників ігор може включати геймдизайнерів, програмістів, художників, звукорежисерів та менеджерів проектів, кожен з яких має свої унікальні навички та погляди.

Співпраця у розробці ігор включає кілька ключових аспектів:

Комунікація: Чітка та ефективна комунікація має вирішальне значення в командній роботі. Це включає передачу ідей, надання та отримання зворотного зв'язку, а також координацію зусиль. Такі інструменти, як Miro, можуть полегшити цю комунікацію за допомогою створення спільного простору, де можна візуально висловлювати та обговорювати ідеї.

Управління завданнями: Коли кілька членів команди працюють над різними аспектами гри, важливо відстежувати, хто, що і коли робить. Інструменти управління завданнями можуть допомогти призначати завдання, відстежувати прогрес і гарантувати, що всі знаходяться на одній сторінці.

Координація: Хороша координація гарантує, що всі різні елементи гри будуть працювати злагоджено. Це включає в себе координацію графіку розробки, інтеграцію різних частин гри (наприклад, мистецтва, звуку та програмування), а також забезпечення відповідності зусиль команди баченню гри.

По суті, ефективна командна співпраця в розробці ігор може підвищити креативність, ефективність і продуктивність, що призведе до більш успішного ігрового проекту.

1.2 Платформа розробки Unity та мова C#

1.2.1 Основи використання Unity для розробки 2D ігор

Unity - це кросплатформенний ігровий рушій, розроблений компанією Unity Technologies. Він надає повний набір інструментів та функцій для створення як 2D, так і 3D ігор. Для розробки 2D ігор Unity надає спеціальний робочий процес з унікальними інструментами та налаштуваннями:

2D Рендерер: 2D Renderer у Unity забезпечує гнучкий та ефективний спосіб створення високоякісної 2D графіки навіть на менш потужному обладнанні. Він підтримує різноманітні ефекти та можливості, такі як рендеринг спрайтів, плитковий рендеринг мап та 2D освітлення для створення візуально привабливих ігор.

2D Physics Engine: Unity включає 2D Physics Engine, який імітує фізичну поведінку об'єктів у грі. Він допомагає створювати реалістичні рухи та взаємодії, включаючи виявлення зіткнень між об'єктами, гравітацію та інші сили.

Редактор спрайтів: Редактор спрайтів у Unity використовується для маніпулювання 2D спрайтами та аркушами спрайтів. Він дозволяє розробникам визначати конкретні області графічного файлу, які представляють окремі спрайти, а також допомагає у створенні анімації, зміні точки повороту або налаштуванні фізичної форми спрайтів.

Tilemap: Функція Tilemap дозволяє розробникам створювати 2D-рівні за допомогою плиток, підвищуючи ефективність і швидкість дизайну рівнів.

1.2.2 Роль C# у розробці Unity

C# - це популярна, універсальна та об'єктно-орієнтована мова програмування, яка в основному використовується в Unity для написання сценаріїв поведінки гри. Вона дозволяє розробникам контролювати кожен аспект гри, від поведінки персонажів до штучного інтелекту, користувацького інтерфейсу тощо. Ключові аспекти C# в Unity включають

Ігрова логіка: C# використовується для реалізації правил та механіки гри. З її допомогою можна контролювати взаємодію ігрових об'єктів між

собою, визначати наслідки цієї взаємодії та загалом керувати тим, як гра реагує на дії гравця.

Анімація та візуальні ефекти: за допомогою C# розробники можуть керувати анімацією та візуальними ефектами, такими як рух ігрових об'єктів, зміна кольорів, запуск ефектів частинок тощо.

Обробка вводу: Скрипти C# в Unity можна використовувати для обробки вводу гравця з клавіатури, миші, сенсорного екрану або ігрового контролера. Це дозволяє грі реагувати на дії гравця, такі як переміщення персонажа, стрільба зі зброї або призупинення гри.

Програмування штучного інтелекту: C# також використовується в Unity для програмування штучного інтелекту для неігрових персонажів. Це може варіюватися від простої поведінки (наприклад, слідування заданим шляхом) до складного прийняття рішень.

По суті, C# в Unity слугує основою ігрової функціональності, дозволяючи створювати складні та інтерактивні ігри.

1.3 Вступ до візуальної платформи Miro

1.3.1 Особливості та функціональні можливості Miro

Miro - це онлайн-платформа для візуальної співпраці, розроблена для полегшення командної роботи та мозкових штурмів. Вона пропонує інтерактивну цифрову дошку, яку можна використовувати в режимі реального часу або асинхронно. Ось деякі ключові особливості та функціональні можливості Miro:

Співпраця в режимі реального часу: Miro дозволяє декільком користувачам одночасно працювати на одній дошці з будь-якої точки світу. Зміни, внесені одним користувачем, миттєво видно іншим, що робить співпрацю безперешкодною та ефективною.

Нескінченне полотно: Miro надає широке, масштабоване полотно, яке може вмістити всі типи контенту - від стікерів і ескізів до зображень, відео та інших мультимедійних матеріалів. Це робить його ідеальним для масштабних мозкових штурмів і сесій планування.

Попередньо створені шаблони: Miro постачається з безліччю готових шаблонів для різних цілей, таких як ментальні карти, блок-схеми та гнучкі дошки. Їх можна використовувати як відправну точку для різних типів спільної роботи.

Інтерактивні дошки: Цифрові дошки Miro є інтерактивними, що дозволяє користувачам переміщувати, змінювати розміри та редагувати об'єкти. Вони також можуть додавати коментарі, залишати відгуки та брати участь в обговореннях прямо на дошці.

Інтеграція: Miro інтегрується з багатьма іншими інструментами, такими як Google Drive, Slack та Jira, що полегшує інтеграцію Miro в існуючі робочі процеси.

1.3.2 Використання Miro у спільних проектах

В контексті спільних проектів, таких як розробка ігор, Miro може виконувати безліч функцій:

Мозковий штурм та генерування ідей: Команди можуть використовувати Miro для мозкового штурму ігрових концепцій, персонажів, механік тощо. Вони можуть створювати ментальні карти, блок-схеми або ескізи, щоб візуально представляти і розвивати свої ідеї.

Планування та управління завданнями: Miro можна використовувати для планування процесу розробки гри та управління завданнями. Команди можуть створювати часові шкали, діаграми Ганта або дошки Kanban, щоб відстежувати прогрес і переконатися, що всі знають про свої обов'язки.

Wireframing та макети: для геймдизайнерів Miro можна використовувати для створення каркасів і макетів ігрових інтерфейсів, рівнів та інших візуальних елементів. Це допомагає візуалізувати кінцевий продукт і полегшує зворотній зв'язок та внесення змін.

Документація та рефлексія: Miro-дошки можуть слугувати центральним сховищем для всієї інформації, пов'язаної з проектом, включно з протоколами зустрічей, проектною документацією та ретроспективами. Це полегшує членам команди залишатися в курсі подій і вчитися на кожному проекті.

Загалом, Miro є універсальним інструментом, який може покращити співпрацю, впорядкувати робочі процеси та підвищити загальну ефективність проекту.

1.4 Огляд альтернативних інструментів для розробки ігор та командної співпраці

1.4.1 Платформи та мови розробки ігор

Окрім Unity та C#, існують інші платформи для розробки ігор та мови програмування, які можна використовувати для досягнення подібних цілей. [14], [15], [16], [17],[18],[19].

Unreal Engine

Unreal Engine, розроблений Epic Games, - це потужний рушій для розробки ігор, найбільш відомий своєю приголомшливою високоякісною графікою та системою візуальних сценаріїв Blueprints. Він широко використовується в індустрії для створення ігор класу triple-A, віртуальної реальності, архітектурних візуалізацій тощо. Ось деякі з її ключових особливостей:

Blueprints Visual Scripting: Ця система дозволяє розробникам створювати ігрову логіку без написання коду. Це інтерфейс на основі вузлів, де функції представлені у вигляді графічних "блоків". Це особливо корисно для дизайнерів і художників, які можуть не мати глибокого розуміння програмування.

Високоякісна графіка: Unreal Engine відомий своєю здатністю створювати високореалістичну графіку та візуальні ефекти. Він підтримує динамічне освітлення, фізичний рендеринг та різноманітні ефекти постобробки, що робить його ідеальним для створення візуально приголомшливих ігор.

Просунута система анімації: Unreal пропонує складну систему анімації. Вона включає такі функції, як машина станів для анімації персонажів, накладання анімації, інверсна кінематика і навіть повнофункціональний кінематографічний редактор.

Мультиплатформенна підтримка: Як і Unity, Unreal Engine підтримує широкий спектр платформ, включаючи Windows, macOS, iOS, Android, VR-платформи, консолі тощо.

Однак Unreal Engine може бути більш ресурсоємним, ніж Unity, а його високий рівень складності може ускладнити його вивчення, особливо для початківців.

Godot

Godot - це безкоштовний ігровий рушій з відкритим вихідним кодом, який підтримує розробку як 2D, так і 3D ігор. Він відомий своїм легким, гнучким дизайном та унікальною системою сцен. Ось деякі основні моменти:

Система сцен: у Godot все є сценою. Сценою може бути ціла гра, рівень, персонаж або навіть одна кнопка. Сцени можуть бути екземплярами і вкладені в інші сцени, що сприяє багаторазовому використанню та чистому, організованому робочому процесу.

GDScript: Це власна мова сценаріїв Godot, розроблена, щоб бути легкою у вивченні та використанні. За синтаксисом вона дуже схожа на Python, що робить її доступною для тих, хто знайомий з Python.

Візуальне написання сценаріїв та підтримка C++: на додаток до GDScript, Godot також пропонує візуальне написання сценаріїв та підтримку C++. Це дає розробникам більше можливостей та гнучкості у написанні ігрової логіки.

Інтегроване середовище розробки: Godot постачається з вбудованим середовищем розробки, яке включає редактор коду, налагоджувач, профілювальник та інші інструменти.

Хоча Godot є потужним і гнучким, він не настільки широко розповсюджений, як Unity або Unreal, що означає, що існує менше ресурсів і навчальних посібників, а спільнота розробників є меншою.

GameMaker Studio 2

GameMaker Studio 2, розроблений YoYo Games, - це рушій для розробки ігор, орієнтований на 2D-ігри. Він відомий своєю доступністю та простотою

використання, що робить його популярним вибором для початківців та аматорів. Ось деякі з його особливостей:

GML (GameMaker Language): GML - це вбудована мова сценаріїв GameMaker. Це високорівнева, інтерпретована мова, яку легко вивчити, але вона достатньо потужна, щоб впоратися зі складною ігровою логікою.

Система перетягування: Ця функція дозволяє створювати ігри користувачам з невеликими знаннями програмування або взагалі без них. Користувачі можуть вибирати дії та події зі списку та перетягувати їх на ігрові об'єкти, щоб визначити їхню поведінку.

Підтримка тайлсетів: GameMaker забезпечує надійну підтримку наборів тайлів, що полегшує створення складних 2D-рівнів.

Маркетплейс: Маркетплейс GameMaker пропонує безліч ресурсів, включаючи скрипти, спрайти та розширення, які можна використовувати для покращення ваших ігор або спрощення процесу розробки.

Хоча GameMaker Studio 2 є чудовим інструментом, особливо для початківців та розробки 2D ігор, він менше підходить для розробки 3D ігор. Крім того, в той час як GML – це незважаючи на те, що її легко вивчити, вона не так широко використовується, як такі мови, як C#, а це означає, що набуті навички можуть не так легко переноситися на інші платформи чи проєкти.

Вибір правильного інструменту для розробки ігор залежить від кількох факторів, зокрема від конкретних вимог проекту, знайомства команди з інструментом, бажаної платформи для фінальної гри та ресурсів, доступних для навчання і вирішення проблем під час розробки. Вкрай важливо врахувати всі ці фактори перед тим, як вибрати платформу та мову розробки гри.

1.4.2 Поглиблений огляд альтернативних платформ для командної роботи та візуалізації

Trello

Trello - це веб-додаток для створення списків у стилі Kanban, який допомагає командам залишатися організованими. Це простий, але гнучкий

інструмент, що дозволяє командам керувати завданнями та візуально відстежувати прогрес проекту.

Дошки, списки та картки: Основними компонентами Trello є дошки, списки та картки. Дошка - це проєкт або робочий простір. Усередині кожної дошки ви можете створювати кілька списків, які зазвичай представляють різні етапи робочого процесу. Кожен список містить картки, які є завданнями або елементами.

Можливості для спільної роботи: Члени команди можуть коментувати картки, прикріплювати файли, створювати контрольні списки і навіть призначати терміни виконання. Trello підтримує оновлення в реальному часі, тому, коли вносяться зміни, кожен член команди може негайно їх побачити.

Інтеграції: Trello підтримує інтеграцію з багатьма іншими інструментами, включаючи Google Drive, Slack і GitHub, що робить його універсальною платформою для управління різними аспектами проекту.

Однак, хоча Trello чудово підходить для управління завданнями і базового відстеження проектів, йому не вистачає деяких розширених функцій управління проектами, які можна знайти в інших інструментах.

Microsoft Whiteboard

Microsoft Whiteboard - це нескінченне цифрове полотно вільної форми, яке ідеально підходить для мозкового штурму, планування та спільної роботи.

Перетворення чорнила на фігуру та перетворення чорнила на стіл: Ці функції дозволяють автоматично перетворювати малюнки на фігури або таблиці, що полегшує створення впорядкованих і читабельних нотаток.

Наліпки та текст: Ви можете додавати на дошку друкований текст або стікери, які можуть бути корисними для запису ідей, завдань або нагадувань.

Інтеграція з екосистемою Microsoft: Якщо ваша команда вже використовує Microsoft 365 або Teams, використання Microsoft Whiteboard може бути корисним завдяки його безшовній інтеграції з цими платформами.

Однак варто зазначити, що Microsoft Whiteboard менш функціональний порівняно з Miro, і рівень співпраці може бути не таким високим.

Asana

Asana - це веб- та мобільний додаток, розроблений, щоб допомогти командам керувати своєю роботою та відстежувати її прогрес. Це більше інструмент для управління проектами, ніж платформа для візуальної співпраці, але він все ж пропонує деякі функції візуального планування.

Управління завданнями: Ви можете створювати завдання, призначати їх членам команди, встановлювати терміни виконання і додавати описи або вкладення. Завдання можна переглядати у вигляді списку, дошки або календаря.

Хронологія проєкту: Функція часової шкали Asana дозволяє створювати діаграму Ганта для вашого проекту, забезпечуючи візуальне представлення вашого розкладу.

Звітність та відстеження: Asana надає розширені функції звітності, що полегшує відстеження прогресу проекту та продуктивності команди.

Інтеграції: Asana інтегрується з широким спектром додатків, включаючи Slack, Google Drive та GitHub.

Хоча Asana пропонує потужний функціонал для управління проектами, вона не має відкритого полотна вільної форми, як Miro, що може обмежувати її використання для певних типів візуальної співпраці.

Figma

Figma - це хмарний інструмент для дизайну, відомий своїми можливостями для спільної роботи. Хоча його переважно використовують для UI/UX дизайну, він також може слугувати платформою для візуальної співпраці.

Співпраця в реальному часі: Кілька користувачів можуть одночасно працювати над одним файлом дизайну, що полегшує спільну роботу в режимі реального часу.

Векторне редагування: Figma надає надійні інструменти векторного редагування, які можна використовувати для створення різноманітної графіки.

Створення прототипів: Функції прототипування у Figma дозволяють створювати інтерактивні прототипи ваших дизайнів, які можуть бути корисними для презентації ідей або отримання зворотного зв'язку.

Компоненти та стилі: Ці функції дозволяють створювати багаторазові елементи та стилі, сприяючи узгодженості та ефективності вашого дизайну.

Однак зосередженість Figma на дизайні може зробити його менш придатним для інших видів візуальної співпраці, таких як мозковий штурм або мапування. Він також має крутішу криву навчання порівняно з більш простими інструментами, такими як Miro.

Вибір між цими інструментами залежить від конкретних потреб вашої команди та проекту. Деякі команди можуть віддати перевагу простоті та гнучкості полотна вільної форми, такого як Miro або Microsoft Whiteboard, тоді як інші можуть скористатися структурою та розширеними функціями управління проектами такого інструменту, як Asana. Для команд, зосереджених на дизайні, найкращим вибором може стати Figma. Як і у випадку з інструментами для розробки ігор, важливо враховувати конкретні вимоги вашого проекту, вподобання та навички вашої команди, а також інтеграцію з іншими інструментами, які ви використовуєте, перш ніж обирати інструмент для спільної роботи та візуального планування.

1.4.3 Огляд методів командної розробки

Agile - це популярна стратегія управління проектами та розробки продуктів, яка особливо добре підходить для середовищ, де вимоги можуть швидко змінюватися, як-от розробка програмного забезпечення. Вона наголошує на гнучкості, співпраці, зворотному зв'язку з клієнтами та ітеративному прогресі, що дозволяє командам ефективно реагувати на зміни та створювати високоякісні продукти.

Ось ключові елементи Agile:

1. Ітеративна розробка:

Agile розбиває проекти на невеликі, керовані одиниці, які називаються ітераціями або спринтом. Це короткі проміжки часу (зазвичай 1-4 тижні),

протягом яких команда працює над виконанням певного набору завдань і створенням робочого продукту.

2. Співпраця та комунікація:

Agile заохочує активну співпрацю та комунікацію між членами команди та зацікавленими сторонами. Регулярні зустрічі, такі як щоденні мітинги, планування спринту та ретроспективи спринту, використовуються для синхронізації, планування роботи, обговорення викликів та постійного вдосконалення процесу.

3. Залучення клієнтів:

В Agile замовник (або його представник) бере активну участь у проекті. Вони допомагають визначати вимоги та встановлювати пріоритети, переглядати прогрес, надавати зворотній зв'язок і навіть можуть змінювати напрямок проекту за необхідності, виходячи зі змін на ринку або нових ідей.

4. Гнучкість до змін:

На відміну від традиційних підходів до управління проектами, Agile вітає зміни у вимогах, навіть на пізніх стадіях розробки. Така гнучкість дозволяє продукту розвиватися і гарантує, що кінцевий продукт з більшою ймовірністю відповідатиме потребам замовника.

5. Постійне вдосконалення:

Гнучкі команди прагнуть до постійного вдосконалення. Після кожного спринту команда розмірковує над тим, що пройшло добре, а що можна покращити. Ці ідеї потім використовуються для того, щоб зробити наступний спринт більш ефективним та результативним.

6. Орієнтованість на цінність:

Agile визначає пріоритети роботи на основі цінності для клієнта. Завдання визначають пріоритети в беклозі, і найцінніші завдання вирішуються в першу чергу. Це гарантує, що проект завжди приносить максимальну користь.

7. Розширення можливостей команд:

Agile вірить у розширення можливостей команд у прийнятті рішень. Команда має автономію вирішувати, як найкраще виконати свою роботу, що сприяє формуванню почуття відповідальності та підзвітності.

Коротко кажучи, Agile - це гнучкий, ціннісно-орієнтований підхід, який ставить людей і взаємодію в центр проекту. Він спрямований на зменшення відходів, підвищення прозорості та створення продуктів, які дійсно відповідають потребам клієнтів. Хоча Agile виник в індустрії програмного забезпечення, його принципи можна застосовувати в різних сферах, де роботу можна розбити на ітеративні завдання.

Scrum

Scrum - це специфічна реалізація Agile, яка наголошує на співпраці, функціонуванні програмного забезпечення, самоуправлінні командою та гнучкості адаптації до нових бізнес-реалій. Ось розбивка ключових компонентів і процесів, що беруть участь у Scrum:

1. Ролі:

Власник продукту: ця роль відповідає за представлення інтересів зацікавлених сторін і замовника, управління бэклогом продукту і забезпечення того, щоб команда працювала над найбільш цінними функціями.

Scrum-майстер: Scrum-майстер - це слуга-лідер, який допомагає команді зрозуміти і впровадити практики і принципи Scrum, усуває перешкоди, сприяє проведенню зустрічей і допомагає команді зосередитися на досягненні цілей спринту.

Команда розробників: Команда розробників - це самоорганізована група професіоналів, яка виконує роботу по створенню потенційно готового до випуску продукту в кінці кожного спринту.

2. Події:

Планування спринту: Це зустріч, яка відбувається на початку кожного спринту, де команда визначає мету спринту і над якими елементами бэклогу продукту вони будуть працювати протягом спринту.

Щоденний Scrum (або стендап): Це коротка (15 хвилин або менше) зустріч, яка відбувається щодня, де команда синхронізує свою роботу і планує на наступні 24 години.

Огляд спринту: Наприкінці кожного спринту команда проводить Sprint Review, щоб перевірити прогрес і за потреби адаптувати відставання.

Ретроспектива спринту: Після огляду спринту і перед наступним плануванням спринту команда проводить ретроспективу спринту, де обговорює, що пройшло добре, а що ні, і як можна покращити роботу в наступному спринті.

3. Артефакти:

Беклог продукту: Це список всіх можливостей, функцій, вимог, удосконалень і виправлень, які становлять зміни, що мають бути внесені в продукт у майбутніх релізах.

Беклог спринту: Це список завдань, визначених командою Scrum, які мають бути виконані під час Scrum-спринту.

Інкремент: Інкремент - це сума всіх елементів Беклогу продукту, завершених під час спринту, і значення інкрементів всіх попередніх спринтів.

Scrum забезпечує структуру, яка допомагає командам працювати разом. Заохочуючи команди вчитися на власному досвіді, самоорганізовуватися під час роботи над проблемою та аналізувати свої перемоги і поразки, щоб постійно вдосконалюватися, Scrum підвищує продуктивність і створює високоякісні продукти.

Kanban

Kanban - це візуальна система управління проектами, що використовується для впровадження Agile, яка візуально відображає роботу на різних етапах процесу за допомогою дошки Kanban. Мета Kanban - виявити потенційні вузькі місця у вашому процесі та виправити їх, щоб робота могла проходити через нього економічно ефективно, з оптимальною швидкістю або пропускною здатністю. Ось ключові аспекти методології Kanban:

1. Візуальні сигнали:

Назва "Kanban" з японської перекладається як "візуальний сигнал" або "картка". Відповідно до своєї назви, Kanban використовує візуальні сигнали - у вигляді дошки і карток - щоб вказати, що виробляти, коли виробляти і скільки виробляти.

2. Дошка Kanban:

Базова дошка Kanban має три колонки: "Зробити", "У процесі" та "Готово". Однак, залежно від потреб команди, дошку можна розширити додатковими стовпчиками, щоб відобразити робочий процес команди. Кожна колонка відображає етап робочого процесу.

3. Картки Kanban:

Кожне завдання або робочий елемент представлений на дошці у вигляді картки. Картка починається в колонці "Потрібно зробити" і рухається по дошці до колонки "Виконано". Картки часто містять таку інформацію, як відповідальний за завдання, короткий опис завдання і як довго завдання знаходиться в поточному стовпчику.

4. Ліміти незавершених завдань (Work In Progress Limits, WIP):

Kanban заохочує обмежувати кількість незавершеної роботи (WIP), щоб уникнути перевантаження команди і виявити вузькі місця в робочому процесі. Наприклад, ви можете обмежити колонку "У процесі" п'ятьма завданнями. Це означає, що після того, як п'ять завдань опиняться в колонці "У процесі", нові завдання не можуть бути додані, поки завдання не буде завершено і переміщено в колонку "Виконано".

5. Безперервний потік:

Kanban підкреслює безперервний потік роботи від початку до кінця. Команда витягує роботу зі стовпчика "Справи", коли у них є можливість, а не коли їм нав'язують роботу з дедлайнами. Мета полягає в тому, щоб робота проходила через процес плавно, без перерв і затримок.

6. Постійне вдосконалення:

Як і всі гнучкі методології, Kanban заохочує регулярний перегляд і адаптацію процесу. Команди повинні відстежувати свій робочий процес, виявляти вузькі місця або області затримок і вносити необхідні поліпшення.

Візуалізуючи робочий процес, обмежуючи незавершене виробництво та активно керуючи робочим процесом, команди, які використовують Kanban, можуть досягти високого рівня ефективності та продуктивності. Його можна використовувати практично в будь-якій галузі чи департаменті, а не лише в розробці програмного забезпечення.

Метод водоспаду(Каскадна модель)

Модель водоспаду, також відома як лінійно-послідовна модель, є традиційним підходом до управління проектами та розробки програмного забезпечення, коли проект поділяється на послідовні фази, кожна з яких залежить від результатів попередньої. Ось ключові етапи моделі водоспаду:

1. Аналіз вимог:

На цьому початковому етапі детально фіксуються всі можливі вимоги до системи, що розробляється. Зазвичай це досягається за допомогою зустрічей із зацікавленими сторонами, інтерв'ю з користувачами, опитувань або інших форм збору вимог. Результатом цього етапу є документ з вимогами, який слугує контрактом для решти проекту.

2. Проектування:

На етапі проектування на основі специфікацій вимог готується проектна документація системи та програмного забезпечення. Ці проектні документи використовуються як креслення для фактичної розробки системи. Етап проектування часто поділяють на дві частини: високорівневе (або архітектурне) проектування та низькорівневе (або детальне) проектування.

3. Реалізація (або кодування):

На етапі реалізації проектна документація передається команді розробників для перетворення проекту в код. Система будується і кодується, дотримуючись інструкцій з кодування, з використанням обраних мов програмування.

4. Тестування:

Після того, як код розроблений, він тестується на відповідність вимогам, щоб переконатися, що продукт дійсно вирішує потреби, визначені на етапі визначення вимог. Тестування проводиться для того, щоб переконатися, що всі функціональні можливості працюють так, як очікувалося, а також знайти і виправити будь-які помилки або проблеми.

5. Розгортання:

Після того, як продукт протестований і затверджений, він готовий до розгортання. Це означає, що продукт випускається у виробниче середовище або доставляється замовнику.

6. Обслуговування:

Після того, як продукт був доставлений, починається фаза обслуговування. Це передбачає внесення змін до системи або окремого компонента з метою зміни атрибутів або покращення продуктивності. Ці модифікації виникають або через запити на зміни, ініційовані замовником, або через дефекти, виявлені під час використання системи в реальних умовах.

Модель водоспаду часто протиставляють гнучким методологіям, таким як Scrum або Kanban, які є ітеративними і забезпечують більшу гнучкість. Хоча модель водоспаду може бути ефективною для проектів, де вимоги добре зрозумілі і навряд чи зміняться, вона може бути проблематичною для тривалих або складних проектів, де вимоги можуть змінюватися.

DevOps

DevOps - це набір практик, що поєднує розробку програмного забезпечення (Dev) та IT-операції (Ops), з метою скорочення життєвого циклу розробки системи та забезпечення безперервної доставки з високою якістю програмного забезпечення. DevOps доповнює Agile-розробку програмного забезпечення; деякі аспекти DevOps прийшли з Agile-методології.

Ось основні компоненти DevOps:

1. Культурні практики та принципи:

В основі DevOps лежить культура співпраці та спільної відповідальності між усіма зацікавленими сторонами, які беруть участь у розробці та розгортанні програмного забезпечення. Вона спрямована на руйнування "силосів" і заохочення розробників, операторів, тестувальників та інших до спільної роботи для досягнення спільних цілей.

2. Автоматизація:

Автоматизація є ключовим елементом DevOps. Вона може застосовуватися до забезпечення інфраструктури, розгортання коду, моніторингу додатків тощо. Ідея полягає в тому, щоб мінімізувати ручну роботу, яка може бути повільною, схильною до помилок і непослідовною.

3. Безперервна інтеграція та безперервне розгортання (CI/CD):

CI/CD є критично важливим компонентом DevOps. Безперервна інтеграція передбачає регулярне об'єднання змін коду в центральному репозиторії, де виконуються автоматизовані збірки і тести. Безперервне розгортання робить ще один крок вперед, автоматично розгортаючи додаток у виробництво.

4. Інфраструктура як код (IaC):

Інфраструктура як код - це практика управління та забезпечення обчислювальної інфраструктури за допомогою машинозчитуваних файлів визначень, а не конфігурації фізичного обладнання або інтерактивних інструментів конфігурації.

5. Моніторинг та ведення журналів:

Моніторинг систем і продуктивності додатків має вирішальне значення в контексті DevOps для забезпечення ефективної роботи додатків і швидкого виявлення та усунення будь-яких проблем, які можуть виникнути.

6. Комунікація та співпраця:

Ефективна комунікація та співпраця є ключем до успіху DevOps. Інструменти для онлайн чату, відстеження проектів, спільного використання коду та інших спільних завдань часто використовуються в середовищах DevOps.

7. Мікросервіси:

Хоча це не є вимогою для DevOps, архітектурний стиль мікросервісів часто асоціюється з ним. Мікросервіси передбачають розробку програми як набору невеликих, незалежно розгорнутих сервісів, що може полегшити масштабування і модифікацію додатків.

Завдяки культурі співпраці та впровадженню таких практик, як автоматизація, безперервна інтеграція та безперервна доставка, DevOps допомагає організаціям частіше надавати програмне забезпечення, зберігаючи при цьому стабільність сервісів та отримуючи швидкість, необхідну для більшої кількості інновацій.

Отже, різні описані методи командної роботи, включаючи Agile, Scrum, Kanban і так далі, пропонують унікальні підходи до управління проектами та досягнення успішних результатів. Ось кілька детальних висновків про ці методи командної роботи. Такі методології, як Agile, Scrum та Kanban, сприяють гнучкості, співпраці та ітеративному прогресу. Вони добре підходять для проектів з мінливими вимогами та необхідністю постійного зворотного зв'язку з клієнтами.

Agile дає командам можливість швидко адаптуватися до змін і створювати високоякісні продукти.

Scrum забезпечує основу для ітеративної та інкрементальної розробки продукту. Він наголошує на тісній співпраці, самоорганізації та регулярному зворотному зв'язку. Scrum дозволяє командам створювати цінність за короткі ітерації та коригувати свій підхід на основі зворотного зв'язку з клієнтами.

Kanban фокусується на візуалізації робочого процесу та обмеженні незавершеного виробництва для оптимізації ефективності. Він допомагає командам виявити вузькі місця, зменшити відходи та досягти безперебійного потоку роботи. Kanban особливо корисний для управління та відстеження роботи на різних етапах процесу.

Водоспад: Методологія водоспаду дотримується послідовного підходу з чіткими фазами, що робить її придатною для проектів з чітко визначеними

вимогами та стабільними цілями. Вона підкреслює структурований і спланований підхід, але може бути менш пристосованою до змін, які можуть виникнути під час розробки.

DevOps: DevOps долає розрив між розробкою та експлуатацією, сприяючи співпраці, автоматизації та безперервному наданню послуг. Це дозволяє пришвидшити цикли зворотного зв'язку, підвищити швидкість розгортання та покращити інтеграцію між командами. DevOps сприяє розвитку культури спільної відповідальності та безперервного вдосконалення.

Кожен метод командної роботи має свої сильні сторони та міркування, і вибір методу залежить від таких факторів, як вимоги проекту, динаміка команди та організаційний контекст. Організації також можуть поєднувати елементи різних методологій, щоб створити гібридний підхід, пристосований до їхніх конкретних потреб.

При виборі методу командної роботи важливо оцінити характеристики проєкту, динаміку команди та вимоги замовника. Гнучкість, співпраця та постійне вдосконалення повинні бути прийняті незалежно від обраної методології. Застосовуючи ефективні методи командної роботи, організації можуть підвищити продуктивність, створювати високоякісні продукти та адаптуватися до мінливих вимог галузі.

РОЗДІЛ 2. АНАЛІЗ І МОДЕЛЮВАННЯ ПРИНЦИПІВ КОМАНДНОЇ РОБОТИ ІЗ ЗАСТОСУВАННЯ ПЛАТФОРМИ MIRO

2.1 Існуючі дослідження та приклади використання

2.1.1. Розуміння вимог до гри

Цей підрозділ присвячений ретельному аналізу вимог до гри. Цей процес передбачає детальне розуміння того, чого має досягти гра і якими функціями вона повинна володіти, щоб досягти цих цілей. Ось деякі з аспектів, які будуть розглянуті:

Жанр гри та основні механіки: Першим кроком є чітке визначення жанру гри (наприклад, платформер, шутер, головоломка тощо) та її основної механіки. Основні механіки - це основні дії, які гравець може виконувати у грі (наприклад, стрибати, стріляти, розв'язувати головоломки). Вони є фундаментом, на якому будується решта гри.

Цільова аудиторія: Розуміння цільової аудиторії має вирішальне значення, оскільки воно впливає на багато аспектів гри, від її візуального стилю та рівня складності до платформ, на яких вона повинна бути доступна. Вподобання та звички цільової аудиторії будуть досліджені та враховані.

Візуальний та звуковий дизайн: Це передбачає визначення художнього стилю гри (наприклад, піксельний, мальований, реалістичний тощо), а також типу музики та звукових ефектів, які вона буде мати. Візуальне та звукове оформлення має відповідати темі гри та бути привабливим для цільової аудиторії.

Платформа(и): Платформи, на яких буде випущена гра (наприклад, ПК, мобільні пристрої, консолі), визначаються на основі цільової аудиторії та типу створюваної гри. Кожна платформа має свої вимоги та обмеження, які необхідно враховувати.

Стратегія монетизації: Якщо гра призначена для отримання прибутку, буде визначена відповідна стратегія монетизації. Це можуть бути внутрішні покупки, реклама, преміум-версії тощо.

Глибоке розуміння цих вимог допоможе керувати процесом розробки та впорядкувати його, зменшуючи ризик дорогих змін або поворотів у майбутньому.

2.1.2. Визначення необхідних інструментів та ресурсів

Після чіткого розуміння вимог до гри важливо визначити інструменти та ресурси, які знадобляться для втілення гри в життя. У цьому підрозділі буде описано процес вибору цих елементів:

Платформа розробки: на основі вимог гри буде обрано відповідну платформу для розробки. У цьому випадку було обрано Unity через її підтримку розробки 2D ігор, широку сумісність з різними платформами та знайомство команди з мовою C#.

Художні та звукові активи: Буде визначено тип художніх ресурсів (наприклад, спрайтів, фонів, елементів інтерфейсу) та звукових ресурсів (наприклад, музичних треків, звукових ефектів), необхідних для гри. Також будуть вивчені ресурси для отримання цих ресурсів, такі як цифрові ринки, замовлення художників або створення їх власними силами.

Бібліотеки та плагіни: Залежно від вимог гри, можуть знадобитися додаткові бібліотеки або плагіни для забезпечення певних функцій. Це можуть бути фізичні рушії, бібліотеки штучного інтелекту або плагіни для інтеграції зі специфічними функціями платформи.

Інструменти для співпраці: Будуть визначені інструменти для управління проектами та командної співпраці, такі як Miro для візуальної співпраці, системи контролю версій, такі як Git для управління та відстеження змін коду, та комунікаційні платформи, такі як Slack для командних обговорень.

Інструменти тестування: Будуть обрані інструменти для тестування гри. Це можуть бути фреймворки для автоматизованого тестування, інструменти для налагодження та платформи для розповсюдження бета-версій гри для тестування користувачами.

Цей процес гарантує, що команда від самого початку забезпечена потрібними інструментами та ресурсами, що може значно підвищити ефективність процесу розробки та якість кінцевого продукту.

2.2 Застосування Miro як візуальної платформи для розробки ігор

2.2.1. Розбивка процесу розробки гри

У цьому розділі процес розробки гри розбито на кілька етапів. Кожен з цих етапів є критично важливим для успішного завершення гри.

Концептуалізація: Це початкова фаза, на якій генеруються та наповнюються конкретним змістом ідеї гри. Сюди входить визначення жанру гри, основних механік, сюжету, персонажів, візуального стилю тощо. Результатом цього етапу часто є документ ігрового дизайну (GDD), який слугує орієнтиром протягом усього процесу розробки.

Препродакшн: на цьому етапі детально розробляються основні компоненти гри. Сюди входить створення детальної проектної документації, концепт-арту, розкадровок і плану проекту, який окреслює завдання, графік і ресурси, необхідні для реалізації проекту.

Виробництво: на цьому етапі відбувається власне розробка гри. Команда починає створювати ресурси, писати код, впроваджувати механіки та будувати рівні на основі проектної документації. Проводяться регулярні зустрічі для обговорення прогресу, вирішення проблем і внесення необхідних коректив у план.

Тестування та ітерації: Протягом усього етапу виробництва проводиться регулярне тестування для виявлення та виправлення помилок, оцінки балансу та складності гри, а також збору зворотного зв'язку. На основі результатів цих тестів гра ітеративно вдосконалюється та шліфується.

Запуск і постпродакшн: Коли гра вважається готовою, її запускають на обраних платформах. Після запуску команда може продовжувати працювати над грою, виправляючи помилки, додаючи новий контент і вносячи покращення на основі відгуків гравців.

Такий розподіл процесу розробки гри забезпечує структурований підхід до управління проектом, гарантуючи, що кожному етапу буде приділено достатньо уваги та ресурсів. Це також сприяє більш точному плануванню і складанню розкладу, зменшуючи ризик затримок і недоглядів.

2.2.2. Застосування Miro у процесі розробки ігор

У цьому підрозділі ми розглянемо, як Miro, онлайн-платформа для спільної роботи з дошками, використовувалася на різних етапах процесу розробки гри.

Концептуалізація: на етапі концептуалізації Miro використовували для створення ментальних карт та блок-схем, щоб візуалізувати та дослідити ідеї щодо сюжету, механіки та дизайну гри. Цей спільний мозковий штурм допоміг команді краще зрозуміти концепцію гри та переконатися, що всі знаходяться на одній сторінці.

Препродакшн: на етапі препродакшну Miro використовувався для створення розкадровок, скетчів персонажів та макетів рівнів. Це дозволило команді візуально представити та обговорити свої ідеї, що допомогло вдосконалити дизайн гри перед запуском у виробництво. Функції спільної роботи на платформі дозволили обговорювати та отримувати зворотній зв'язок у реальному часі, пришвидшивши прийняття рішень та забезпечивши узгодженість дій усіх членів команди.

Виробництво: на етапі виробництва Miro слугував інструментом управління проектом. Були створені дошки завдань для відстеження прогресу, призначення завдань та управління дедлайнами. Це допомагало тримати команду організованою та зосередженою, а також гарантувало, що кожен знав про свої обов'язки та поточний стан проекту.

Тестування та ітерації: Miro також використовувався на етапі тестування. Відгуки та звіти про помилки збиралися та організовувалися на дошці Miro, що полегшувало перегляд та визначення пріоритетів для наступних ітерацій. Також були створені діаграми та блок-схеми для планування та візуалізації рішень виявлених проблем.

Запуск та постпродакшн: Після запуску гри Miro продовжували використовувати для планування оновлень та розширень. Було зібрано та впорядковано відгуки гравців, а майбутній розвиток планувався та візуалізувався на дошках Miro.

Завдяки використанню Miro процес розробки гри став більш спільним, організованим та наочним, що призвело до покращення комунікації, продуктивності та, зрештою, до кращого кінцевого продукту.

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РОЗРОБКА 2D ГРИ ІЗ ЗАСТОСУВАННЯМ ПЛАТФОРМИ MIRO

3.1 Використання Miro у проекті з розробки 2D ігор

3.1.1. Налаштування середовища для розробки ігор у Miro

У пункті 3.1.1 основна увага приділена налаштуванню середовища для розробки ігор у Miro, платформі для візуальної співпраці. У цьому розділі описано кроки, необхідні для налаштування Miro для проєкту з розробки ігор. Нижче наведено деталі:

Ознайомлення з платформою: Ми почали з ознайомлення команди з платформою Miro та її можливостями. Провели навчання і надали ресурси, щоб члени команди розуміли можливості Miro і те, як вона може підтримати їхню спільну роботу.

Налаштування проєкту: Ми створили новий проєкт, дошку в Miro, спеціально присвячену проєкту розробки гри. Це буде нашим центральним робочим простором, де команда буде співпрацювати, обмінюватися ідеями та керувати інформацією, пов'язаною з проектом.

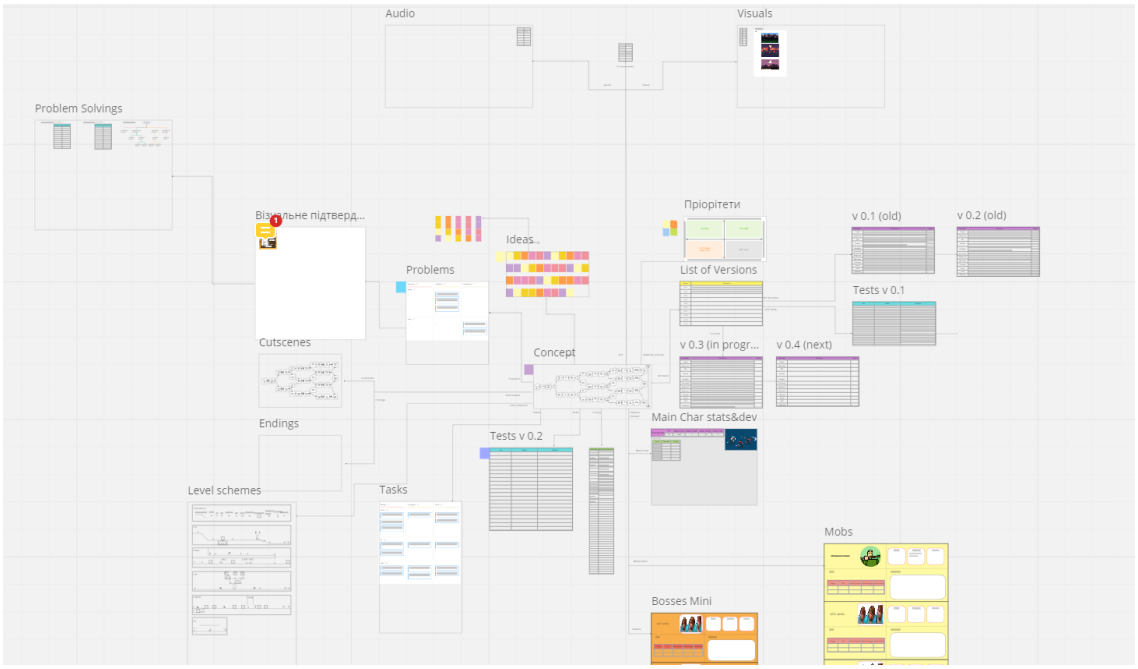

Рис.3.1. Вигляд дошки на одному із етапів розробки

Структура дошки: Визначили структуру і макет дошки Miro, виходячи з конкретних вимог проекту з розробки гри. Розглянули можливість створення окремих розділів або областей для різних аспектів, таких як дизайн гри, управління активами, відстеження завдань і документація.

Ігровий дизайн та концептуалізація: Ми використовували можливості Miro для спільної роботи, щоб полегшити обговорення геймдизайну та мозкові штурми. Використовували віртуальні дошки, стікери та інструменти для мапінгу, щоб фіксувати та впорядковувати ідеї, пов'язані з концепцією, механікою, персонажами, рівнями та загальним візуальним стилем гри.

Створення фреймворків та прототипів: Ми використовували інструменти малювання та фігур Miro для створення каркасів або низькоточних прототипів користувацького інтерфейсу, макетів рівнів та взаємодій гри. Ці візуальні представлення допомагають команді візуалізувати структуру та потік гри перед тим, як перейти до етапу реалізації.

Управління активами: Ми використовували можливості Miro для обміну файлами та інтеграції для управління ігровими ресурсами, такими як арт-активи, звукові файли та документація. Завантажували та впорядковували ресурси на дошці Miro, забезпечуючи легкий доступ та контроль версій для всієї команди.

Відстеження та планування завдань: Ми використовували дошки Kanban Miro, картки завдань або інші шаблони управління завданнями, щоб відстежували і планували завдання і етапи проекту. Створювали стовпчики, що представляють різні етапи процесу розробки, наприклад, "Зробити", "У процесі" та "Готово". Призначали завдання членам команди, встановлювали дедлайни та відстежували прогрес.

Документація та співпраця: Ми використовували функції спільної роботи Miro, щоб полегшити командну комунікацію та документування. Заохочували членів команди долучатися до створення документації, керівних принципів проектування, стандартів кодування та будь-якої іншої інформації,

пов'язаної з проектом. Ми використовували коментарі, згадки та співпрацю в реальному часі для заохочення зворотного зв'язку та обговорень.

Інтеграція з іншими інструментами: Ми вивчили можливості інтеграції з іншими інструментами, що використовуються у процесі розробки гри. Наприклад, інтегрували Miro з інструментами управління проектами,

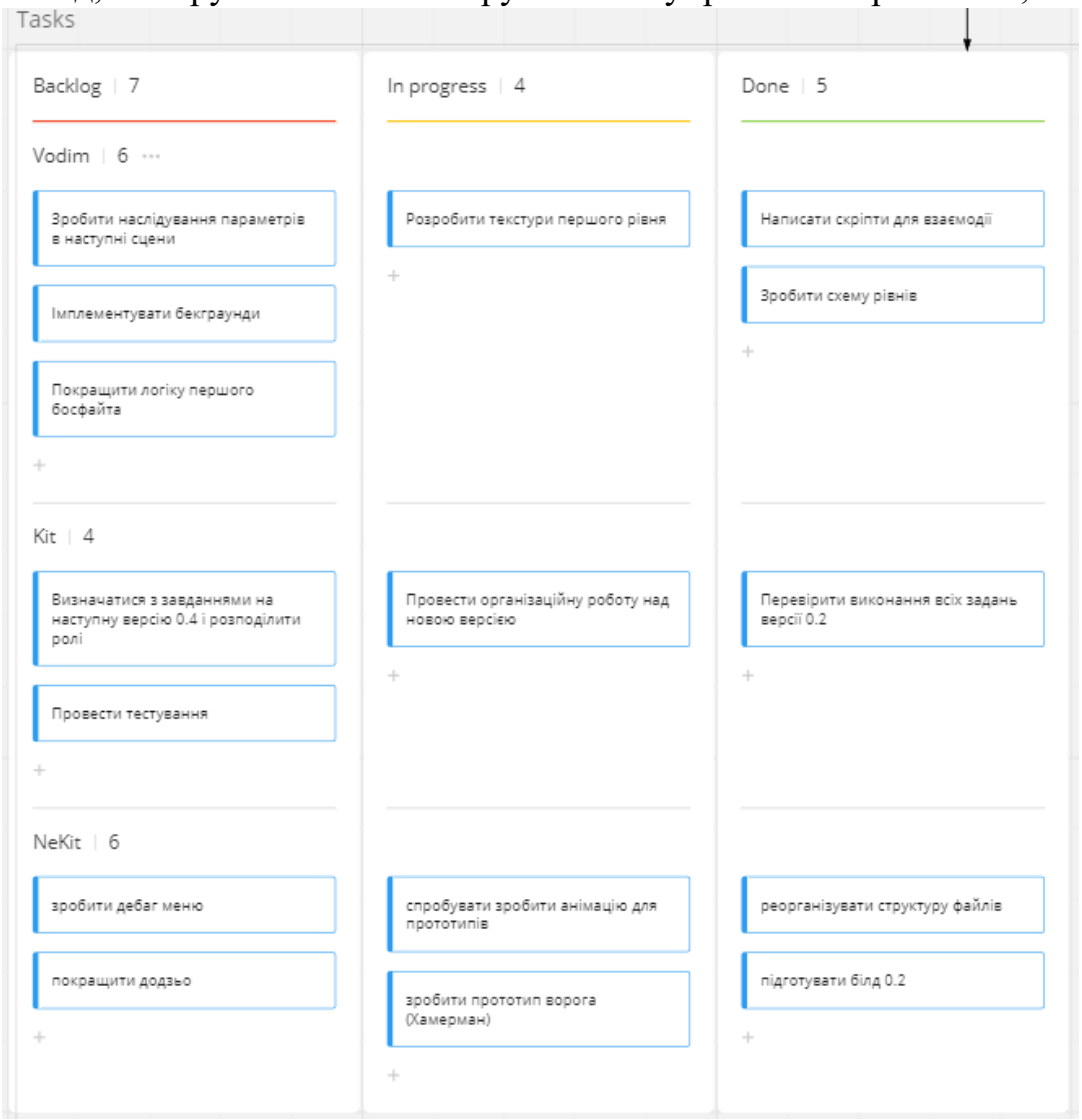

Рис 3.2. Вигляд Kanban дошки

системами контролю версій або комунікаційними платформами, щоб впорядкувати робочі процеси і забезпечити безперешкодну співпрацю між Miro та іншими важливими інструментами.

Навчання та підтримка: Забезпечували постійне навчання та підтримку членів команди щодо ефективного використання Miro. Вирішували будь-які питання та проблеми, що виникають, і заохочували до постійного навчання та

вивчення нових функцій Miro, які можуть покращити співпрацю та продуктивність.

Налаштувавши середовище для розробки ігор у Miro, команда може використовувати його функції для спільної роботи, щоб спростити комунікацію, покращити обмін ідеями та полегшити управління проектами. Візуальна природа Miro та гнучкі інструменти дозволяють команді ефективно планувати, проектувати та керувати процесом розробки, сприяючи створенню більш ефективного та спільного робочого середовища.

3.1.2. Сприяння командній співпраці за допомогою Miro

У пункті 3.1.2 основна увага приділяється тому, як Miro полегшує командну співпрацю у проекті розробки гри. У цьому розділі розглядаються різні способи, якими Miro покращує командну роботу та уможливлює ефективну співпрацю. Нижче наведено подробиці:

Співпраця у реальному часі: Miro надає середовище для спільної роботи в реальному часі, де члени команди можуть працювати разом одночасно. Кілька членів команди можуть одночасно отримувати доступ до дошки Miro і робити свій внесок, сприяючи синхронній співпраці та забезпечуючи миттєвий зворотній зв'язок і обмін ідеями.

Інструменти для візуальної співпраці: Miro пропонує ряд візуальних інструментів для спільної роботи, які сприяють творчості та полегшують ефективну комунікацію. Члени команди можуть використовувати віртуальні дошки, стікери, ментальні карти, діаграми та інші візуальні елементи для висловлення ідей, мозкового штурму та впорядкування інформації. Такий візуальний підхід заохочує залучення та покращує розуміння і узгодженість між членами команди.

Віддалена співпраця: Miro особливо корисний для віддалених команд, дозволяючи географічно розподіленим членам команди безперешкодно співпрацювати. Завдяки Miro члени команди можуть працювати разом незалежно від їхнього фізичного розташування, усуваючи обмеження часових

поясів або меж офісу. Віддалена співпраця дозволяє створити більш різноманітне та інклюзивне командне середовище.

Канали комунікації: Miro пропонує такі комунікаційні функції, як потоки коментарів, згадки та чат. Ці інструменти дозволяють членам команди обговорювати конкретні елементи гри, ділитися ідеями, ставити запитання та надавати відгуки. Чіткі та ефективні канали зв'язку в Miro спрощують співпрацю та гарантують, що всі залишаються в курсі подій.

Управління проектами та планування: Miro надає можливості управління проектами в рамках платформи для спільної роботи. Команда може використовувати дошки Kanban, картки завдань і часові рамки для планування, відстеження та управління прогресом проекту. Завдання, дедлайни та оновлення прогресу можна візуалізувати та керувати ними безпосередньо в Miro, забезпечуючи прозорість та підзвітність.

Фасилітація зустрічей: Miro може слугувати центральною платформою для проведення віртуальних зустрічей, семінарів та мозкових штурмів. Члени команди можуть використовувати інтегровані в Miro функції відеоконференцій або спільного доступу до екрану, щоб співпрацювати і обговорювати теми, пов'язані з грою, в реальному часі. Візуальні інструменти співпраці в Miro сприяють більш інтерактивним та продуктивним зустрічам.

Процеси зворотного зв'язку та рецензування: Miro спрощує процеси зворотного зв'язку та рецензування в рамках проекту з розробки гри. Члени команди можуть надавати зворотній зв'язок безпосередньо в Miro, використовуючи коментарі або анотації до конкретних елементів дизайну гри, каркасів або документації. Це сприяє чіткому та дієвому зворотному зв'язку, що призводить до ітеративних покращень.

Контроль версій та історія: Miro зберігає історію активності на дошці, дозволяючи членам команди повертатися до попередніх версій і змін, зроблених на дошці. Ця функція контролю версій дозволяє легко відстежувати розвиток проекту, зберігати записи обговорень і рішень, а також підтримує можливість відкотити зміни, якщо це необхідно.

v 0.1 (old)

Parameter	Description	Status
Project	Пов'язати сцени між собою за допомогою вбивства моба.	Done
Player Char	Поки безсмертний чолов'яга, який може махати мечем, бігати, стрибати. Модель та анімації взяти з ассетів. Налаштувати стати.	Done
NPC	<null>	Done
Enemies	Один вид ворогів на рівні "Forest of flowers", "Cave. Fights" та "Under siege". Не приводити їх в рух, лише задати параметри хп та атаки. Також вони мають атакувати гравця при наближенні у зону атаки.	Overdone
Bossfights	Зробити босс-файт з однією стадією. Босс має патрулювати та атакувати гравця при контакті.	Done
Enviroment	Додати землю та обмеження рівнів.	Done
Background	Однакові бекграунди на всі рівні, рандомка картинка, що не буде напрягати око	Done
Main menu	Створити меню за 8 кроків	Overdone
Cutscenes	Створити схему катсцен (miro)	Done
Sound	<null>	Done
HUD	<null>	Done
Additional		

Рис.3.3. Опис завдань версії 0.1 і їх статус виконання

Інтеграція із зовнішніми інструментами: Miro інтегрується з різноманітними зовнішніми інструментами, які зазвичай використовуються в процесі розробки ігор, такими як програмне забезпечення для управління проектами, інструменти для дизайну проєкту або системи відстеження завдань. Інтеграція з цими інструментами забезпечує безперебійний робочий процес, уникаючи необхідності повторного введення даних і підвищуючи продуктивність.

Шаблони та робочі процеси для спільної роботи: Miro пропонує ряд готових шаблонів і робочих процесів, спеціально розроблених для спільної роботи над проектами з розробки ігор. Ці шаблони є відправною точкою та

найкращими практиками для організації інформації, проектування ігрових елементів, управління завданнями та полегшення співпраці.

Використовуючи функції спільної роботи Miro, команда розробників ігор може покращити комунікацію, стимулювати творчість і спростити управління проектами. Співпраця в режимі реального часу, візуальні інструменти та інтеграції Miro підтримують ефективну командну роботу, уможливлюючи ефективну співпрацю, обмін ідеями та прийняття рішень. Платформа слугує центральним центром для командної співпраці, сприяючи більш продуктивному та згуртованому процесу розробки ігор.

3.2. Приклад розробки 2D гри з використанням Unity, C# та Miro

3.2.1. Розробка та планування ігор за допомогою Miro

У пункті 3.2.1 основна увага приділяється використанню Miro для проектування та планування ігор. У цьому розділі розглядається, як Miro підтримує процес проектування та планування гри, полегшуючи організацію та візуалізацію ігрових елементів. Нижче наведено деталі:

Документація для геймдизайну: Miro слугує центральною платформою для створення та організації нашої ігрової документації. Члени команди можуть використовувати віртуальні дошки, стікери та текстові поля в Miro, щоб фіксувати та вдосконалювати ідеї, пов'язані з концепцією гри, механікою, рівнями, персонажами та елементами розповіді. Таке візуальне представлення документації з ігрового дизайну допомагає команді узгодити своє розуміння та уявити загальну структуру гри.

Візуальний мозковий штурм: Візуальні інструменти співпраці Miro дозволяють команді проводити мозкові штурми та генерувати ідеї візуально. Команда може використовувати ментальні карти, блок-схеми та діаграми, щоб дослідити різні можливості дизайну, намітити ігрові механіки та встановити взаємозв'язки між елементами гри. Візуальний мозковий штурм у Miro заохочує творчість і сприяє спільному розумінню між членами команди.

Wireframing та дизайн рівнів: Miro надає інструменти для створення каркасів та чорнових ескізів користувацького інтерфейсу гри та макетів рівнів. Команда може використовувати фігури, іконки та інструменти для малювання в Miro, щоб окреслити розміщення ігрових елементів, таких як кнопки, меню та інтерактивні об'єкти. Wireframing у Miro допомагає команді візуалізувати структуру гри та користувацький потік перед тим, як перейти до етапу реалізації.

Розкадровка: Miro підтримує створення розкадровок, які є візуальним представленням сюжету та ігрових послідовностей гри. Команда може використовувати інструменти малювання та стікери Miro для створення кадрів розкадровки та розташування їх у послідовності, яка відображає потік подій у грі. Розкадровка в Miro полегшує ефективну розповідь і допомагає забезпечити цілісний та захопливий ігровий досвід.

Організація ігрових ресурсів: Можливості Miro для обміну файлами та інтеграції дозволяють команді організувати ігрові ресурси на дошці Miro. Художні ресурси, звукові файли, довідкові матеріали та інші відповідні ресурси можуть бути завантажені в Miro і пов'язані з відповідними ігровими елементами. Така централізована організація ресурсів у Miro забезпечує легкий доступ до них і допомагає підтримувати контроль версій для всієї команди.

Ітерації та керування версіями: Спільна природа Miro підтримує ітеративний дизайн гри та створення версій. Команда може повторювати елементи геймдизайну, вдосконалюючи та розвиваючи їх на основі відгуків та інсайтів. Функції контролю версій Miro дозволяють членам команди повертатися до попередніх ітерацій, порівнювати зміни та відстежувати прогрес у розробці гри.

Спільне обговорення дизайну: Miro полегшує спільний перегляд дизайну та збір відгуків. Члени команди можуть додавати коментарі, анотації або стікери безпосередньо до елементів ігрового дизайну в Miro, ділячись своїми думками та пропозиціями. Цей процес зворотного зв'язку в режимі

реального часу в Miro забезпечує спільний та ітеративний підхід до вдосконалення ігрового дизайну.

Візуалізація дизайнерських рішень: Miro допомагає візуалізувати дизайнерські рішення та їхній вплив на гру в цілому. Команда може створювати матриці рішень, концептуальні карти або візуальні представлення компромісів у дизайні в Miro. Така візуалізація допомагає зрозуміти наслідки вибору дизайну та забезпечує узгодженість дій між членами команди.

Майстер-класи з ігрового дизайну: Miro можна використовувати як платформу для проведення віртуальних воркшопів з геймдизайну. Члени команди можуть збиратися на спільній дошці Miro, використовуючи візуальні інструменти співпраці та інтерактивні функції для полегшення мозкового штурму, генерації ідей та спільного прийняття рішень. Можливості воркшопів Miro дозволяють ефективно співпрацювати віддалено і гарантують, що всі члени команди можуть брати активну участь у процесі проектування.

Доступність проектної документації: Хмарна природа Miro забезпечує легкий доступ до проектної документації з будь-якого місця і в будь-який час. Члени команди можуть звертатися до дошки Miro, щоб отримати доступ до проектної документації, переглянути проектні рішення та бути в курсі прогресу в розробці гри. Така доступність сприяє спільному та інклюзивному процесу розробки, дозволяючи членам команди робити свій внесок незалежно від їхнього фізичного місцезнаходження.

Використовуючи Miro для геймдизайну та планування, команда може ефективно збирати, організовувати та візуалізувати елементи геймдизайну. Функції спільної роботи та візуальні інструменти Miro сприяють творчості, полегшують мозковий штурм і дозволяють команді приймати обґрунтовані дизайнерські рішення. Гнучкість і доступність платформи сприяють більш ефективному та спільному процесу геймдизайну, що в кінцевому підсумку призводить до створення добре структурованої та захопливої гри.

3.2.2. Кодування та тестування з використанням Unity та C#

У пункті 3.2.2 основна увага приділяється використанню Unity та мови програмування C# для кодування та тестування гри. Ось деякі деталі:

Палітра тайлів та створення середовища: Ми використали систему tilemap Unity і створили палітру тайлів для проектування та побудови ігрового середовища. Використовуючи різні ресурси плиток, ми створили рівні, включаючи платформи, перешкоди та інтерактивні елементи. Палітра плиток дозволила легко розміщувати та компонувати тайли, забезпечуючи візуально привабливе та функціональне середовище для ігрового процесу.

Фізика та зіткнення: Ми інтегрували фізику в ігрове середовище, щоб створити реалістичну взаємодію між ігровими об'єктами. Ми впровадили фізичний рушій Unity для обробки гравітації, виявлення зіткнень та динаміки жорстких тіл. Це дозволило об'єктам реагувати на силу, демонструвати реалістичні рухи та точно зіштовхуватися один з одним, покращуючи загальний ігровий досвід.

Механіка головного героя та ворогів: Ми розробили головного героя, оснастивши його елементами керування рухом, стрибками та механікою атаки. Рухи персонажа були чутливими та плавними, що дозволило гравцям легко орієнтуватися в ігровому світі. Ми реалізували анімацію атаки та бойову логіку, щоб головний герой міг ефективно взаємодіяти з ворогами.

Штучний інтелект ворогів та бої: ми розробили та впровадили системи штучного інтелекту ворогів, які керували поведінкою та діями різних типів ворогів. Вороги демонстрували патерни руху, стратегії нападу та реакції на дії гравця. Ми створили ворогів ближнього бою з атаками з близької відстані та ворогів-лучників, здатних стріляти стрілами на відстані. Штучний інтелект ворогів додав складності та різноманітності в ігровий процес.

Система здоров'я та смерть персонажа: Ми додали систему здоров'я як для головного героя, так і для ворогів. Кожен персонаж мав атрибут здоров'я (HP), який зменшувався при отриманні ушкоджень. Коли рівень здоров'я

```
    [SerializeField] int maxHp = 1;  "100"
    int currentHp;
    [SerializeField] int trapdamage = 1;  Unchanged
    [SerializeField] float delay = 3f;  "1.5"
    private Animator            m_animator;

    Event function  sh4zzard
    void Start()
    {
        m_animator = GetComponent<Animator>();
        currentHp = maxHp;
    }

    Event function  sh4zzard
    public void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Trap"))
        {
            Debug.Log(currentHp);
            currentHp -= trapdamage;
        }
        if (currentHp <= 0)
        {
            m_animator.SetTrigger( name: "Death");
            Invoke( methodName: "LoadGameOverScene", delay);
        }
    }
    Frequently called  2 usages  sh4zzard
    public void TakeDamage2(int damage)
    {
        m_animator.SetTrigger( name: "Hurt");
        currentHp -= damage;
        Debug.Log( message: "HK got dmg");
        Debug.Log(currentHp);
        if (currentHp <= 0)
        {
            m_animator.SetTrigger( name: "Death");
            Invoke( methodName: "LoadGameOverScene", delay);
        }
```

Рис.3.4. Фрагмент з коду, який відповідає за здоров’я ігрового персонажа

головного героя досягав нуля, він помирав, викликаючи відповідні умови завершення гри. Аналогічно, коли рівень здоров'я ворогів досягав нуля, вони зазнавали поразки, сприяючи прогресу гравця.

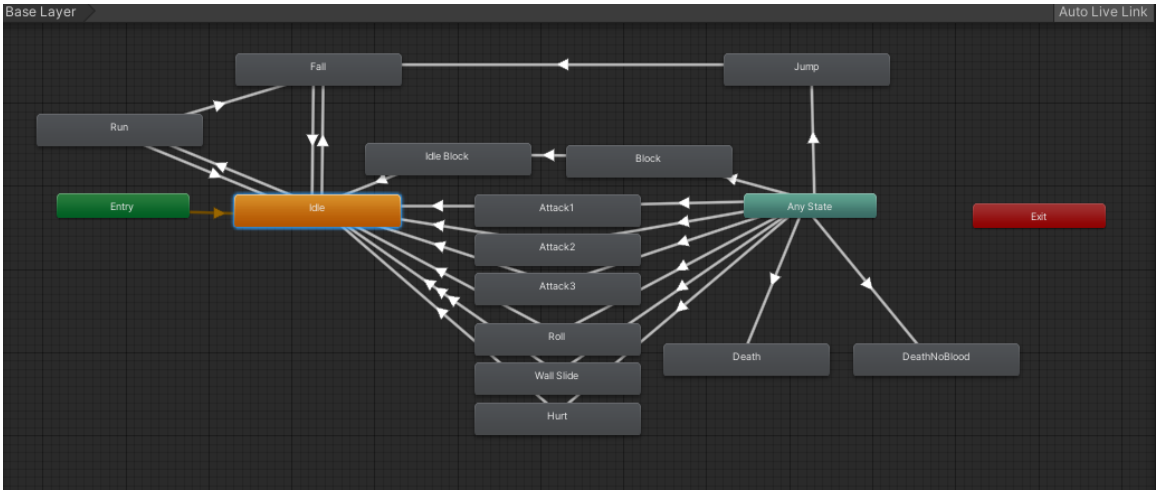

Рис.3.5. Вигляд аніматора для ігрового персонажа

Додаткові можливості: На додаток до основної механіки, ми реалізували кілька додаткових можливостей, щоб збагатити ігровий досвід. Ці функції включали взаємодію з драбинами для вертикального переміщення, взаємодію

з неігровими персонажами (NPC) для діалогів і проходження квесту, пастки для створення перешкод і викликів.

Також ми додали елементи користувацького інтерфейсу, такі як головне меню і меню паузи для покращення зручності використання гри, а також для просування оповіді зробили cut-scene.

```
64      }
65
66      //public void EnemyAttacks()
        Event function  Hammak +1
67      void Update()
68      {
69          timer += Time.deltaTime;
70
71          Vector2 target = new Vector2(player.position.x, rb.position.y);
72
73          if (bossIndicator)
74          {
75              tryBossAttack();
76          }
77      }
78
        Frequently called  1 usage  Hammak +1
79      private void tryBossAttack()
80      {
81          Boss.stages currentStage = Boss.stages.first;
82          float currentStageCooldown = Cooldown;
83
84          if (currentHealth <= boss.secondPhaseHPThreshold * maxHealth)
85          {
86              currentStage = Boss.stages.second;
87              currentStageCooldown = boss.secondPhaseCooldown;
88          }
89
90          if (Vector2.Distance( (Vector2) a: player.position,  b: rb.position) <= attackRange)
91          {
92              if (timer >= currentStageCooldown)
93              {
94                  Debug.Log( message: "Stage: " + currentStage + " CD: " + currentStageCooldown);
95                  BossAttack(currentStage);
96                  timer = 0f;
97              }
98          }
99      }
100
        Frequently called  1 usage  Hammak +1
```

Рис.3.6. Фрагмент коду із скріпта для поведінки противників

Тестування та налагодження: Протягом усього етапу кодування ми проводили ретельне тестування та налагодження, щоб забезпечити функціональність, стабільність та збалансованість гри. Ми проводили систематичні сеанси ігрового тестування, щоб виявити та виправити будь-які проблеми, доопрацювати механіку ігрового процесу та оптимізувати продуктивність. Тестування включало перевірку коректності рухів персонажів, механіки атак, поведінки ворогів, системи здоров'я та загальної функціональності додаткових можливостей. Хоч ми і намагалися швидко виправляти будь-які помилки і баги, які виникали в процесі розробки, з появою

і доданням нових елементів і механік, несправностей виникає більше і більше. Виправлення цих несправностей вимагає значних ресурсів.

Рис.3.7. Меню налаштувань

Реалізувавши та протестувавши ці функції, ми успішно втілили нашу гру в життя за допомогою Unity та C#. Поєднання надійної палітри тайлів, взаємодії на основі фізики, добре реалізованої механіки персонажів, штучного інтелекту ворога, системи здоров'я та додаткових функцій призвело до захопливого та захоплюючого ігрового процесу. Процес тестування та налагодження забезпечив якість, стабільність та дотримання запланованого дизайну гри. Загалом, цей етап розробки зробив значний внесок у реалізацію нашої 2D-гри та продемонстрував нашу майстерність у використанні Unity та C# для створення захопливого ігрового досвіду.

3.2.3. Співпраця у команді та управління проектом за допомогою Miro

У пункті 3.2.3 основна увага приділяється використанню Miro для командної співпраці та управління проектами у розробці ігор. У цьому розділі розглядається, як Miro покращує командну співпрацю, полегшує управління проектами та сприяє згуртованому та ефективному процесу розробки. Ось подробиці:

Командна комунікація та координація: Miro слугує центральною платформою для командної комунікації та координації. Члени команди можуть використовувати функціонал чату Miro, потоки коментарів та згадки для обміну ідеями, надання оновлень та пошуку роз'яснень. Miro сприяє співпраці в режимі реального часу, гарантуючи, що всі залишаються поінформованими та узгодженими.

Планування проєктів та складання дорожніх карт: Функції спільної роботи Miro підтримують планування проектів та складання дорожніх карт. Команда може створювати візуальні дорожні карти, діаграми Ганта або шаблони часових шкал в Miro, щоб окреслити етапи, завдання і залежності проекту. Таке візуальне представлення допомагає команді зрозуміти графік проекту, визначити пріоритетність завдань і ефективно розподілити ресурси.

Призначення та відстеження завдань: Miro надає можливості управління завданнями, дозволяючи команді призначати завдання, встановлювати терміни виконання та відстежувати прогрес. Kanban-дошки, картки завдань або інші шаблони управління завданнями в Miro дозволяють команді візуалізувати робочий процес, виявляти вузькі місця та забезпечувати виконання завдань у визначені терміни.

Гнучка та ітеративна розробка: Miro підтримує гнучкі та ітеративні підходи до розробки. Команда може використовувати дошки Kanban або гнучкі шаблони Miro для планування та відстеження спринтів, бэклогу та користувацьких історій. Візуальна природа Miro сприяє прозорості, дозволяючи команді візуалізувати прогрес кожного спринту та адаптувати плани за потреби.

Спільне прийняття рішень: Miro полегшує спільне прийняття рішень в команді розробників. Команда може використовувати функції голосування, системи пріоритетів або шаблони прийняття рішень Miro для колективного прийняття рішень щодо особливостей гри, вибору дизайну або пріоритетів проекту. Такий спільний процес прийняття рішень гарантує, що всі члени

команди мають право голосу, а також сприяє формуванню почуття причетності та спільної відповідальності.

Спільна робота над дизайном та ілюстраціями: Miro підтримує спільну роботу над ігровим дизайном та ілюстраціями. Члени команди можуть використовувати візуальні інструменти співпраці Miro, щоб ділитися концепціями дизайну, ескізами або посиланнями на мистецтво. Відгуки та коментарі можна надавати безпосередньо до елементів дизайну в Miro, гарантуючи, що вся команда залучена до творчого процесу.

Документація та обмін знаннями: Miro слугує платформою для документації та обміну знаннями. Команда може створювати та організовувати документацію на дошці Miro, включаючи керівні принципи дизайну, стандарти кодування, нотатки зустрічей та найкращі практики. Це центральне сховище інформації гарантує, що члени команди мають легкий доступ до відповідних знань, пов'язаних з проектом.

Фасилітація зустрічей: Miro можна використовувати для фасилітації віртуальних зустрічей. Члени команди можуть використовувати інтеграцію відеоконференцій або можливості спільного доступу до екрану Miro для проведення віртуальних зустрічей, мозкових штурмів або оглядів дизайну. Візуальні інструменти співпраці Miro дозволяють проводити інтерактивні та продуктивні зустрічі, сприяючи активному залученню та участі.

Процеси зворотного зв'язку та рецензування: Miro впорядковує процеси зворотного зв'язку та рецензування в рамках проекту з розробки ігор. Члени команди можуть надавати зворотній зв'язок безпосередньо в Miro, використовуючи коментарі, анотації або стікери до певних елементів проекту. Цей процес зворотного зв'язку в реальному часі полегшує ітеративні покращення та забезпечує спільний та інклюзивний процес розробки.

Міжфункціональна співпраця: Miro підтримує міжфункціональну співпрацю в команді розробників. Художники, дизайнери, програмісти та інші члени команди можуть співпрацювати на спільній дошці Miro, ділячись своїм досвідом та працюючи над завданнями, що вимагають міждисциплінарної

співпраці. Miro підтримує мультидисциплінарний підхід, сприяючи цілісному та інтегрованому процесу розробки.

3.3 Оцінка ефективності використання Miro у проекті

3.3.1. Оцінка ефективності та продуктивності команди

У пункті 3.3.1 основна увага приділяється оцінці ефективності та продуктивності команди у проекті розробки гри, зокрема, у зв'язку з використанням Miro. У цьому розділі описано методи та метрики, що використовуються для оцінки ефективності роботи команди та впливу Miro на продуктивність. Ось деталі:

Метрики для оцінки ефективності команди: Ми визначили метрики для вимірювання ефективності команди, такі як рівень виконання завдань, дотримання дедлайнів і загальна продуктивність. Ці показники можна відстежувати за допомогою функцій управління завданнями Miro, дошок Kanban або інших інструментів управління проектами, інтегрованих з Miro.

Відстеження часу та розподіл ресурсів: Ми відстежували час, витрачений командою на різні завдання та види діяльності, використовуючи інструменти або функції Miro для відстеження часу. Аналізуйте розподіл ресурсів, щоб переконатися, що члени команди ефективно використовують свій час і що робочі навантаження збалансовані в команді.

Оцінка співпраці та комунікації: Ми оцінили ефективність командної співпраці та комунікації за допомогою Miro. Зібрали відгуки членів команди за допомогою опитувань або інтерв'ю, щоб оцінити вплив Miro на ефективність комунікації, обмін знаннями та співпрацю між різними ролями та дисциплінами.

Прогрес і прозорість виконання завдань: Оцінили видимість і прозорість прогресу виконання завдань в Miro. Оцінили, наскільки добре функції Miro, такі як дошки Kanban або картки завдань, дозволяють команді відстежувати і візуалізувати прогрес виконання завдань, виявляти вузькі місця і розуміти загальний статус проекту.

Ефективність зустрічей: Оцінили ефективність нарад, проведених за допомогою відеоконференцій та функцій спільної роботи Miro. Оцінили такі фактори, як тривалість зустрічі, рівень залученості та здатність ефективно приймати рішення або вирішувати питання, пов'язані з проектом.

Вплив на прийняття рішень: Проаналізували вплив Miro на процеси прийняття рішень в команді. Оцінили, чи функції спільного прийняття рішень Miro, такі як голосування або система пріоритетів, покращили здатність команди приймати обґрунтовані рішення, узгоджувати пріоритети, а також сприяли формуванню почуття причетності та спільної відповідальності.

Зворотний зв'язок та ітеративне вдосконалення: Оцінили, наскільки ефективно функції зворотного зв'язку та ітерацій були використані в Miro для покращення процесу розробки гри. Оцінили, чи зворотній зв'язок, наданий у Miro, призвів до ітеративного покращення дизайну, коду чи інших елементів проекту, що призвело до створення більш досконалої та відшліфованої гри.

Міжфункціональна співпраця: Оцінили ефективність міжфункціональної співпраці за допомогою Miro. Проаналізували, чи сприяла Miro співпраці та обміну знаннями між різними ролями та дисциплінами в команді, покращуючи загальні результати проекту та інтеграцію різних аспектів розробки гри.

Порівняння з попередніми проектами: Порівняли ефективність і продуктивність команди в поточному проекті з використанням Miro з попередніми проектами, які не використовували Miro або подібні інструменти для спільної роботи. Проаналізували відмінності в ключових показниках, таких як рівень виконання завдань, ефективність зустрічей або процесів прийняття рішень, щоб визначити вплив Miro на продуктивність команди.

Якісний зворотній зв'язок та рефлексія: Зібрали якісні відгуки та роздуми членів команди про їхній досвід роботи з Miro. Провели інтерв'ю або обговорення у фокус-групах, щоб отримати уявлення про переваги, виклики та пропозиції щодо подальшого вдосконалення використання Miro для командної співпраці та підвищення продуктивності.

Оцінюючи ефективність та продуктивність команди за допомогою метрик, відгуків та роздумів, можна оцінити вплив Miro на проект розробки гри. Ця оцінка дає цінну інформацію про ефективність Miro як платформи для спільної роботи, визначає сфери для вдосконалення та допомагає приймати рішення щодо використання подібних інструментів у проєктах з розробки ігор у майбутньому.

3.3.2. Зворотній зв'язок щодо використання Miro

У пункті 3.3.2 основна увага приділяється збору відгуків та рефлексії щодо використання Miro у проекті розробки гри. У цьому розділі розглядаються відгуки, отримані від членів команди щодо їхнього досвіду роботи з Miro, а також аналізуються сильні сторони, виклики та потенційні покращення, пов'язані з його використанням. Ось подробиці:

Збір відгуків: Ми зібрали відгуки від членів команди про їній досвід роботи з Miro протягом усього проекту розробки гри. В нашому випадку ми використовували формат інтерв’ю. Кожен із учасників проекту задоволений функціоналом Miro, зазначає про підвищення ефективності виконання завдань, завдяки сформованими нами принципам командної розробки, і про зручність й спрощення комуніцікації і відслідковування процесу завдяки написаній документації.

Сильні сторони Miro: В ході нашого інтерв’ю, ми спробували визначити сильні сторони і переваги Miro у проєкті розробки гри. Кожен із опитаних нами учасників проєкту зазначав як позитивні фактори: Kanban дошки, які підвищують якість і швидкість виконання поставлених задач, документації виконаних завдань в кожній версії і наявність чітких і зрозумілих схем, які дозволяють завжди чітко розуміти напрямок і ціль нашого проекту.

Покращена співпраця і комунікація: ми також опитали учасників проєкту про вплив на їх співпрацю і комунікацію наших інструментів командної розробки. Кожен із учасників зазначав, що завдяки Miro підвищився рівень співпраці та комунікації між членами команди. Завдяки схемам і документації нівелювалися процеси безкорисної комунцікації і

збільшило часові рамки для безпосередньо розробки. Також завдяки інструментам вирішення проблем, конфліктів між учаниками проєкту стало значно менше.

Візуальне представлення ідей: ми зібрали відгуки щодо візуального представлення ідей та концепцій в Miro. Візуальні інструменти співпраці Miro, такі як віртуальні дошки, стікери та діаграми, допомогли членам команди висловлювати свої ідеї, проводити мозкові штурми та організовувати інформацію. Ці візуальні репрезентації значно покращили ясність, розуміння та узгодженість у команді.

Виклики та обмеження: В ході нашого інтер'ю, ми також зібрали інформацію про виклики та обмеження, що виникли під час використання Miro у проєкті з розробки гри. В рамках нашої розробки критичних обмежень чи недостатностей функціоналу виявлено не було. Єдиним зауваженням було погрішення швидкості роботи платформи із збільшенням записанної інформації.

Інтеграція з робочим процесом: В ході нашого інтер'ю, ми також дали можливість учаникам оцінити, наскільки добре Miro інтегрований з існуючими робочими процесами та інструментами команди. Загалом, враховуючи відносно невеликі розміри нашого проєкту, проблем із інтеграцією з робочим процесом не виникало. Учасники доволі швидко звикли записувати документацію до Miro і зазначали позитивні зміни завдяки цьому.

Пропозиції щодо вдосконалення: В ході нашого інтер'ю, ми також дали можливість учаникам висловити пропозиції та рекомедації щодо подальшого покращення використання Miro у майбутніх проектах з розробки ігор. Учаники зазначали необхідність додаткових сховищ для збереження графічної та аудіо інформації. Також, було зазначено необхідність оптимізації робочої дошки, для підвищення швидкодійності і кращого ефекту сприйняття.

Роздуми про ефективність та продуктивність: Аналізуючи відгуки про те, як Miro вплинув на рівень виконання завдань, ефективність зустрічей, процеси прийняття рішень і загальні результати проекту, можна сказати, що

принципи командної розробки сформовані нами й інструменти, які використовувалися в ході розробки, дали позитивний результат, добре вплинули на виконання завдань, підвщили ефективність та збільшили продуктивність.

Вивчені уроки та найкращі практики: Узагальнюючи, можна сказати, що методи командної розробки позитивно впливають на роботу команди, всі члени команди залишаються в хорошому вражені від командної роботи. Також, збільшується продуктивність, якість і швидкість виконання завдань.

Збираючи відгуки та розмірковуючи над використанням Miro, команда може отримати цінну інформацію про сильні сторони, виклики та сфери для вдосконалення, пов'язані з його використанням. Ці відгуки та роздуми створюють основу для постійного вдосконалення та інформують про майбутні рішення щодо вибору та використання інструментів для спільної роботи у майбутніх проектах з розробки ігор.

ВИСНОВКИ

Кваліфікаційна робота має своєю темою застосування візуальної платформи Miro для командної розробки 2D гри. У першому, другому та третьому розділі проекту було досліджено можливості Miro для покращення співпраці, комунікації та управління проектами в процесі розробки ігор. Основні результати та висновки, що ґрунтуються на розглянутих у дипломній роботі питаннях, є наступними:

Перший розділ роботи присвячений розгляду передумов, актуальності та сфери дослідження. У розділі встановлено важливість ефективного розвитку команди в проектах з розробки ігор та визначено платформу Miro як потенційне рішення для покращення співпраці та комунікації. Також, розділ включає опис та аналіз альтернативних інструментів для досягнення цілей дипломної роботи. Для цього різні інструменти, включаючи Trello, Asana та Jira, були порівняні з Miro з точки зору їхніх функцій, можливостей та придатності для командної розробки в 2D ігровому проєкті.

У першому розділі ми на основі аналізу літератури зробили огляд розробки ігор та важливості командної співпраці у цьому процесі. Ми представили основи розробки 2D ігор, роль командної співпраці та дослідили платформу розробки Unity і мову C#. Крім того, ми представили візуальну платформу Miro та її застосування у спільних проектах.

Варто відзначити міждисциплінарну природу розробки ігор, де художники, дизайнери, програмісти та інші фахівці працюють разом над створенням захоплюючого ігрового досвіду. Підкреслюючи важливість ефективної командної роботи та комунікації, відзначаємо, що співпраця в командах розробників ігор призводить до успішних ігрових проектів.

Платформа для розробки ігор Unity має потужні інструменти та функціональні можливості для створення візуально привабливих та інтерактивних 2D ігор. Мова програмування C# застосовується у розробці ігор на Unity, при цьому варто підкреслити її важливість у написанні сценаріїв поведінки гри та реалізації механік.

Візуальна платформа Miro - інструмент для покращення командної співпраці у проектах з розробки ігор. Miro має такі можливості та функції як віртуальні дошки, стікери та співпраця в реальному часі, які сприяють ефективній комунікації та обміну ідеями в команді. Широкого розповсюдження набуло практичне застосування Miro у спільних проектах, що демонструє її здатність оптимізувати робочі процеси та сприяти візуальному представленню ідей.

У Розділі 1 також було розглянуто сутність і проблематику розробки ігор, командної співпраці, платформи розробки Unity, мови C# та візуальної платформи Miro. Розуміння основ розробки ігор, важливості співпраці та інструментів, доступних для підтримки командної роботи, створює основу для вивчення застосування Miro у командній розробці 2D гри.

Другий розділ включає дослідження та аналіз і моделювання принципів командної роботи із застосуванням платформи Miro. Ми визначилися з вимогами до гри і провели роботу з пошуку необхідних інструментів і ресурсів. Ми провели аналітичну роботу, спроектували процес розробки гри і змоделювали середовище для командної співпраці.

Аналіз показав, що створення гри вимагає ретельного планування та координації. Кожен етап процесу, від розробки концепції до пост-продакшну, має свої унікальні виклики та вимоги. Ефективність процесу розробки значною мірою залежить від чіткої комунікації, організованого управління завданнями, а також здатності адаптуватися та ітерації на основі зворотного зв'язку та тестування.

Ефективне використання інструментів, тобто правильно підібрані інструменти можуть значно покращити процес розробки гри. Unity було визначено як потужну та універсальну платформу розробки, здатну задовольнити специфічні вимоги гри. Тим часом, функції візуальної співпраці Miro виявилися надзвичайно корисними для полегшення мозкового штурму, планування, управління завданнями та збору зворотного зв'язку.

Проведений аналіз підкреслив важливість співпраці та командної роботи у розробці ігор. Використання такого інструменту, як Miro, допомогло тримати всіх в курсі подій, уможливило зворотній зв'язок та обговорення в режимі реального часу, а також полегшило візуалізацію та розуміння складних ідей і планів.

Третіл розділ (практична частина) демонструє застосування Miro у проєкті з розробки 2D-гри. Вона охоплювала налаштування середовища розробки ігор, полегшення командної співпраці, використання Miro для дизайну та планування ігор, кодування та тестування з використанням Unity та C#, а також загального управління проектом. У цьому розділі висвітлено переваги візуальних інструментів співпраці Miro, каналів комунікації в реальному часі та функцій управління завданнями для оптимізації процесу розробки та підвищення ефективності й продуктивності команди.

У роботі надано оцінку ефективності використання Miro у проекті. Загальна оцінка ефективності включає оцінку ефективності та продуктивності команди, збір відгуків про використання Miro, а також пропозиції щодо сильних і слабких сторін проєкту, проблеми та потенційні покращення, пов'язані з його використанням.

Впродовж усього проєкту, описаного у третьому розділі, платформа Miro відігравала центральну роль у полегшенні командної співпраці, впорядкуванні комунікації та покращенні управління проектами в процесі розробки ігор. Практичне застосування Miro продемонструвало її ефективність у наступних сферах:

Дизайн та планування ігор: Miro забезпечує візуальну платформу для спільної роботи над документацією, фреймворком, розкадровкою та управлінням активами. Вона підтримувала ітеративні процеси проектування, сприяла міжфункціональній співпраці та покращувала процес прийняття проектних рішень.

Кодування та тестування: Miro, у поєднанні з Unity та C#, просуває ефективні практики кодування та тестування. Це полегшило спільну роботу

над кодом, документування та контроль версій. Miro підтримує модульне тестування, інтеграційне тестування та налагодження, забезпечуючи розробку високоякісного коду без помилок.

Командна співпраця та управління проектами: Miro слугує центром для командної співпраці та управління проектами. Це покращувало комунікацію, сприяло командній роботі та підтримувало гнучкі методології розробки. Такі функції Miro, як відстеження завдань, візуальне планування та спільне прийняття рішень, сприяли ефективному виконанню проектів.

Оцінка ефективності Miro в проекті показала позитивні результати. Miro покращила співпрацю, комунікацію та процес прийняття рішень в команді. Платформа забезпечила централізовану платформу для обміну ідеями, збору відгуків та відстеження прогресу. Візуальна природа Miro покращила розуміння та узгодженість між членами команди, що призвело до покращення результатів проекту. Відгуки та роздуми членів команди підкреслили сильні сторони Miro у сприянні співпраці, комунікації та управлінні проектами.

На закінчення, кваліфікаційна робота продемонструвала, що застосування візуальної платформи Miro в командній розробці 2D гри має значні переваги. Функції спільної роботи, візуальні інструменти та можливості управління завданнями Miro сприяють ефективній командній роботі, оптимізують робочі процеси та покращують результати проекту. Поєднання Miro з Unity та C# забезпечує комплексне рішення для проектів з розробки ігор, дозволяючи безперешкодно інтегрувати дизайн, кодування, тестування та управління проектами.

Кваліфікаційна робота має практичне значення, демонструючи практичне застосування та ефективність Miro в контексті розробки ігор. Отримані результати можуть використовуватися при розробці навчальних програм, лабораторних робіт, посібників тощо припідготовці/перепідготовці майбутніх команд розробників ігор, які прагнуть оптимізувати співпрацю, комунікацію та управління проєктами. Обмеження та проблеми, що

виникають при використанні Miro, надають можливості для подальших досліджень та вдосконалення.

Загалом, кваліфікаційна робота підкреслює важливість використання інструментів для спільної роботи, таких як Miro, для покращення командного розвитку в проектах з розробки ігор. Успішна інтеграція Miro в проект розробки 2D-гри демонструє його потенціал для позитивної зміни командної співпраці та управління проектами в ігровій індустрії. Використовуючи інноваційні інструменти, такі як Miro, невеликі команди розробників ігор можуть повністю розкрити свій творчий потенціал, оптимізувати процеси та ефективно створювати високоякісні ігри.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. GamesIndustry.biz. GamesIndustry.biz presents. The Year in Numbers 2020. 2020. Режим доступу до ресурсу: https://www.gamesindustry.biz/articles/2020-12-21-gamesindustry-biz-presents-the-year-in-numbers-2020.
2. Whitson, J. R., The New Spirit of Capitalism in the Game Industry, Television & New Media. 2019. Vol.20(8). P. 789–801.
3. Newzoo. Mobile Revenues Account for More Than 50% of the Global Games Market as It Reaches $137.9 Billion in 2018. 2018. Режим доступу до ресурсу: https://newzoo.com/insights/articles/global-games-marketreaches-137-9-billion-in-2018-mobile-games-take-half/.
4. Czech Game Developers Association. Czech video game industry PC, console and mobile game developers in Czech Republic 2020. Режим доступу до ресурсу: https://gda.cz/wpcontent/uploads/2020/07/GDACZ_Study_2020.pdf.
5. Polish Agency for Enterprise Development. The game industry of Poland — Report 2020. Режим доступу до ресурсу: https://gic.gd/polish-game-industry/.
6. Miro [Електронний ресурс] – Режим доступу до ресурсу: https://miro.com.
7. Jackson, S. (2016). Mastering Unity 2D Game Development (2nd ed.). Packt Publishing Ltd. ISBN 978-1-78646-345-612.
8. Kelly, C. (2012). Programming 2D Games.CRC Press.ISBN 978146650870534.
9. Geig, M. (2018). Unity 2018 Game Development in 24 Hours, Sams Teach Yourself. Pearson Education. ISBN 978-0-13-499891-656.
10. Lanzinger, F. (2020). 2D Game Development with Unity (1st ed.). CRC Press. ISBN 97804293286647.
11. Halpern, J. (2019). Developing 2D Games with Unity: Independent Game Programming with C#. Apress.

12. Game Development and Production" by Erik Bethke: Бетке, Е. (2003). Розробка та виробництво ігор. Wordware Publishing, Inc. ISBN: 978-1556229510.

13. "The Art of Game Design: A Book of Lenses" by Jesse Schell: Шелл, Дж. (2008). Мистецтво дизайну ігор: Книга лінз. CRC Press. ISBN: 978-0123694966.

14. Microsoft teams [Електронний ресурс] – Режим доступу до ресурсу: https://products.office.com/en-us/microsoft-teams/group-chat-software.

15. Google drive [Електронний ресурс] – Режим доступу до ресурсу: https://www.google.com/drive/.

16. Trello [Електронний ресурс] – Режим доступу до ресурсу: https://trello.com/.

17. Online Whiteboards in 2019 [Електронний ресурс] – Режим доступу до ресурсу: https://zapier.com/blog/best-online-whiteboard/.

18. AWW [Електронний ресурс] – Режим доступу до ресурсу: https://awwapp.com/.

19. Mural [Електронний ресурс] – Режим доступу до ресурсу: https://mural.co/.