МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Обслуговування комп'ютерних систем і мереж»

Група: 4КС-57

Дипломний проект

здобувача освіти денної форми навчання
КС.57.01.000.ДП

АЛЕКСЄЄНКА
ДМИТРА СЕРГІЙОВИЧА

м. Одеса
2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Обслуговування комп'ютерних систем і мереж»**

Група: **4КС-57**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

Розробка 2D-гри в жанрі платформер з використанням фізики програмного рушія Unity

Проектний матеріал складається з пояснювальної записки на 70 сторінках та графічного (презентаційного) матеріалу на 12 аркушах (слайдах)

Дипломник _____ (Алексєєнко Д.С.)

Керівник _____ (Кірєєв І.А.)

Консультанти:

з економічного розділу _____ (Іванченков В.С.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист «19» 06 2024 р. Протокол ЕК № 3

Оцінка ЕК 4 (добре) 75 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ІП*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Обслуговування комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"11" 01 2024 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Алексєєнко Дмитру Сергійовичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка 2D-гри в жанрі платформер з використанням фізики програмного рушія Unity*

затверджена наказом по коледжу від "02" 11 2023 р. № 244-АД

2. Термін здачі закінченого проекту 10.06.2024

3. Вихідні данні до проекту *Використання програмного рушія Unity; Використання мови програмування C# та бібліотек Unity для розробки гри; Використання принципів модульності у проектуванні та розробці ігрових проектів; Реалізація основних ігрових механік комп'ютерної гри в жанрі платформер; Гра повинна мати мінімальний інтерфейс; Гравець має мати змогу проходити рівень, використовуючи фізику програмного рушія Unity.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Аналіз ігрового процесу ігор в жанрі платформер; Аналітичний огляд програмних рушіїв з умовно безкоштовним доступом; Вибір інструментів розробки; Формування концепту ігрового процесу гри; Проектування основних елементів ігрового процесу та принципи їх роботи; Реалізація основних елементів ігрового процесу; Тестування працездатності ігрових елементів.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Особливості ігрового жанру платформер; Особливості програмного рушія Unity та причини його вибору для розробки проекту; Особливості ігрових механік розроблюємого проекту; Огляд основних елементів ігрового процесу; Принцип роботи основних елементів ігрового процесу; Етапи реалізації основних елементів ігрового процесу; Хід тестування гри. Скріншоти гри.

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Кірєєв І.А.*		
Економічний розділ	*Іванченков В.С.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 15.01.24.

Керівник *Кірєєв І.А.* _____
(підпис)

Завдання прийняв до виконання *Алексєєнко Д.С.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка виконан
1	*Вступ. Постановка мети та задач проектування*	*20.05.2024*	
2	*Аналіз існуючих рішень*	*23.05.2024*	
3	*Аналітичний огляд та вибір програмного рушія*	*25.05.2024*	
4	*Формування концепту ігрового процесу гри*	*28.05.2024*	
5	*Проектування основних елементів ігрового процесу*	*30.05.2024*	
6	*Проектування графічної частини гри*	*01.06.2024*	
7	*Реалізація графічної частини гри*	*03.06.2024*	
8	*Реалізація основних елементів ігрового процесу*	*05.06.2024*	
9	*Імплементація та відлагодження ігровіх елементів*	*07.06.2024*	
10	*Тестування працездатності елементів гри*	*09.06.2024*	
11	*Виправлення виявлених помилок*	*10.06.2024*	
12	*Аналіз результатів, підготовка слайдів презентації*	*11.06.2024*	
13	*Економічні розрахунки та питання з охорони праці*	*12.06.2024*	
14	*Підготовка графічної частини проекту*	*13.06.2024*	
15	*Підготовка проекту до захисту та тестування ПП*	*14.06.2024*	

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КС 57. 01 000. 00 ДП	Дипломний проект		
А4			КС 57. 01 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КС 57. 01 000. 00 ДП		
Розробив		Алексєєнко Д.С.		10.06	Розробка 2D-гри в жанрі платформер з використанням фізики програмного рушія Unity	Літ.	Аркуш 4 / Аркушів 70
Перевірив		Кірєєв І.А.		10.06		НДП	
Н. контр.		Петрашова В.І.		10.06		ВСП "ОТФК ОНТУ" гр.4КС-57	
Затверд.		Кривченко Ю.В.		10.06			

ЗМІСТ

ВСТУП

У сьогоденні індустрія комп'ютерних ігор займає велику частину IT-сектору. В ній приймають участь такі спеціалісти як програмісти, аніматори, керівники IT-проектів, інженери програмного забезпечення та інші. Кількість ігрових проектів, що виходять, з кожним роком збільшується, і з такою ж динамікою розвиваються технології створення ігрових проектів, методи проектування та менеджменту. Все це є складовою великих проектів, але комп'ютерна ігрова індустрія складається не лише з них, а ще й з невеликих проектів, які створюються невеличкими колективами, або зовсім однією людиною. Такі ігри завжди залишаються актуальними для споживача.

Сама ігрова індустрія знаходиться лише на початку свого розвитку, починаючи свій шлях з малих та обмежених проектів, поетапно ігри стають все більшими та складними. Спочатку створені для інженерних пристроїв, ігри тепер розробляють для комп'ютерів, ігрових приставок, мобільних телефонів та спеціалізованих пристроїв, а способи розповсюдження майже відійшли від фізичних носіїв до доставки по інтернету.

Зважаючи на те, що ігрова індустрія сьогодні є важливою складовою в IT, важливим є оволодіння основними принципами розробки ігрових проектів. Ці принципи формують весь процес розробки гри від початку, до релізного білду. Темою мого дипломного проекту є «Розробка 2D-гри в жанрі платформер з використанням фізики програмного рушія Unity» тому, через створення даного проекту можна відточити ці принципи.

Окрім того, як основний елемент ігрового процесу в даній грі буде використано фізику програмного рушія Unity. Це дасть змогу більш тісно познайомитись із принципами його роботи та використати для створення дещо іншого ігрового процесу, ніж в інших платформерах.

Оскільки основний задум гри та її жанр не є унікальними, це дає змогу вивчити саме розповсюджені проблеми розробки ігрових проектів, способи їх вирішення. Висока технологічність проекту зумовлена актуальними інструментами розробки, які будуть використані у роботі.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз ігрового процесу ігор в жанрі платформер

1.1.1 Огляд існуючих аналогів в ігровій індустрії

Початок розробки будь-якої гри починається з формування концепт-документу. В свою чергу, для вдалого та вірного формування концепт-документу, необхідно володіти знаннями щодо ігрової сфери, в якій розроблюється проект. Оскільки темою мого дипломного проекту є «Розробка 2D-гри в жанрі платформер з використанням фізики програмного рушія Unity», мені необхідно створити гру у жанрі платформер з використанням фізики. Для початку необхідно встановити, що таке ігровий жанр платформер.

Ігровий жанр платформер, також відомий як платформна гра, це вид відеоігор, в якому ігровий процес в основному пов’язаний зі стрибками по платформах і обходом перешкод.

Однією з перших платформерів вважається гра Donkey Kong, випущена компанією Nintendo в 1981 році. У ній гравець керував персонажем на ім’я Джампмен (пізніше відомим як Маріо), метою якого було врятувати дівчину від гігантської мавпи. 1980-і та 1990-і роки стали «золотим століттям2 для платформерів з виходом таких знакових ігор, як Super Mario Bros, Sonic the Hedgehog, Mega Man та Prince of Persia. На рисунку 1.1 можна побачити скріншот класичного платформеру Prince of Persia:

Рисунок 1.1. Скріншот класичного платформеру Prince of Persia

Сучасні платформери часто включають елементи головоломок, рольових ігор та інших жанрів, пропонуючи складніші та різноманітніші ігрові механіки. Інді-розробники також зробили значний внесок у жанр, створюючи унікальні та інноваційні платформери, такі як Limbo, Braid та Celeste. На рисунку 1.2 можна побачити скріншот сучасного платформеру Celeste:

Рисунок 1.2. Скріншот сучасного платформеру Celeste

1.1.2 Основні риси ігрового жанру платформер

Як у будь-якого ігрового жанру, платформери мають визначені риси та характеристики, ось основні характеристики цього жанру:

- Управління персонажем: Гравець управляє персонажем, який може бігати, стрибати, іноді лазити сходами чи мотузками та виконувати інші акробатичні дії;
- Платформи: рівні містять безліч різноманітних платформ різних розмірів і форм, на які персонаж повинен застрибувати або з яких зістрибувати.
- Перешкоди і вороги: На шляху персонажа зустрічаються різні перешкоди, такі як ями, шипи, платформи, що рухаються, а також вороги, яких потрібно уникати або на яких можна стрибати для їх знищення;
- Збір предметів: Часто в іграх цього жанру є збір предметів, таких як монети, ключі, бонуси, які можуть покращувати характеристики персонажа або відкривати нові рівні.

Також, як будь-який інший ігровий жанр, платформери мають свої технічні

аспекти, які наведені нижче:

- Фізика: Важливою частиною платформерів є реалістичне чи стилізоване моделювання фізики, особливо гравітації та інерції, що впливає на рух та стрибки персонажа.
- Графіка і арт-стиль: Жанр може варіюватися від простої піксельної графіки до складної тривимірної візуалізації, і часто має яскравий арт-стиль, що запам’ятовується.

Платформери продовжують бути популярними завдяки своїй простоті, веселому та захоплюючому геймплею, а також можливості творчого підходу до дизайну рівнів та світу гри. Цей жанр вплинув на розвиток відеоігор і продовжує надихати розробників на створення нових захоплюючих проектів.

Отже можна зробити висновок про основні риси платформерів. По-перше, це ігровий процес, який центрований на переміщені по платформах, стрибках та вирішені просторових, або ситуаційних головоломок. Платформери можуть мати ворогів, або не мати їх зовсім, в залежності від того, яку історію розповідає гра. Визначившись із жанровою складовою проекту та її особливістю, необхідно перейти до технічної сторони розробки.

1.2 Аналітичний огляд програмних рушіїв з умовно безкоштовним доступом

1.2.1 Аналітичний огляд ігрового програмного рушія Unity

Розробка ігрового проекту також основана на виборі програмного забезпечення для вирішення питання технічної реалізації та супроводу. Вибір ігрового програмного рушія є дуже важливою частиною початку розробки, оскільки наперед визначає як процес розробки та ресурси розробки, так і підтримку проекту. Дуже часто ігрові програмні рушії диктують розробникам те, як буде працювати їх проект і чи буде він працювати зовсім. Наприклад, одною із причин, чому вважають провальним гру Anthem – це її технічна сторона, коли для створення рольової гри із елементами кастомізації та менеджменту, відкритого світу та глибокої мультиплеєрної взаємодії, використовувався внутрішній ігровий

програмний рушій компанії Electronic Arts, що початково створювався як ігровий програмний рушій для ігор жанру шутер. Через це розробникам доводилось витрачати багато часу та зусиль для того, щоби створювати окремий інструментарій для розробки рольових ігор.

Ігрові програмні рушії можна розділити на безкоштовні, умовно безкоштовні та платні. До останніх найчастіше відносяться ігрові програмні рушії створені всередині крупних ігрових компаній для внутрішніх проектів, найчастіше, ці компанії не діляться своїми рішеннями з конкурентами, тільки у випадках партнерства, або фінансових відносин. Безкоштовні ігрові програмні рушії найчастіше розповсюджуються у вигляді бібліотек із мінімалістичним інструментарієм для розробки, що викликає досить серйозні проблеми для розробки. Ідеальними у відношенні ціна-якість є ігрові програмні рушії, що розповсюджуються на умовно безплатній основі. Ці програмні рушії мають величезний інструментарій для розробки, а також не потребують плати за використання, при виконанні окремих умов.

Тема мого дипломного проекту «Розробка 2D-гри в жанрі платформер з використанням фізики програмного рушія Unity». Не зважаючи на те, що в темі прописано обраний програмний рушій, я вважаю важливим проілюструвати, чому мій вибір зупинився саме на Unity.

Unity – це популярний багатоплатформний ігровий програмний рушій та середовище розробки, створений компанією Unity Technologies. Він надає розробникам інструменти для створення 2D та 3D ігор, додатків віртуальної та доповненої реальності, симуляцій, візуалізацій та багато іншого. Unity відрізняється простотою використання та доступністю, що робить його привабливим вибором для новачків у розробці ігор, а також для досвідчених розробників. Він надає зручний графічний інтерфейс, безліч готових ресурсів та компонентів, а також потужну мову програмування C# для створення інтерактивності та логіки ігор. На рисунку 1.3 можна побачити приклад роботи у ігровому програмному рушії Unity.

Рисунок 1.3. Приклад роботи у ігровому програмному рушії Unity

Однією з ключових особливостей Unity є багатоплатформність. Він підтримує різні платформи, включаючи ПК, консолі, мобільні пристрої, віртуальну реальність та доповнену реальність, що дозволяє розробникам досягати більшої аудиторії та випускати свої продукти на різних пристроях.

З точки зору ліцензування, то ігровий програмний рушій надає повний доступ до свого функціоналу безкоштовно, для створення ігор будь-якого рівня. Необхідність покупати ліцензію з’являється лише якщо сума зароблених грошей на користувача дійде до рівня у двісті тисяч доларів.

1.2.2 Аналітичний огляд ігрового програмного рушія Unreal Engine

Для того, щоби дати розуміння причини обрання ігрового рушія Unity, розглянемо його двох найближчих конкурентів. Почнемо з Unreal Engine. Це потужний і широко використовуваний ігровий програмний рушій, розроблений компанією Epic Games. Він надає розробникам інструменти для створення високоякісних ігор та візуалізацій, а також розробки віртуальної реальності (VR), доповненої реальності (AR) та багатьох інших інтерактивних проектів. Unreal Engine пропонує широкий набір функцій, включаючи потужний рушій графіки, підтримку фізичного рушія, інструменти для роботи зі штучним інтелектом, анімацією персонажів, звуком та багатьом іншим. Він також включає графічний інтерфейс користувача, який спрощує процес розробки та створення контенту. На

рисунку 1.4 можна побачити приклад роботи із анімацією у ігровому рушії Unreal Engine.

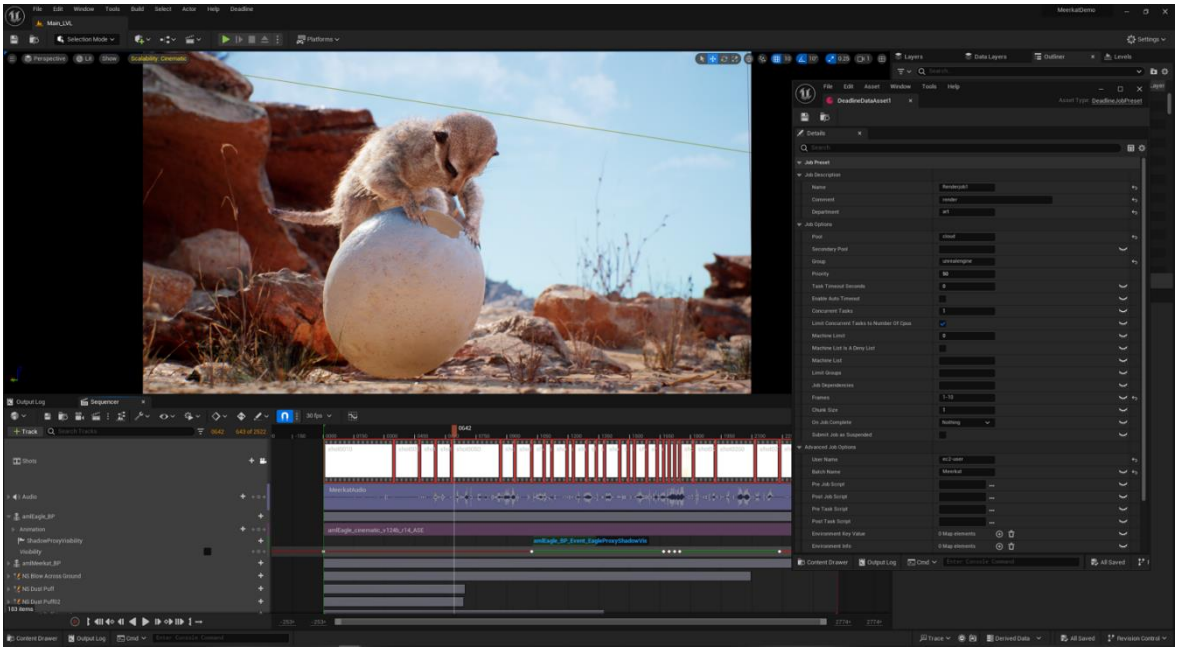


Рисунок 1.4. Приклад роботи із анімацією у ігровому рушії Unreal Engine

Однією з ключових особливостей Unreal Engine є його графічний рушій, який підтримує передові технології, такі як трасування променів (ray tracing), що дозволяє створювати неймовірно реалістичні графічні ефекти. Це робить Unreal Engine популярним вибором для розробників ААА-ігор, а також для створення вражаючих візуалізацій та симуляцій в інших галузях. На рисунку 1.5 можна побачити приклад графіки яку відтворює Unreal Engine.

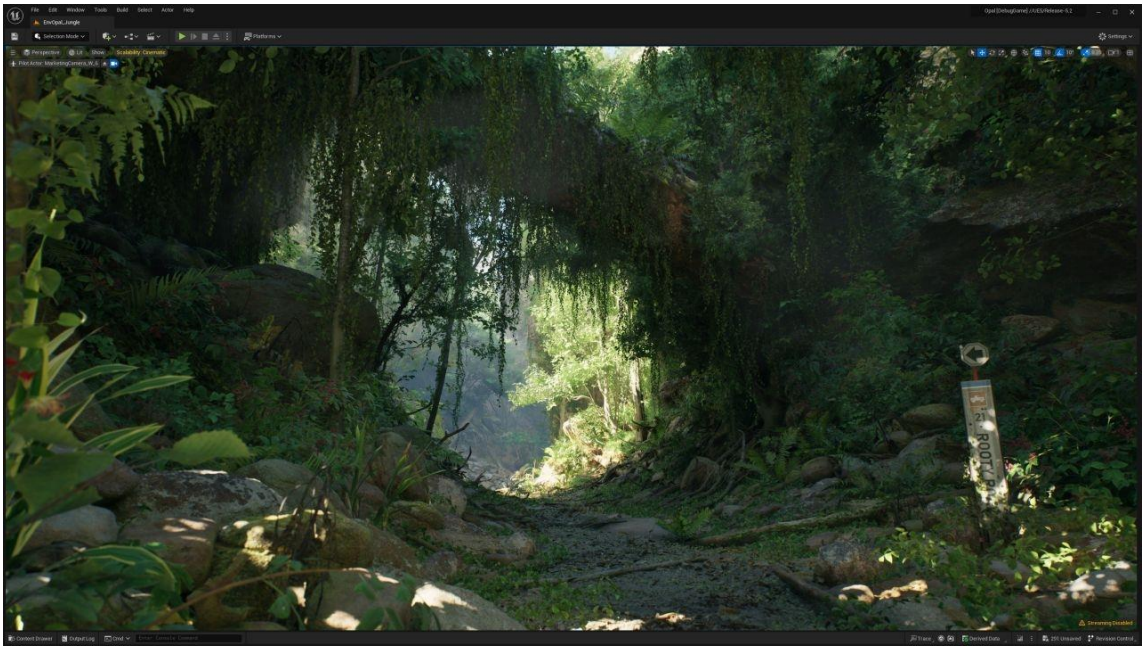


Рисунок 1.5. Приклад графіки яку відтворює Unreal Engine

З точки зору ліцензування, то Unreal Engine є безкоштовним, проте не дає доступу до коду самого ігрового програмного рушія, деякі платформи розробки та можливості заблоковані, а також закритий доступ до виділеного середовища розробників та підтримки.

1.2.3 Аналітичний огляд ігрового програмного рушія CryEngine

Ще одним конкурентом ігрового програмного рушія Unity є CryEngine – це потужний ігровий програмний рушій, розроблений німецькою компанією Crytek. Crytek – це німецька компанія, заснована братами Джеваном, Авні та Фарук Йерлі у 1999 році. Сам ігровий програмний рушій відомий своїми вражаючими графікою та фізикою, а також широкими можливостями для створення ігрових світів. CryEngine був використаний у різних популярних іграх, таких як серія Crysis, Ryse: Son of Rome та Hunt: Showdown. Початково отримав відомість як ігровий програмний рушій, що відтворює неймовірно реалістичну графіку, а також правдоподібну симуляцію фізики із симуляцією руйнування довкілля. Через свої неймовірні візуальні та фізичні показники отримав відомість, як бєнчмарк для комп'ютерів того часу. На рисунку 1.6 можна побачити приклад графіки, що відтворює програмний рушій CryEngine.

Рисунок 1.6. Приклад графіки, що відтворює програмний рушій CryEngine

З точки зору можливостей, то однією з ключових особливостей CryEngine є його підтримка технології трасування променів (ray tracing), яка дозволяє досягти

фотореалістичних ефектів освітлення та відображення. Крім того, CryEngine має високий рівень масштабованості, що дозволяє створювати як невеликі інді-проекти, так і великі ААА-ігри. Програмний рушій також забезпечує розробників інструментами для створення інтерактивного та живого ігрового середовища, включаючи систему штучного інтелекту, керування анімацією персонажів, а також інструменти для створення звукової атмосфери. CryEngine також пропонує гнучку систему багатоплатформної розробки, що дозволяє випускати ігри на різних платформах, включаючи ПК, консолі та мобільні пристрої. На рисунку 1.7 можна побачити приклад роботи з вбудованим аніматором рушія CryEngine.

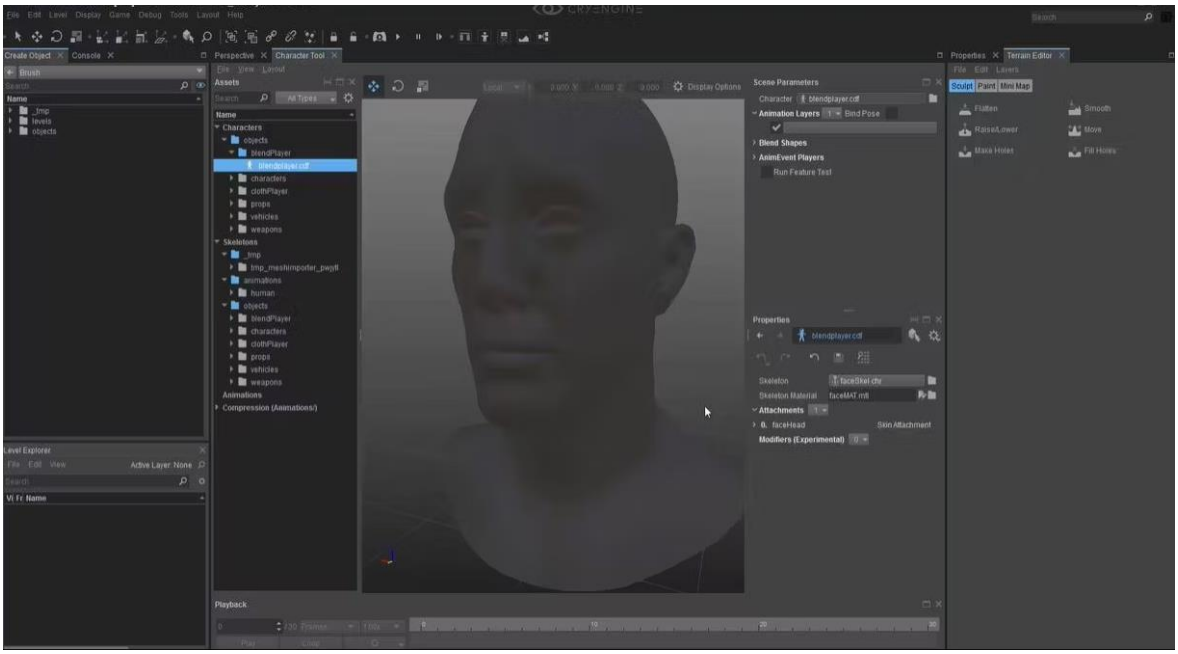


Рисунок 1.7. Приклад роботи з вбудованим аніматором рушія CryEngine

Загалом ігровий програмний рушій CryEngine є безкоштовним, втім система побудована таким чином, що використовувати індивідуальну підтримку, бібліотеку ассетів, можна лише з покупкою ліцензії. Окрім того, її все одно необхідно буде купувати, якщо кількість фінансового обороту користувача програмного рушія перевищить визначеного рівня.

1.2.4 Принцип обрання ігрового програмного рушія Unity

Проаналізувавши ігрові програмні рушії було вирішено використовувати рушій Unity. Рішення було зумовлено в першу чергу доцільністю використання можливостей ігрових рушіїв. CryEngine та Unreal Engine створені у передусім для відтворення дуже красивих з графічної точки зору проектів, які створюються з

прицілом на серйозну технічну базу. В той же час Unity створювався для невеликих, простих та легких проектів, що відповідає грі дипломного проектування. Окрім того умови ліцензування дуже прості та зрозумілі для користувача. Документація та спілка розробників початково доступна, присутній доступ до магазину ассетів. Через вказані причини, вибір програмного рушія був зроблений на користь Unity.

1.3 Вибір інструментів розробки

Наступним важливим кроком у підготовці до розробки ігрового проекту є обрання інструментарію. Програмне забезпечення для розробки має дуже велику роль у процесі роботи, адже кожне рішення має свої особливості та що не менш важливо, користувач може мати свої улюблені редактори, в яких має навички роботи та знає всі нюанси. Більшість крупних компаній розробників досить вільно відносяться до прикладного інструментарію розробки, наприклад, як редактори коду.

Редактор коду – це спеціалізоване програмне забезпечення, яке використовується для створення, модифікації та організації вихідного коду програм. Такі редактори зазвичай мають інтуїтивно зрозумілий інтерфейс, підтримують підсвічування синтаксису, автоматичне доповнення коду, пошук та заміну тексту, а також інтеграцію з іншими інструментами розробки, включаючи системи контролю версій, дебагери та компілятори. Серед відомих редакторів коду можна виділити Visual Studio Code та Sublime Text.

У процесі вибору з численних редакторів коду, я віддав перевагу Microsoft Visual Studio. Цей вибір був зумовлений тим, що ігровий движок Unity працює на мові програмування C#, що вимагає від середовища розробки не лише підтримки форматування коду, але й здатності взаємодіяти з іншими класами, методами, ігровими об'єктами та бібліотеками. Крім того, Microsoft Visual Studio дозволяє легко доступитися до проекту гри і є рекомендованим вибором серед розробників та підтримки Unity.

Що стосується роботи з графікою, то потрібен графічний редактор. Графічний редактор – це програмне забезпечення, призначене для створення,

модифікації та оптимізації графічних зображень. Він надає користувачам інструменти для створення ілюстрацій, редагування фотографій та інших видів графіки, включаючи кисті, олівці, спреї, штампи, інструменти виділення та багато іншого, а також можливості для коригування кольору, розміру, прозорості та інших параметрів зображення. Серед популярних графічних редакторів можна назвати Adobe Photoshop, GIMP, CorelDRAW та Adobe Illustrator.

У контексті розробки цієї гри я вирішив використовувати Adobe Photoshop, оскільки у мене вже є досвід роботи з цим інструментом. На даному етапі розробки гра не вимагає створення складної графіки, а лише потребує розробки кількох спрайтів та редагування зображень, включаючи зміну пропорцій та розмірів.

Важливою частиною процесу розробки є також система контролю версій. Система контролю версій – це інструмент, який дозволяє розробникам відслідковувати зміни у вихідному коді та інших файлах проекту, порівнювати версії, відкочувати до попередніх станів та об'єднувати роботу різних учасників. Вона також забезпечує історію змін, що сприяє кращому розумінню процесу розробки та співпраці над проектом. До відомих систем контролю версій належать Git, Subversion та Mercurial.

У моєму випадку я використовую BitBucket та SourceTree. BitBucket – це веб-базована система контролю версій, яка є безкоштовною та інтегрована з багатьма іншими сервісами в рамках екосистеми Atlassian. SourceTree – це клієнтська програма, яка забезпечує графічний інтерфейс для роботи з системами контролю версій, що робить процес більш зручним і інтуїтивно зрозумілим. Ці інструменти є незамінними для збереження результатів розробки в онлайн-середовищі.

1.4 Формування концепту ігрового процесу гри

Для подальшої розробки ігрового проекту, згідно теми дипломного проектування, потрібно виконати формування концепту ігрового процесу гри. Це один із важливіших кроків у розробці будь-якого проекту, оскільки він включає в себе попереднє планування, що у подальшому дуже полегшує розробку. Концепт-

документ гри – це документ, який містить основні ідеї, концепції, механіки геймплею, сюжетні елементи та інші ключові аспекти, пов’язані з розробкою гри. Цей документ зазвичай є основою для її подальшого розвитку. У концепт-документі гри частіше за все розглядають такі питання:

- Опис гри: Загальний опис ігрового світу, сюжету, атмосфери та основної мети гри;
- Механіка геймплею: Опис ігрових механік, механіки управління, фізики ігрового світу, системи прогресії, системи бою та інших ігрових аспектів;
- Сюжет та персонажі: Опис сюжету гри, персонажів, їх характеристик, мотивацій та відносин між ними;
- Графіка та аудіо: Концепції та ідеї з візуального стилю, арт-дизайну, анімації та звукового оформлення гри;
- Світогляд і фон: Додаткові деталі про світ гри, його історію, культуру, технології та інші аспекти, які можуть впливати на ігровий досвід.
- Технічні вимоги та можливості: Обговорення технічних аспектів розробки гри, включаючи платформи, програмні рушії, інструменти та інше.

Концепт-документ гри допомагає команді розробників чітко визначити напрямок розробки, уникнути непорозуміння та уточнити ключові аспекти гри ще до початку фактичної роботи над проектом. Завдяки цьому команди, які використовують принцип формування концепту гри частіше дотримуються плану розробки та не допускають розповсюдженої помилки, коли розробники починають імплементувати у гру контент, якого в ній не повинно було бути зовсім, через що розробка затягується, або зовсім закінчується та починається спочатку.

Тема мого дипломного проекту «Розробка 2D-гри в жанрі платформер з використанням фізики програмного рушія Unity», отже мені потрібно розробити платформер. Вже було проведено аналітичну частину розробки, окрім того визначено з ігровим програмним рушієм, а також із інструментами розробки. Виходячи з того, що було визначено і обрано, можна сформувати концепт ігрового

процесу.

Оскільки розробляється платформер до основи гри мають війти платформи по котрим гравець буде переміщуватись. У своїй розробці я базувався на враженнях від двох ігор минулого, як Marble Madness – рисунок 1.8 – та Gravity Defied яка зображена на рисунку 1.9:

Рисунок 1.8. Скріншот гри Marble Madness

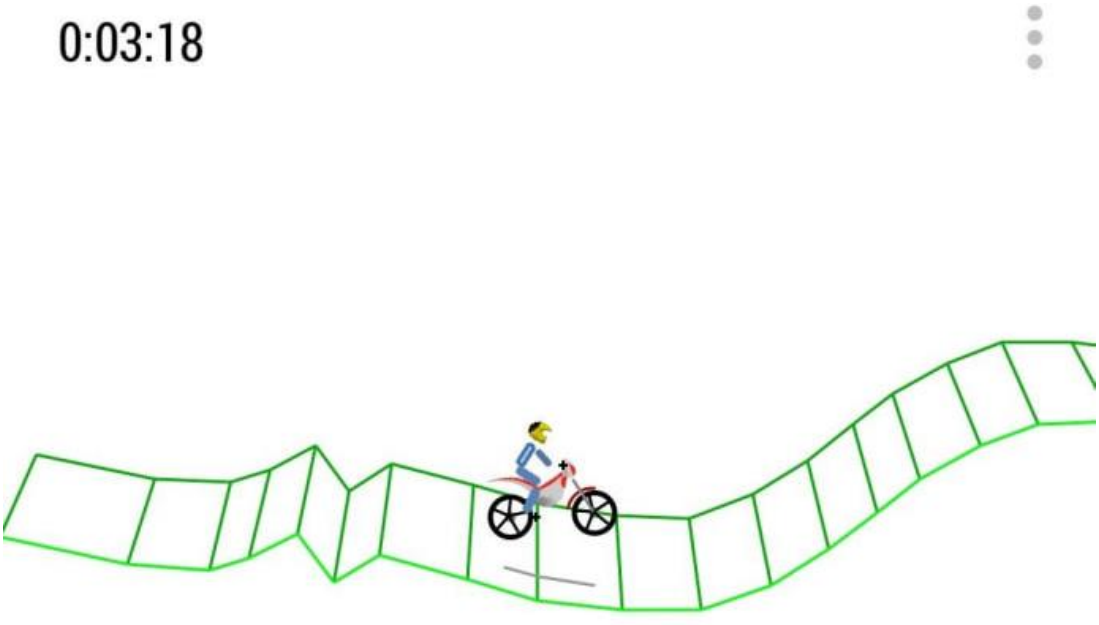


Рисунок 1.9. Скріншот гри Gravity Defied

З першої гри я взяв ідею використання сфери, як аватару гравця. Це дасть змогу більше відчувати плив сил фізики на ігровий процес, адже сфера майже ніколи не зможе знаходитись у стані спокою, тому її потрібно буде весь час контролювати. З другої гри було взято використання положення камери, а також мінімалістичний підхід до візуальної складової, використання складних положень рельєфу рівнів. Комбінація таких ідей згенерувала концепт ігрового процесу зображеного на рисунку 1.10:

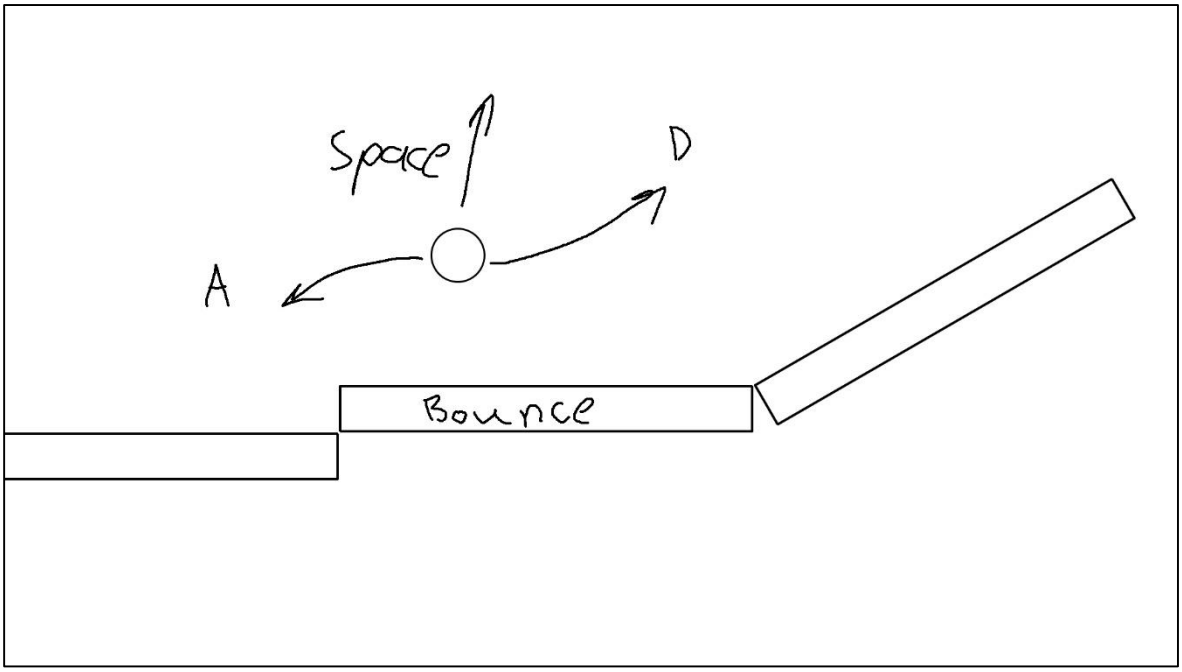


Рисунок 1.10. Концептуальне зображення ігрового процесу проекту

Як можна бачити з рисунку, поверхні мають властивості стрибучості, що також буде додавати складності до ігрового процесу. Гравець буде мати у своєму розпорядженні можливість рухатись вліво та вправо, а також виконувати стрибки за допомогою клавіши пробілу на клавіатурі. Розглянемо цю концепцію детальніше на рисунку 1.11.

З рисунку можна побачити, що гравець не буде мати змогу виконувати безкінечні стрибки. Гравець, або сфера, буде мати кількість енергії, яку можливо буде використовувати для виконання стрибків. Кожний стрибок буд віднімати 20 одиниць енергії. В той же час рух вліво-вправо не буде потребувати ресурсів сфери.

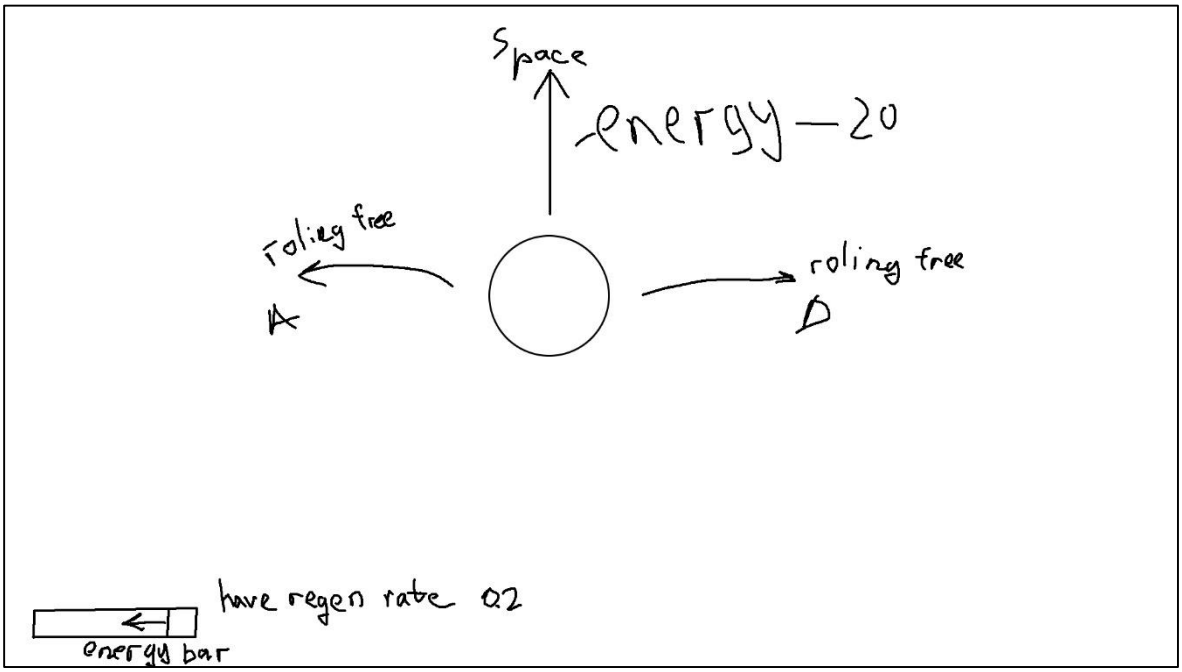


Рисунок 1.11. Детальне зображення концепту ігрового процесу

Також потрібно уточнити концепцію взаємодії з платформами. Концепцію їх використання я зобразив на рисунку 1.12. Як можна побачити із рисунку, можна виконати модифікацію ігрового процесу за допомогою внесення змін до властивостей платформ, на яких зможе переміщуватись сфера гравця. Наприклад на даному етапі розробки гри можна передбачити декілька видів платформ, які зображені на рисунку 1.12. Вони дадуть змогу не тільки ускладнювати гру, але й будувати ситуації фізичних головоломок.

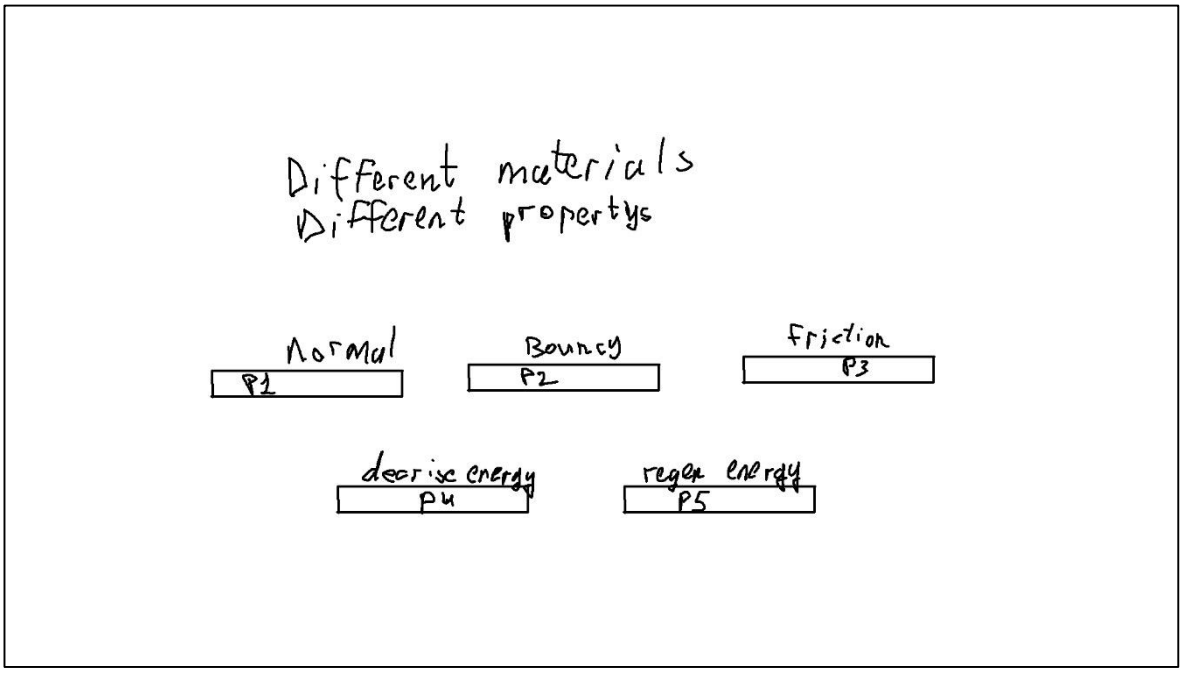


Рисунок 1.12. Деталізація концепту використання платформ у грі

З рисунку можна побачити декілька видів платформ, які будуть відрізнятись зовнішнім виглядом – кольором. Властивості можуть бути різними. Наприклад, одна з платформ має звичайні властивості поверхні, а інша збільшену стрибучість. Також можна виділити дві платформи, які впливають на кількість енергії сфери. Так одна платформа зменшує енергію, а інша навпаки збільшує коефіцієнт її оновлення.

За допомогою створеного концептуального матеріалу можна розпочати проектування основних елементів ігрового процесу.

1.5 Проектування основних елементів ігрового процесу та принципи їх роботи

Для подальшої розробки проекту також важливою частиною є проектування його основних складових, елементів ігрового процесу та взагалі описання процесу роботи цих складових. Це важливо з тієї точки зору, що раніше спроектовані елементи набагато легше розробляти, оскільки під час процесу проектування більшість логічних та структурних проблем ігрових або логістичних механік.

Для початку проектування необхідно продумати загальну структуру проекту та його складових. Це є важливою частиною, адже дає змогу відразу визначити те, що потрібно реалізувати у проекті, структуру каталогів, зв’язок елементів гри та принципи їх роботи. На рисунку 1.13 можна побачити загальну структурну схему проекту.

Як можна побачити з рисунку, основу проекту складає блок ігрового процесу, а ігровий інтерфейс та фізична складова значно менші. Так виходить с того, що на даному етапі розробки перед проектом ставиться завдання створити ігровий процес – базу гри. На її основі можна у подальшому побудовувати інші складові гри, такі як глибокий інтерфейс та меню. Потрібно відмітити і складові ігрового процесу: він складається з блоку властивостей сфери, властивостей платформ та взаємодії з оточенням. Можна сказати, що гра базується на трьох стовпах сфері, платформ та оточення, що досить точно ілюструє основні складові жанру платформер.

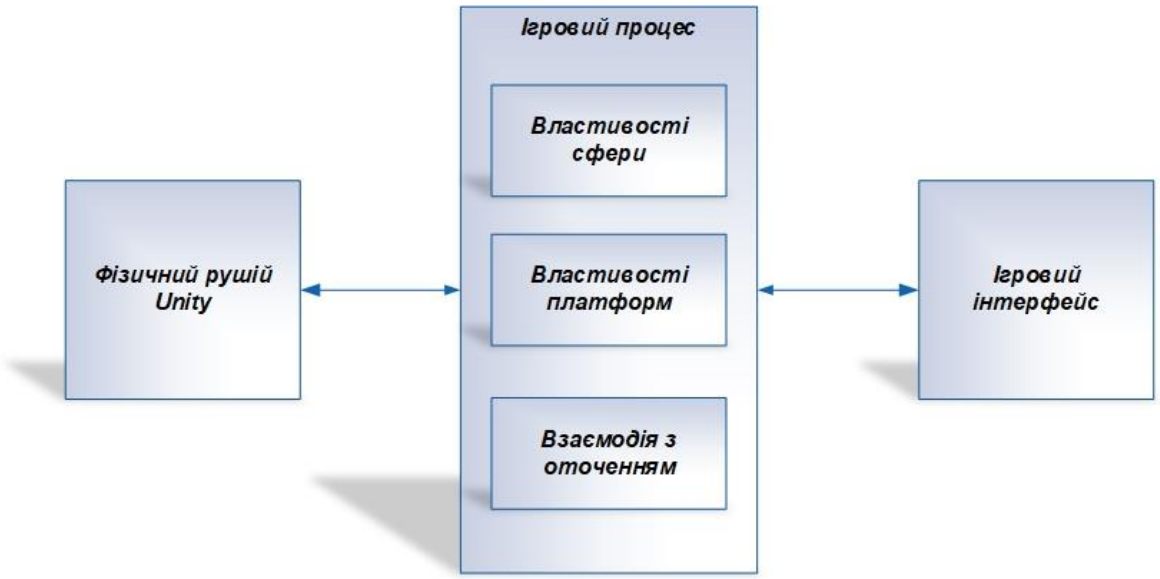


Рисунок 1.13. Загальна структурна схема проекту

На наступній схемі, що зображена на рисунку 1.14, зображено більш масштабовану структуру складових проекту. Можна побачити, що властивості сфери поєднують у собі такі елементи, як управління сферою та параметри гравця. Це зроблено для уточнення потреб зі сторони концепції ігрового процесу, який потребує присутності параметру енергії гравця. Можна було б розмістити параметри гравця у блоці управління, проте це буде не вірно з точки зору ООП та подальшого розвитку проекту. Тому потрібно заздалегідь створити дві сутності, які потім можна розширювати новим функціоналом.

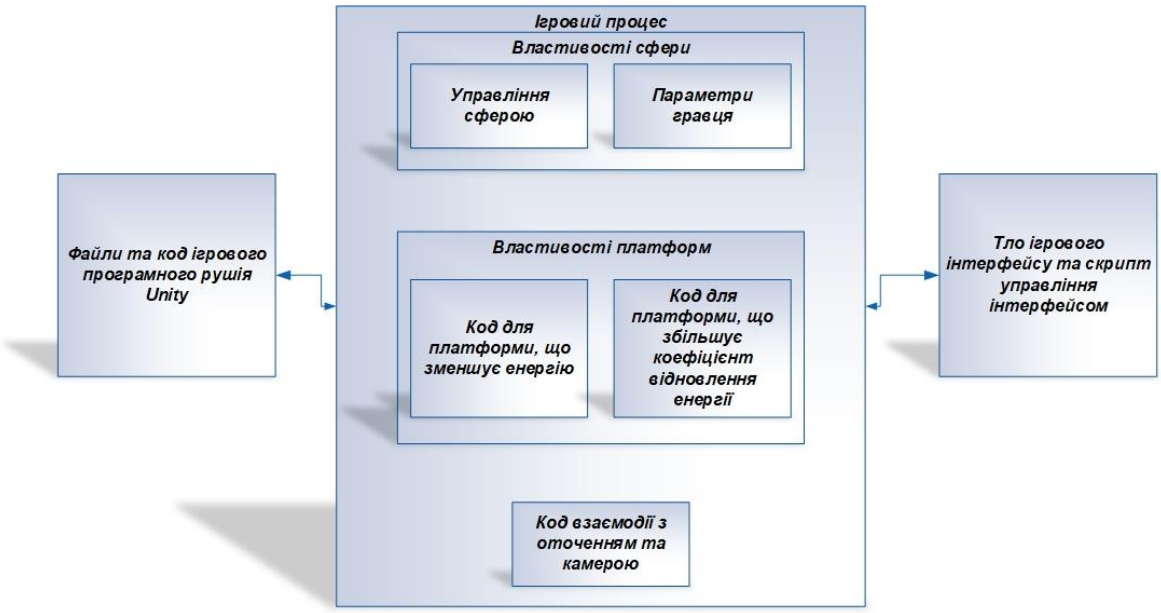


Рисунок 1.14. Масштабована структурна схема проекту

Також, можна побачити, що було враховано потребу реалізувати механіки

взаємодії платформ зі сферою гравця. Тому на схемі відображено необхідність створити двох модулів коду для платформ що зменшують енергію гравця та таких, що її збільшують. Також на схемі можна побачити, що код для взаємодії з оточенням та камерою винесено в окремий блок. Таке рішення є спірним, втім процес безпосередньо реалізації покаже, яким чином буде правильніше реалізувати це.

Останнє про що потрібно зазначити, це про структуру платформ, оскільки в темі дипломного проекту зазначено використання фізики ігрового програмного рушія Unity, то необхідно більш детально розглянути питання фізики в Unity.

Фізична складова ігрового рушія Unity, відома як Unity Physics, є ключовим елементом для створення реалістичної інтерактивності в іграх. Вона базується на фізичному движку PhysX від компанії NVIDIA, який забезпечує симуляцію різноманітних фізичних явищ, таких як гравітація, зіткнення, тертя, та еластичність. Нижче приведено деякі ключові аспекти фізичної складової Unity:

- Rigidbody: Компонент, який додається до об’єктів, щоб вони могли реагувати на сили та зіткнення відповідно до законів фізики. Rigidbody може бути динамічним (піддається силам і зіткненням), кінематичним (рухається за заданими параметрами, але не реагує на сили) або статичним (не рухається);
- Colliders: Компоненти, які визначають форму об’єкта для системи фізики та використовуються для виявлення зіткнень. В Unity є різні типи колайдерів, включаючи бокс, сферу, капсулу, меш та інші;
- Physics Materials: Матеріали, які визначають властивості тертя та відскоку при зіткненні об’єктів;
- Joints: Компоненти, які дозволяють об'єктам з’єднуватися один з одним різними способами, імітуючи фізичні з’єднання, такі як шарніри, пружини, та інші.

Physics Engine Settings: Налаштування, які дозволяють контролювати загальну поведінку фізичної системи, включаючи гравітацію, час симуляції, та інші параметри.

Unity також надає інструменти для оптимізації фізичних розрахунків, такі як Layer Collision Matrix, яка дозволяє визначити, які шари (layers) можуть взаємодіяти між собою, та Physics Raycaster, який використовується для виявлення об’єктів у сцені за допомогою променів.

Фізична складова Unity є потужним інструментом для розробників, який дозволяє створювати ігри з реалістичною фізикою без необхідності глибокого розуміння складних фізичних розрахунків. Це робить Unity популярним вибором для розробки ігор різних жанрів. Тому в ході проектування та реалізації я буду використовувати можливості фізики цього програмного рушія. На рисунку 1.15 зображено приклад будови ігрового об’єкту платформи та компонентів, що взаємодіють із фізикою Unity:

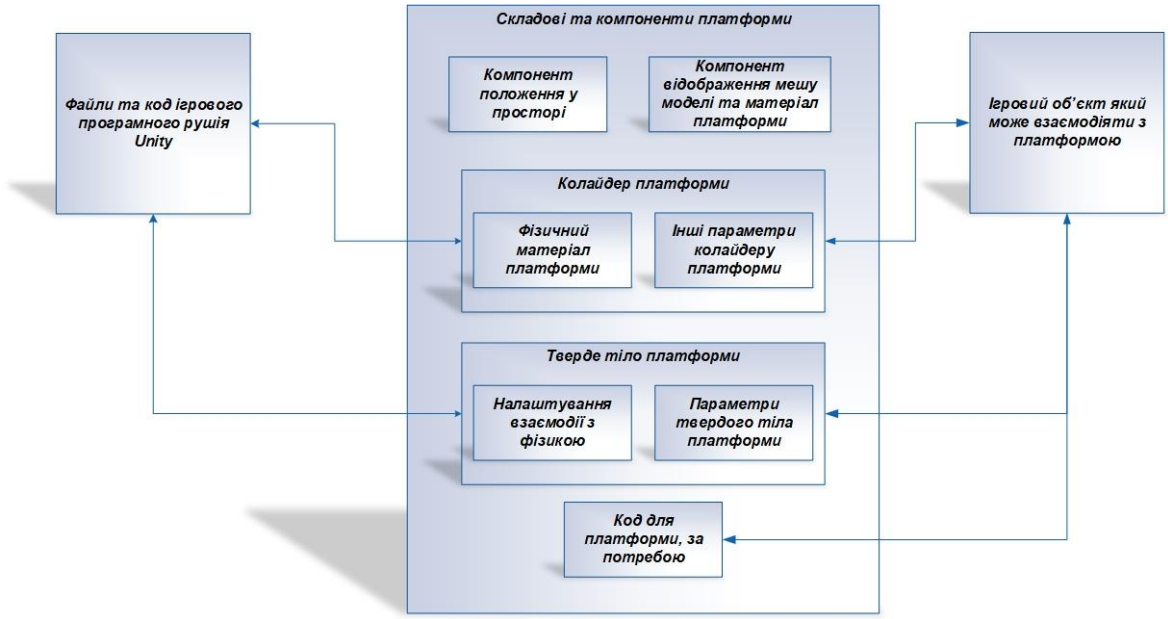


Рисунок 1.15. Приклад будови ігрового об’єкту платформи та компонентів, що взаємодіють із фізикою Unity

Як можна побачити то із фізикою ігрового програмного рушія Unity взаємодіють такі компоненти як колайдер та тверде тіло. Якщо точніше, то виділені складові цих компонентів. Деякі з цих складових можна налаштовувати, а інші працюють із «чорного ящику».

З проектуванням основних ігрових елементів було закінчено. Отримано структурні схеми, які у майбутньому допоможуть реалізувати всі необхідні складові гри.

1.6 Реалізація основних елементів ігрового процесу

Слід розпочати реалізацію всіх елементів гри з його графічної частини. Це не тільки практично, але й правильно з тої точки зору, що можна використати створені ассети під час розробки та відразу бачити результат.

Оскільки в даному проекті дуже мінімалістична графічна складова, на даному етапі створювати зображення у великій кількості немає потреби. Тому всі спрайти були створені за допомогою Adobe Photoshop та розміщені у папці проекту.

Втім, у грі є й інша графічна складова. Проект заявлений як 2D-гра в жанрі платформер з використанням фізики програмного рушія Unity. Це зовсім не означає, що проект має бути створений за допомогою спрайтів та анімацій. Досить часто 2D-ігри створюють з використанням 3D-графіки, а свою двовимірність ці ігри зберігають через ігровий процес, або розміщення камери. В моєму проекті буде використано фіксовану камеру, яка буде існувати у трьохвимірному просторі.

Оскільки гра використовує 3D-об’єкти, до них можна застосовувати два види редагування. Перший – це створення текстури із UV-розгортки моделі, але це потребує вмінь у роботі з графікою та деякий час. Другий варіант – використовувати матеріали. Матеріали використовуються в тому разі, коли ви хочете взяти модель, але надати їй візуальних властивостей інших матеріалів, наприклад, металів, дерева або скла. Кожний такий матеріал можна додати до ігрового об’єкту, який має компонент MeshRenderer, що автоматично приєднає матеріал до всієї моделі.

Отже, для початку, потрібно звернутись до концепту та визначити, скільки нам потрібно створити матеріалів, та яким чином вони будуть виглядати. Згідно з концепцією, мені необхідно створити дві платформи із активними властивостями, та три пасивні платформи. Кожна з них буде мати різні фізичні властивості. В свою чергу, основуючись від правил геймдизайну, ігрові елементи та об’єкти, які виконують яку небудь роль у ігровому процесі мають бути якимось чином позначені. Окрім того, бажано, щоби позначення давало гравцю підсвідому

інформацію про те, яку роль виконує цей об’єкт. Тому, кожна така платформа має отримати свій матеріал, який відповідає своїй ролі. На рисунку 1.16 можна побачити типове налаштування матеріалу для платформи:

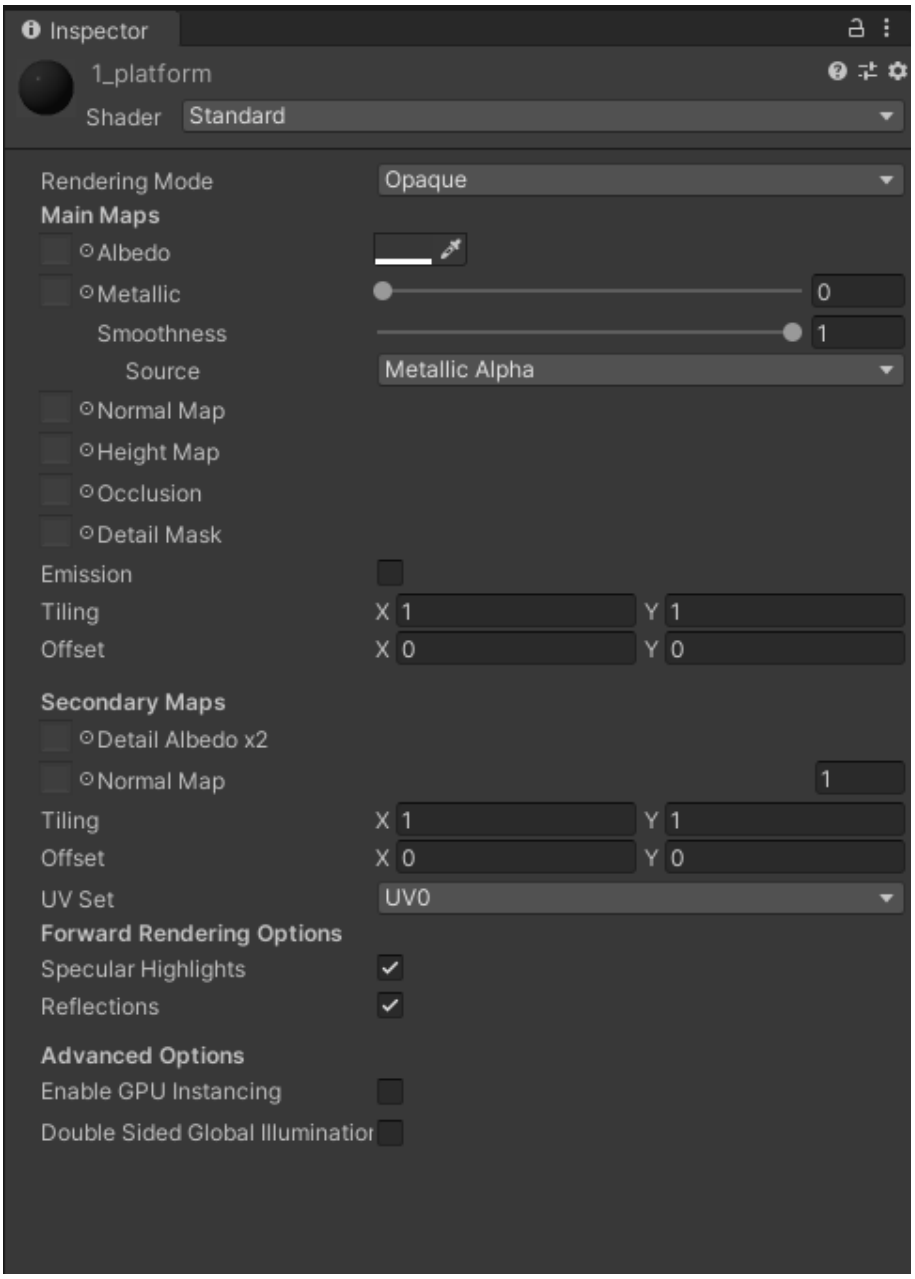


Рисунок 1.16. Приклад типового налаштування матеріалу для платформи

Як можна побачити з рисунку 1.16, у ігровому рушії є досить широкі можливості з редагування властивостей матеріалів, починаючи від кольору та прозорості, закінчуючи обранням шейдерів. У випадку із матеріалами, які використовуються в проекті, то їм виставляється значення «металічності» 0, а «плавності» 1. Завдяки цьому, платформи не блищать, але відражають тіні та світло, створюючи досить приємне зображення. Таким самим чином створюються

матеріали для всіх платформ. На рисунку 1.17 показано всі створені матеріали для платформ у грі:

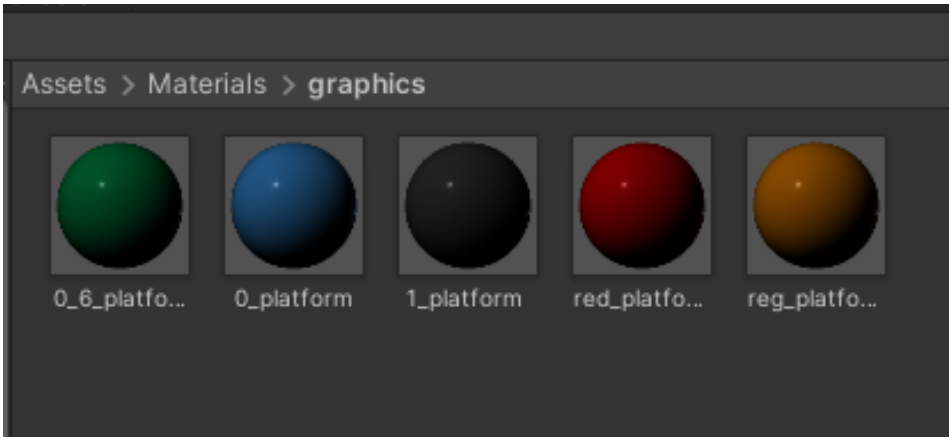


Рисунок 1.17. Перегляд всіх матеріалів для платформ у грі

Наступним кроком у підготовці до створення платформ є реалізація «фізичних матеріалів». Це окремий вид матеріалів, які передають до фізичного рушія Unity додаткову інформацію про те, яким чином він має симулювати взаємодію об’єктів. Такі матеріали створюються таким же чином, як і матеріали для візуально зміни мешей моделей. Щодо кількості таких моделей, принцип зберігається, оскільки кожна платформа повинна мати свою фізичну можель взаємодії зі сферою гравця. На рисунку 1.18 зображено приклад типового налаштування фізичного матеріалу для всіх платформ гри:

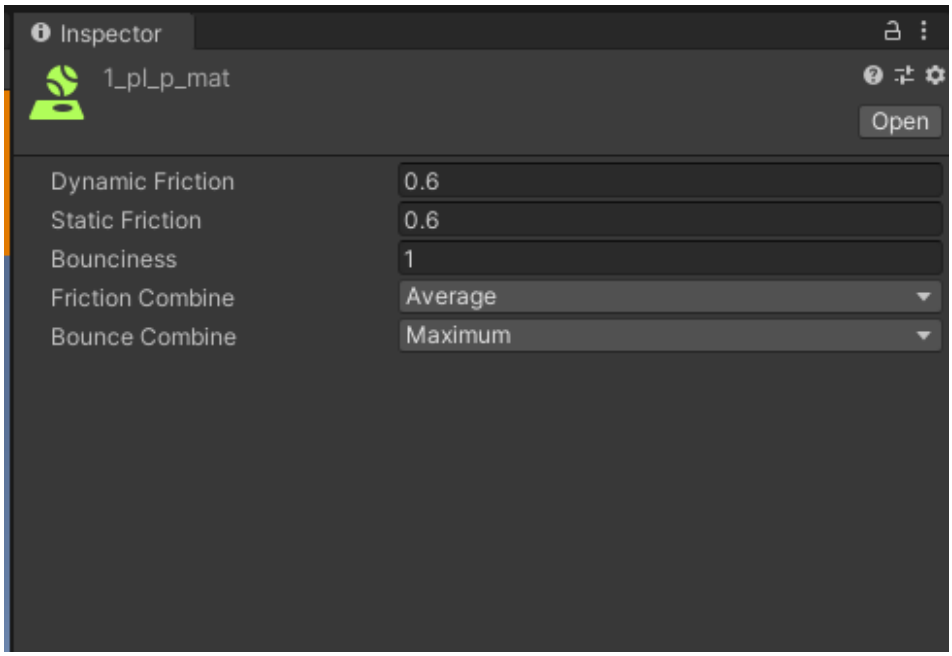


Рисунок 1.18. Приклад типового налаштування фізичного матеріалу для всіх платформ гри

На рисунку 1.18 зображено створення фізичного матеріалу для платформи із низькими показниками тертя та максимальним показником стрибучості. Як можна побачити з рисунку, у параметрі Bounce Combine обрано режим Maximum для того, щоби сфера відлітала максимально далеко, відповідно до тої сили, що вона прикладе при падінні. На рисунку 1.19 зображено перелік всіх фізичних матеріалів для всіх платформ гри:

Рисунок 1.19. Перелік всіх фізичних матеріалів для всіх платформ гри

Наступним кроком є безпосередньо створення самих платформ. Оскільки вони створюються для багаторазового використання, немає сенсу кожну платформу створювати окремо за кожною навою необхідністю. Набагато правильніше буде створити так звані «prefabs» або префаби. Це такі ігрові об’єкти, які ніби переміщають з ієрархії ігрових об’єктів до переліку ассетів. Після цього такі префаби можна використовувати кожний раз, створюючи копію цього префаба. Найсильнішою особливістю цього методу є той факт, що кожна створена копіяпрефабу сама по собі унікальний об’єкт тому її можна змінювати так, як потрібно. При цьому внесення змін у сам префаб, якій знаходиться в ассетах, автоматично змінює всі його копії. Таким чином, у разі якщо посеред розробки виявилась помилка, або з’явилась потреба ввести нову ігрову механіку, то це можна зробити лише один раз із самим префабом, а зміни автоматично перенесуться до його клонів.

Вже визначено, що для реалізації концепту гри, потрібно створити 5 платформ, кожна з котрих матиме свій матеріал, свій фізичний матеріал, а також префаб. На рисунку 1.20 можна побачити приклад виконання налаштувань типового префабу платформи для гри:

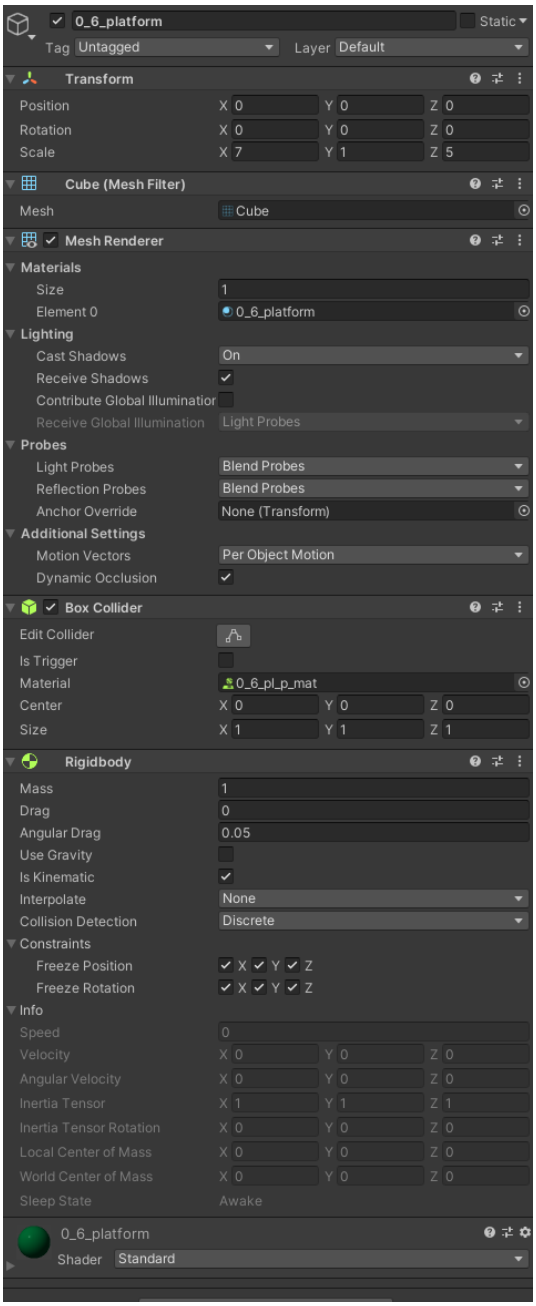


Рисунок 1.20. Приклад виконання налаштувань типового префабу платформи для гри

Як можна побачити із рисунку 1.20, дана платформа має свої параметри налаштування, а також складається з тих компонентів, які ми розглядали під час проектування відповідного ігрового об’єкту. Ми можемо бачити, що до ігрового об’єкту платформи вже додано створені графічний матеріал та фізичний матеріал, які надають ігровому об’єкту кубу його особливості. Також, можна побачити, що були змінені параметри масштабування – Scale – ігрового об’єкту платформи. За замовченням параметр масштабування у всіх об’єктів має значення 1, 1, 1, що відповідає прямому використанню параметрів мешу ігрової моделі. В даному

випадку можна побачити, що були змінені значення масштабування по сторонам X та Z, тобто куб, з якого створюється платформа був розтягнутий по ширині да довжині. На рисунку 1.21 можна побачити зелену платформу із невеликою стрибучістю та тертям:

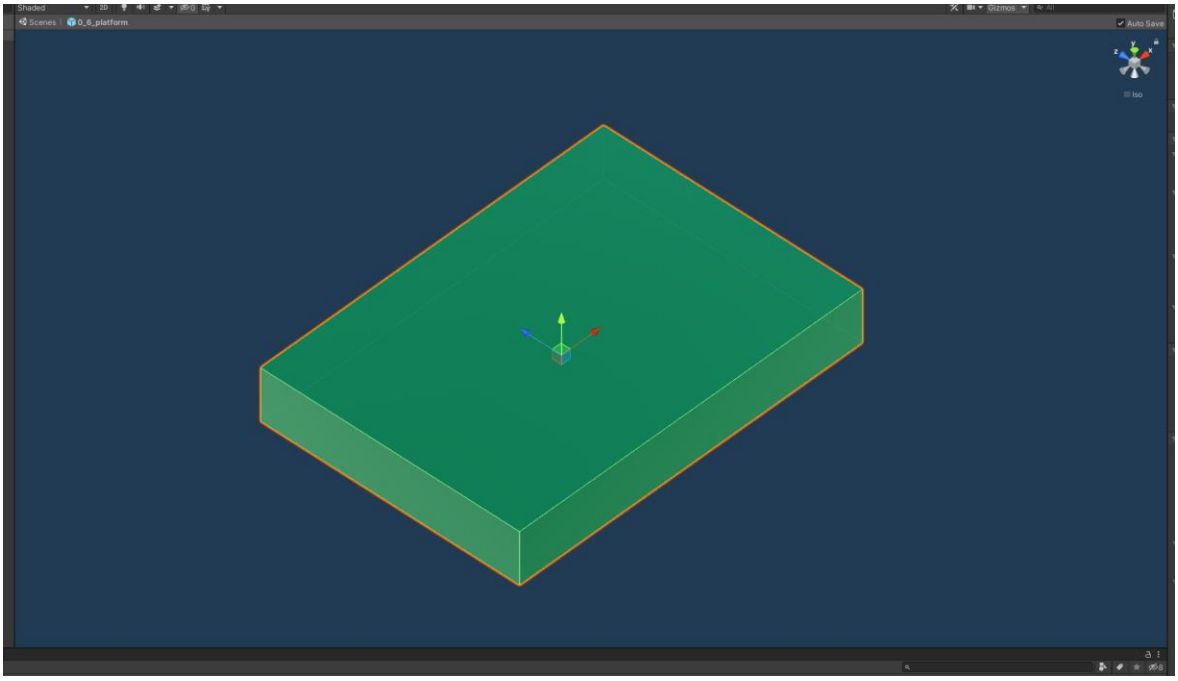


Рисунок 1.21. Зелена платформа із невеликою стрибучістю та тертям

Для створення префабу потрібно спочатку створити ігровий об’єкт, виставити в ньому всі потрібні компоненті, після чого перетягнути ігровий об’єкт до зони ассетів гри, як показано на рисунку 1.22.

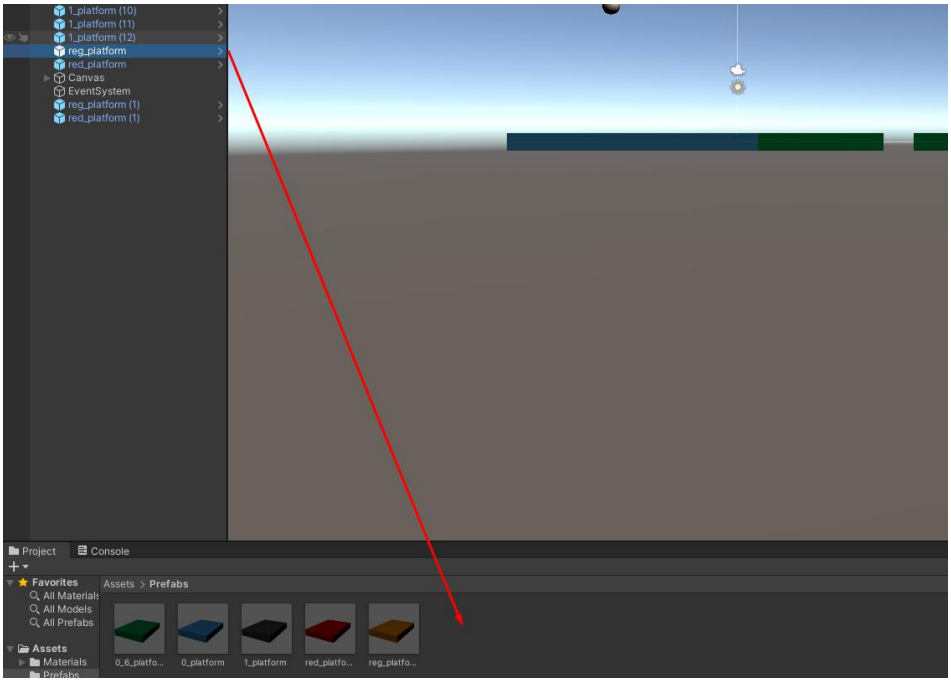


Рисунок 1.22. Процес створення префабу платформи для гри

Після цієї маніпуляції ігровий об’єкт займе своє місце серед ассетів гри. Насправді, буде створений файл з даними щодо того, які використовувати ресурси із якими властивостями для майбутнього відтворення.

Після створення графічних та фізичних матеріалів, а також ігрових платформ, потрібно створити ігровий об’єкт гравця. На рисунку 1.23 зображено компоненти такого об’єкту:

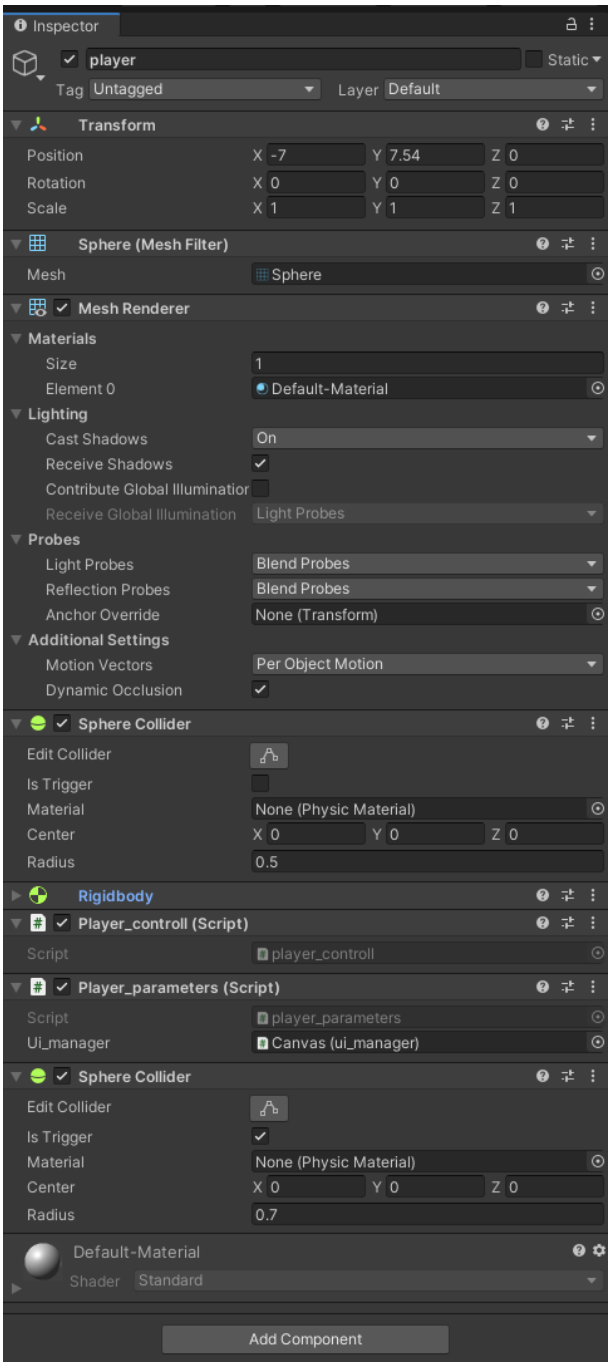


Рисунок 1.23. Компоненти ігрового об’єкту гравця

Як можна бачити з рисунку 1.23, гравець реалізується за допомогою примітиву сфери, яка створюється у панелі ієрархії об’єктів. З деталей можна

виділити той факт, що у об’єкту гравця є два колайдери. Один працює в звичайному режимі, а другий використовується, як тригер. До причин такого використання колайдеру, а також способу його роботи повернемося пізніше. Також слід зазначити, що у об’єкта, в компоненті RigidBody вимкнено повороти об’єкту по осям X та Y. Це зроблено для того, щоби об’єкт не отримував неконтрольованого руху по платформам, а міг рухатись лише вліво, або вправо. На рисунку 1.24 можна побачити зовнішній вигляд ігрового об’єкту гравця:

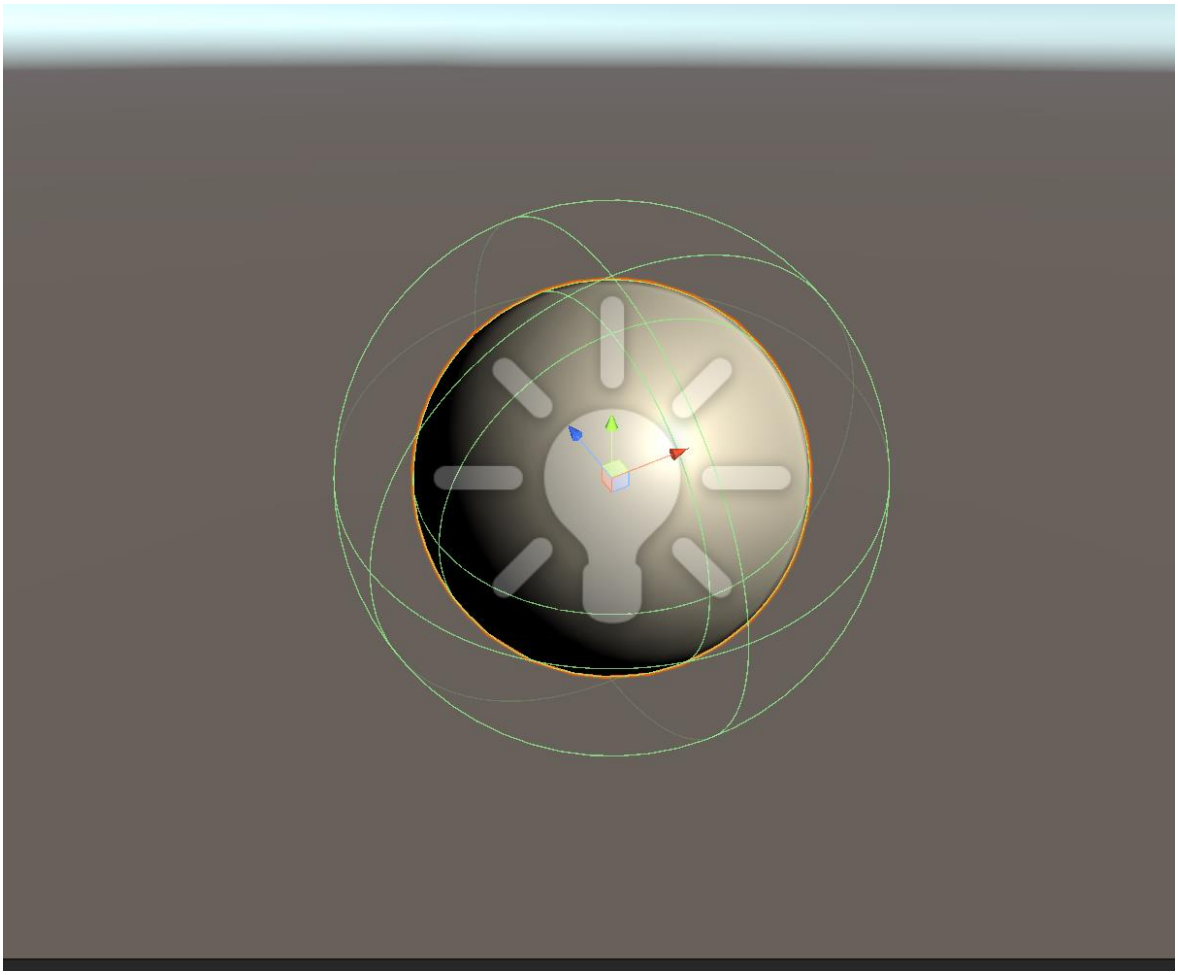


Рисунок 1.24. Зовнішній вигляд ігрового об’єкту гравця

Можна побачити, що всередині об’єкту відображено лампочку – це об’єкт об’ємного освітлення, який приєднуються до ігрового об’єкту гравця та створює ефекти освітлення в тих місцях, де глобальне світло може не потрапляти.

Наступним кроком буде створення безпосередньо самого рівня. Використовуючи префаби платформ, їх можна перетягувати з панелі ассетів та розміщувати у ігровому просторі. Потрібно пам’ятати, що проект будується у 3D-просторі, тому необхідно слідкувати за тим, де розміщується ігровий об’єкт

платформи. Вони мають бути розміщені в одну лінію по Z-координаті, оскільки саме вона відповідає за «глибину» кадру. Як вже було зазначено у розділі розробки концепції гри, камера буде відображати частину платформи. Тому всі платформи мають бути розміщені в одну лінію по глибині. На рисунку 1.25 зображено процес створення рівня та його частину:

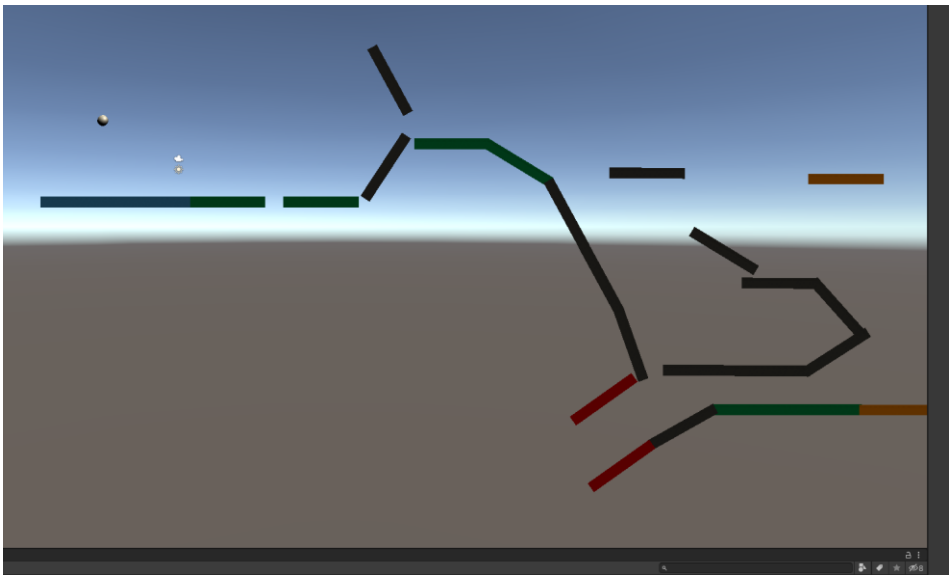


Рисунок 1.25. Процес створення рівня та його частина

Редактор Unity дозволяє вільно використовувати камеру редактору. Тому можна по різному виставляти її положення, щоби оцінювати розміщення ігрових об’єктів на сцені. На рисунку 1.26 можна побачити частину рівня з іншого ракурсу:

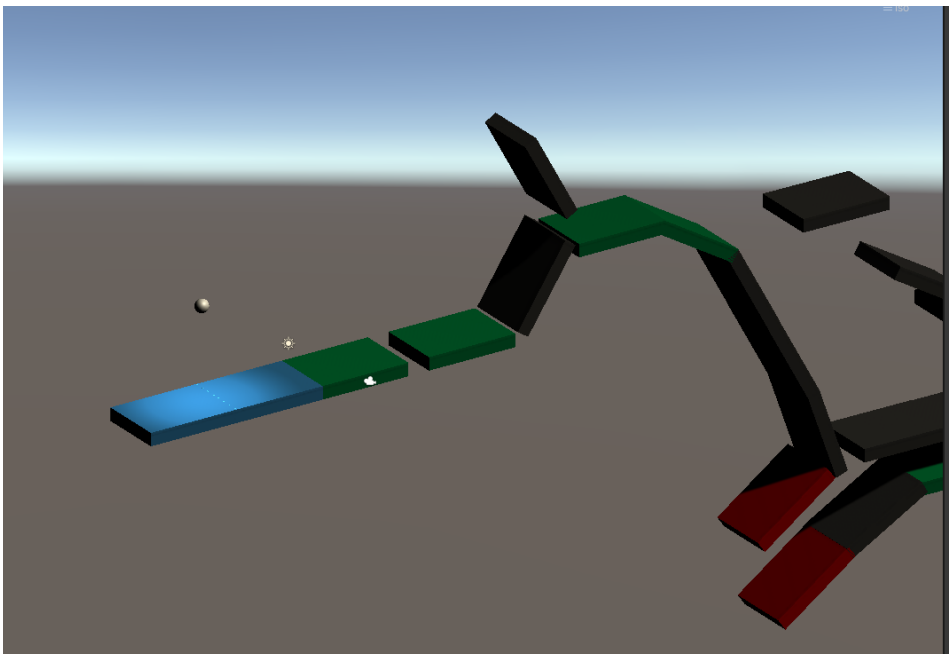


Рисунок 1.26. Частина рівня з іншого ракурсу

Коли всі ігрові об’єкти розміщені та налаштовані, потрібно присупити до реалізації коду гри. Спочатку необхідно закріпити камеру за гравцем, для того, щоби вона давала гравцю добрий огляд ігрового процесу. Код у ігровому програмному рушії Unity реалізується за допомогою скриптів, які працюють лишу у зв’язці із ігровими об’єктами. Скрипти, по суті, є те, що у програмуванні ми могли б назвати класами. Вони мають у собі деякий код, які якимось чином впливає на ігровий процес. На рисунку 1.27 зображено шаблон нового скрипту у ігровому програмному рушії Unity:

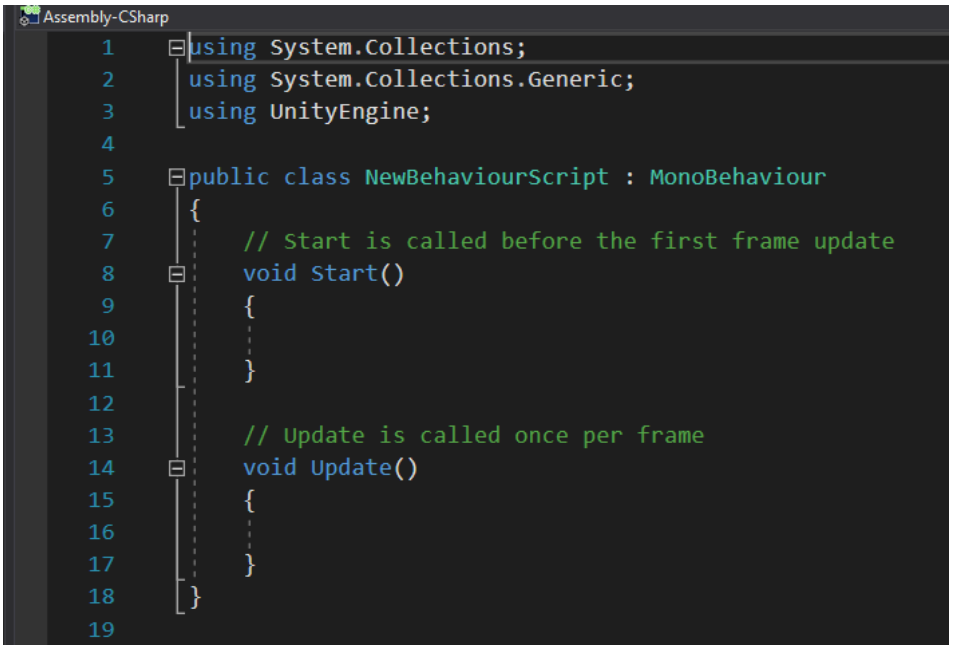


Рисунок 1.27. Шаблон нового скрипту у ігровому програмному рушії Unity

Кожний ігровий скрипт дає у розпорядження розробника два методи за замовченням: Start() та Update(). Перший виконує свій зміст лише в момент, коли скрипт починає свій цикл життя, тобто тоді, коли його ігровий об’єкт, до якого він доданий, з’являється на сцені. Другий метод виконує свій зміст кожний можливий кадр, який може зобразити ігрова платформа, на якій працює гра. Тобто, якщо комп’ютер відмальовує 60000 кадрів, то зміст цього методу буде виконано 60000 разів. Не завжди ці методи мають бути використані, тому їх можна як залишити, так і видалити. У нашому випадку коду для камери, ми видаляємо метод Start(), оскільки його зазвичай використовують для ініціалізації яких-небудь даних.

Для того, щоби камера слідкувала за гравцем потрібно відстежувати положення гравця у просторі. За це відповідає компонент Transform, який є у

кожного ігрового об’єкту. Тому потрібно створити комірку для отримання цього компоненту від ігрового об’єкту гравця. Після чого, в кожному кадрі присвоювати властивості position компоненту transform нові координати але отримувати з позиції гравця лише координати осей X та Y. Z-координата залишається незмінною, оскільки тоді камера буде переміщуватись на позицію гравця. Слід зазначити, що також необхідно дещо змінювати координату Y, та збільшувати її на 4 пункти, для того, щоби камера була трохи вище за рівень висоти гравця. Нижче приведено код цього скрипту:

```
[SerializeField] Transform player_transform;

private void Update()
{
  transform.position = new Vector3(player_transform.position.x, player_transform.position.y+4f, transform.position.z);
}
```

Наступним кроком буде створення системи управління сферою гравця. Для цього потрібно буде створити скрипт управління player_controll.cs. У ігровому програмному рушії Unity є багато способів реалізації переміщення об’єкту гравця по ігровому простору. Наприклад, самім простим може здаватись переміщення об’єкту гравця по координатам X та Y відповідно від того, на які кнопки він натискає. І насправді, можна напряму передавати у компонент transform, у його компонент position нові координати. Наприклад при натисканні кнопки D на клавіатурі, переміщувати ігровий об’єкт на деяку кількість пунктів вправо. Втім, це не буду коректним рухом, а скоріше переміщенням об’єкту в просторі. Фактично, ми будемо телепортувати ігровий об’єкт гравця у просторі кожний кадр. Це зовсім не працює, оскільки в такому випадку він буде залазити в середину інших об’єктів.

Іншим способом управління гравцем є Character Controller. Це спеціальний компонент, який створено для реалізації управління ігровим об’єктом гравця, відразу уникаючи проблеми пов’язані із фізикою ігрового простору. Фактично, цей компонент пересуває ігровий об’єкт у просторі по фізиці, але ігнорує деякі фізичні явища. Окрім того, в такому компоненті автоматизована вираховування

підіймання по сходах, інерція, гравітація та багато чого іншого. Також він дає змогу дуже просто зчитувати натискання кнопок переміщення та запускати рух гравця. Втім, для реалізації моєї ігрової концепції він не підходить, саме через ігнорування цим елементом деякої ігрової фізики, наприклад стрибучості поверхонь. Через це, використовувати Character Controller не має сенсу.

У випадку даного проекту єдиним працюючим варіантом для реалізації руху є використання властивостей твердого тіла – RigidBody – на пряму, без посередників, або обмежень. Єдине, що потрібно зробити для правильної роботи цього компоненту під час ігрового процесу, це заборонити кручіння навколо осей Y та X. Причина тому лежить у використанні для переміщення методу AddForce компоненту RigidBody. Цей метод надає вказану величину сили до твердого тіла у вказаному напрямку. Напрямок задається тьохмірним вектором, а сила змінною з плаваючою комою. Наприклад, якщо потрібно перемістити вправо, то необхідно надати вектор напрямку (1, 0, 0) та силу більшу за нуль. Після цієї команди тверде тіло отримає імпульс сили, який спробує рухати об’єкт у заданому векторі. Якщо тверде тіло має форму куба, то він трохи зсунеться зі свого місця, а сфера почне котитись вправо, оскільки імпульс створить умови для інерціонального руху. Іншими словами, в обох випадках ми ніби будемо штовхати ігровий об’єкт у вказану сторону.

Отже, як було написано вище, ми будемо використовувати у своєму русі ігрової сфери компонент RigidBody. Для цього, його необхідно отримати з ігрового об’єкту гравця, оскільки скрипт не може використовувати компонент напряму, бо не відомо чи є цей компонент у ігрового об’єкту, чи буде використовуватись компонент зовні. Томи створюю змінну-посилання типу Rigidbody, та у методі Start() скрипту переміщення намагаємось отримати компонент. Є декілька способів це зробити, але оскільки ми будемо шукати компонент у тому ж самому ігровому об’єкті, в якому буде знаходитись скрипт, то достатньо просто викликати команду GetComponent<>() і вказати у трикутних дужках Rigidbody. Якщо такий компонент буде у складі ігрового об’єкту, в якому знаходиться цей скрипт, тоді змінна-посилання отримає посилання на ньому, у

іншому випадку, це посилання буде мати значення Null і намагання звернутись до нього призведе до помилки. Нижче приведено фрагмент коду скрипту переміщення:

```
player_parameters _player_parameters;
Rigidbody player_rb;
float player_speed = 10f;

void Start()
{
   _player_parameters = GetComponent<player_parameters>();
   player_rb = GetComponent<Rigidbody>();
}
```

Як можна побачити з фрагменту коду вище, окрім отримання доступу до компоненту Rigidbody, ми таким самим чином створюємо змінну-посилання для скрипту параметрів гравця. Механізм отримання посилання такий же, як і у випадку твердрого тіла. Окрім того, тут створюється змінна швидкості гравця, яка має вплив на те, як швидко сфера гравця буде рухатись у ігровому просторі.

Для подальшої реалізації механізмів управління, нам потрібно реалізувати скрипт параметрів гравця, оскільки для стрибків має використовуватись параметр енергії гравця, а цю величину на етапі проектування було вирішено розмістити в окремому модулі. Отже, створюємо новий скрипт player_parameters.cs. Для реалізації механіки енергії, створюємо змінну типу float та встановлюємо їй значення 200f. Це значення буде початковим максимальним значенням енергії сфери гравця. Для реалізації механіки зменшення енергії створимо метод, код якого приведено нижче:

```
public void reduce_player_energy(float amount)
{
   player_energy -= amount;
   if (player_energy < 0)
      player_energy = 0;
}
```

Як можна побачити з приведеного вище фрагменту коду скрипту параметрів гравця, метод для зменшення кількості показника енергії має параметр типу float, який має на увазі, перед виконанням свого тіла, отримання значення кількості енергії, яку потрібно відняти від поточного показника. В тілі методу, виконується

віднімання, та перевірка значення енергії гравця. Оскільки воно не може бути від'ємним, то в разі його випадкового зменшення нижче 0, значення енергії встановлюється в 0. Додатково до цього, потрібно перевіряти кількість енергії на етапі намагання використати дію стрибка, тому створимо ще один метод, який буде передавати значення енергії гравця, цей фрагмент коду зображено нижче:

```
public float get_player_energy()
{
   return player_energy;
}
```

Тепер, коли всі необхідні методи створені, ми можемо до кінця реалізувати механіку переміщення гравця. Тому повертаюсь до коду player_controll та створюю необхідний код. А саме, нам потрібно отримувати натискання трьох кнопок управління: Space, A, D. Перша кнопка реалізує стрибок. Друга та третя кнопка реалізує рух вліво-вправо, відповідно.

Є деяка особливість у тому, як їх отримувати ти виконувати рух. Річ у тім, що як було написано вище, Update() викликається різну кількість разів, яка залежить від кількості кадрів, що рисує ігрова платформа. Тому, якщо ми будемо переміщувати об'єкт у цьому методі, то він кожний раз буде переміщуватись на різну відстань, з різною швидкістю. Є інший метод FixedUpdate() – він працює так само як і Update(), але викликається завжди фіксовану кількість разів у секунду. Через це, очевидно, що даний метод буде більш валідним для реалізації переміщення гравця. На рисунку 1.28 схематично зображено різницю у роботі цих методів.

Отже, можна зробити вивід, що переміщення буде реалізовано у методі FixedUpdate(), що є невірно. Оскільки плюси цього методу створюють і його мінуси. Він ідеально підходить для отримання постійних величин та їх реалізації, коли ми натискаємо та утримуємо клавіші. Навпаки, коли ми маємо натискати на клавішу всього один раз – будуть створюватись проблеми, оскільки не завжди можна буде «попасти» натисканням клавіші у момент роботи цього методу.

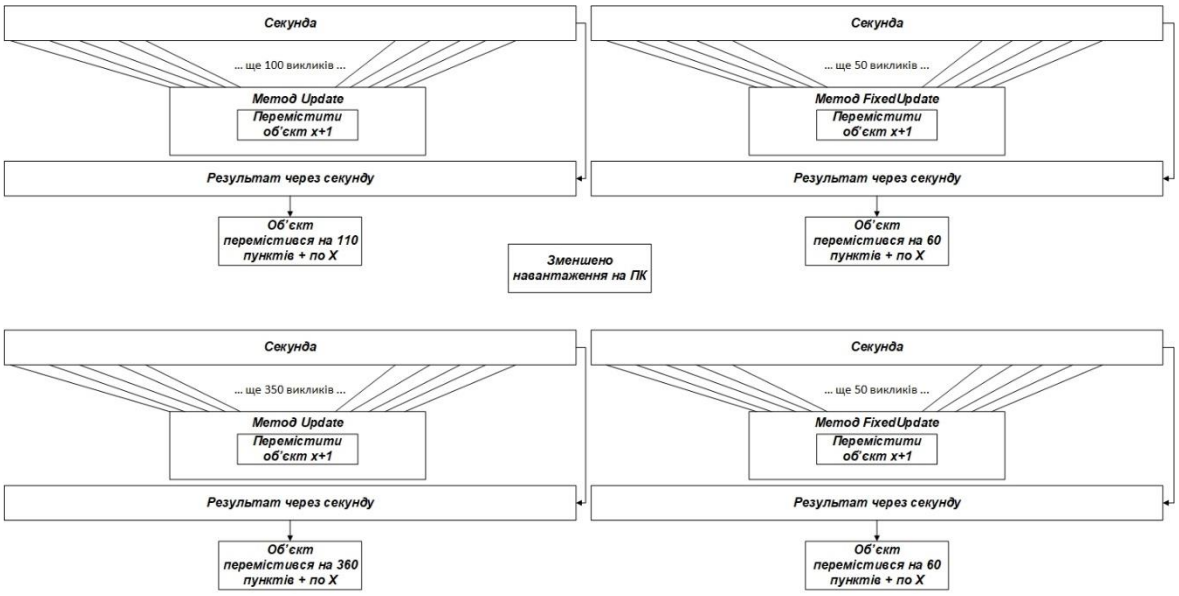


Рисунок 1.28. Різниця у роботі методів Update() та FixedUpdate()

Тому в методі Update() буде реалізовано механізм стрибків, оскільки він буде «слухати» натискання кожний такт виконання гри, а другий метод FixedUpdate(), буде виконувати постійний рух. Фрагмент коду з цими методами зображено нижче:

```
void Update()
{
   if (Input.GetKeyDown(KeyCode.Space))
   {
      if(_player_parameters.get_player_energy() > 20f)
      {
         player_rb.AddForce(Vector3.up * 200f);
         _player_parameters.reduce_player_energy(20f);
      }
   }
}

private void FixedUpdate()
{
   if (Input.GetAxis("Horizontal") > 0f)
   {
      player_rb.AddForce(Vector3.right * player_speed);
   }
   if (Input.GetAxis("Horizontal") < 0f)
   {
      player_rb.AddForce(Vector3.right * -player_speed);
   }
}
```

З фрагменту коду можна побачити, що зчитування натискання кнопок виконується за допомогою такого об’єкту як Input. Він постійно «слухає» статус всіх кнопок управління, які тільки може зчитувати ігровий програмний рушій Unity. І який би не був їх статус, завжди можна звернутись до цього об’єкту та отримати значення натискання на кнопку, тригер, клавішу або стік. Тут же, у коді, можна побачити і різний підхід до зчитування значень. Так, якщо йдеться про стрибки – натискання на кнопку – ми використовуємо умовний оператор if та визначаємо чи натиснута кнопка (клавіша) за допомогою метода GetKeyDown об’єкту Input та передаємо до нього ідентифікатор кнопки, що хочемо натискати. У разі, якщо клавіша натиснута, тоді виконується наступна перевірка, яка звертається до коду параметрів гравця player_parameters.cs та отримує значення енергії сфери гравця. Якщо це значення вище 20 із плаваючою комою, тоді виконується надання сили до твердого тіла ігрового об’єкту сфери гравця, після чого знову звертаємось до скрипту параметрів гравця та зменшуємо значення енергії гравця.

У другій частині коду, яку виконуємо у методі FixedUpdate, ми знову створюємо умовний оператор if який знову використовує об’єкт Input, але використовує вже метод GetAxis(). Цей метод зчитує так звані осі управління, які задаються у файлі параметрів проекту. Кожна з таких осей має своє ім’я, тому щоби вказати яку саме ось ми хочемо зчитати, потрібно вказати її ім’я у форматі змінної строкового типу string. Цей метод завжди буде повертати значення, не зважаючи на те, чи була натиснута кнопка, чи ні, відрізняється лише значення, яке цей метод поверне. Якщо гравець не використовує ось до якої буде виконано звернення, то метод поверне значення «0». Окрім того, обидві клавіші A та D знаходяться на одні осі “Horizontal”, тому ми можемо отримувати їх значення з одної осі.

Виходячи з написаного вище, зрозуміло, що при натисканні на кнопку A будуть отримані від’ємні значення осі, а при натисканні на D – позитивні. У разі покою клавіші буде отримано нуль. Таким чином, зрозуміти, яку кнопку управління натискає гравець – дуже просто. Якщо значення від’ємне, то A, а якщо

позитивне – D. З цього і виходить, що в одному випадку ми будемо передавати у метод AddForce твердого тіла позитивне значення швидкості гравця, а в інше – негативне. Таким чином сфера буде рухатись або вліво, або вправо, відповідно до натискання на клавіші A і D.

Реалізувавши код управління та параметрів гравця, потрібно відображати на інтерфейсі гравця зміну кількості енергії. Для цього створюю скрипт ui_manager.cs. Разом з тим, у редакторі ігрового рушія Unity я створюю площину інтерфейсу користувача, та розміщую на ньому елемент слайдеру Slider. Це дуже зручно використовувати такий елементі інтерфейсу для зображення величин, які будуть зменшуватись, або збільшуватись, при цьому не зважаючи на їх величину, розмір панелі залишиться незмінним. На рисунку 1.29 зображено створений слайдер:

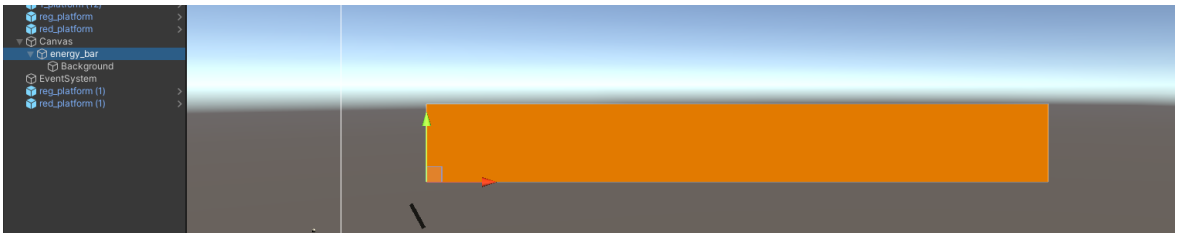


Рисунок 1.29. Слайдер та його ігровий об’єкт у Ієрархії ігровим об’єктів

Після цього достатньо у коді ui_manager.cs створити поле, в яке ми додаємо посилання на об’єкт слайдеру. Після чого напишу метод, який буде отримувати значення кількості енергії та змінювати положення слайдеру інтерфейсу. Дуже важливо, щоби максимальна кількість енергії дорівнювала властивості MaxValue компоненту Slider. Нижче приведено код скрипту ui_manager.cs:

```
[SerializeField] Slider energy_bar;

public void set_energy_bar_value(float value)
{
    energy_bar.value = value;
}
```

Також, оскільки ми використовуємо енергію, потрібно її відновлювати. Для цього потрібно реалізувати відповідну функцію у скрипті параметрів гравця. Втім, потрібно пам’ятати, що можуть бути два джерела відновлення енергії – власне та зовнішнє. Щоби оптимізувати код, створюємо метод регенерації енергії

regen_player_energy(float amount) в який будемо передавати як аргумент кількість енергії, що хочемо додати. Сам метод виконує додавання цієї кількості до кількості енергії, перевіряти чи значення енергії не перевищує 200. Якщо кількість енергії більша за 200, то її кількість встановлюється у 200. Потім викликається скрипт інтерфейсу гравця ui_manager.cs та його метод з публічним доступом set_energy_bar_value(), для відображення отриманих змін.

Власну регенерацію енергії можна реалізувати достатньо просто – створюю метод FixedUpdate() та в ньому, у разі якщо кількість енергії нижча за 200, викликаю метод regen_player_energy() та подаю як аргумент значення 0.1f, щоб регенерація була досить повільною. Я використовую метод FixedUpdate(), а не Update() з тієї ж причини, що і під час руху, адже у другому випадку енергія буде відновлюватись дуже швидко. Нижче приведено написаний код механіки регенерації:

```
public void regen_player_energy(float amount)
{
   player_energy += amount;
   if (player_energy > 200f)
      player_energy = 200f;

   _ui_manager.set_energy_bar_value(player_energy);
}
private void FixedUpdate()
{
   if(player_energy < 200)
   {
      regen_player_energy(0.1f);
   }
}
```

Залишилось реалізувати код впливу платформ на енергію сфери гравця. Для цього створюю обом платформам свої скрипти reg_pl_code.cs та red_pl_code.cs. Перший буде відповідати за регенерацію енергії а другий за її зменшення. Код для цих скриптів дуже простий, втім є одна важлива деталь. Для створення гнучкості в плані ігрового дизайну, кількість генерації енергії та її зменшення потрібно зробити плаваючою, щоби розробник міг сам встановлювати це значення в залежності від ігрової ситуації. Код для скрипту регенерації зображено нижче:

```
[SerializeField] float regen_rate = 0.3f;

public float return_reg_rate()
{
   return regen_rate;
}
```

Як можна побачити код складається з створення поля змінної кількості генерації енергії для компоненту скрипта, а також повернення по виклику цієї кількості. Нижче зображено код для скрипта зменшення енергії гравця:

```
[SerializeField] float reduce_rate = 0.3f;

public float return_red_rate()
{
   return reduce_rate;
}
```

Принцип роботи цього коду абсолютно однаковий, але він створений в різних скриптах не випадково. Механізм зміни енергії буде виконуватись у скрипті параметрів гравця player_parameters.cs. Раніше, для сфери гравця було створено компонент колайдеру, який працює в режимі тригеру. Цей колайдер буде зчитувати, коли сфера взаємодіє із іншими колайдерами. Таким чином, ми можемо зрозуміти, коли гравець торкається платформи з особливими властивостями. А для того, щоби відрізняти ці властивості і було створено два різних коди.

В скрипті параметрів гравця player_parameters.cs, я викликаю тригер залишання колайдеру-тригеру у іншому калайдеру. Тому, якщо гравець зупиниться на особливій платформі, цей тригер почне виконуватись. Нижче приведено код:

```
private void OnTriggerStay(Collider other)
{
   red_pl_code _red_pl_code;
   reg_pl_code _reg_pl_code;
   _red_pl_code = other.GetComponent<red_pl_code>();
   _reg_pl_code = other.GetComponent<reg_pl_code>();
   if(_red_pl_code)
   {
      reduce_player_energy(_red_pl_code.return_red_rate());
   }
```

```
        if (_reg_pl_code)
        {
            regen_player_energy(_reg_pl_code.return_reg_rate());
        }
    }
```

Як можна бачити, спочатку створюються змінні-посилання типу скриптів особливих платформ. Після цього виконується намагання отримати посилання на ці скрипти, як компоненти. Наступною дією є перевірка, чи було отримано будь-яке із посилань, і якщо перевірка повертає істину, тоді виконуються методи reduce_player_energy() або regen_player_energy(), відповідно до того, на якій платформі залишається гравець.

Було реалізовано основні ігрові функції та модулі проекту. Реалізовані графічні та фізичні матеріали, префаби платформ. Таким чином проект отримав свою кінцеву структуру, яку можна побачити на рисунку 1.30:

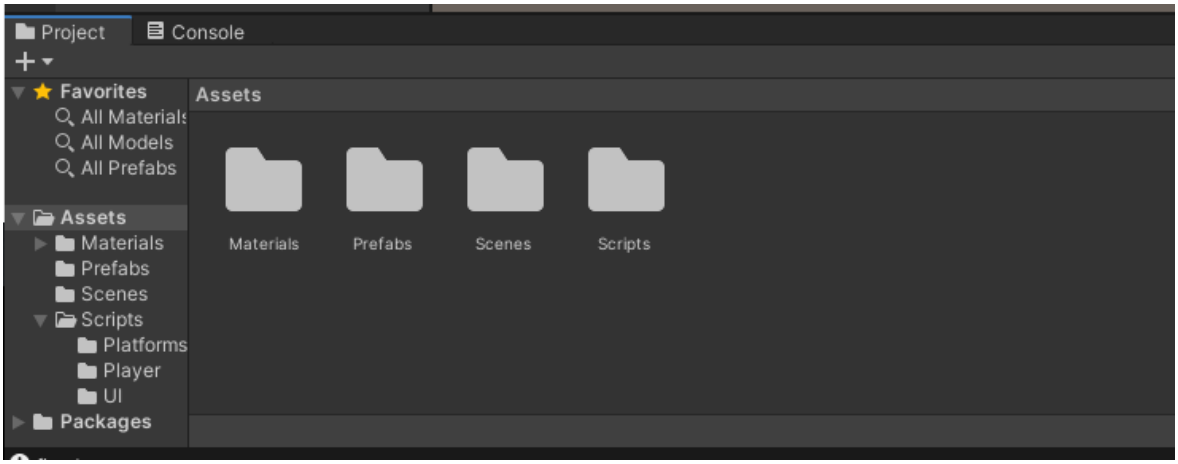


Рисунок 1.30. Кінцева структура проекту гри

Як можна бачити всі основні файли, з якими проводилась робота розміщені у підкаталогах. Також до складу проекту входять файли ігрового програмного рушія Unity. Всі файли проекту було розподіллено по підкаталогам розділу проекту Assets, розміщення котрих можна побачити на рисунку 1.31.

Поточна реалізація проекту дає змогу зіграти у платформер із використанням фізики ігрового програмного рушія Unity, створити більш складні рівні та ігрові ситуації. Втім така реалізація також дає можливість розширювати ігровий процес. Додавати нові типи платформ, або способи взаємодії із ними, а також у подальшому додавати нові режими гри, наприклад для гри на виділений

час, або навпаки для створення рекордів та змагання між гравцями у мережі Інтернет.

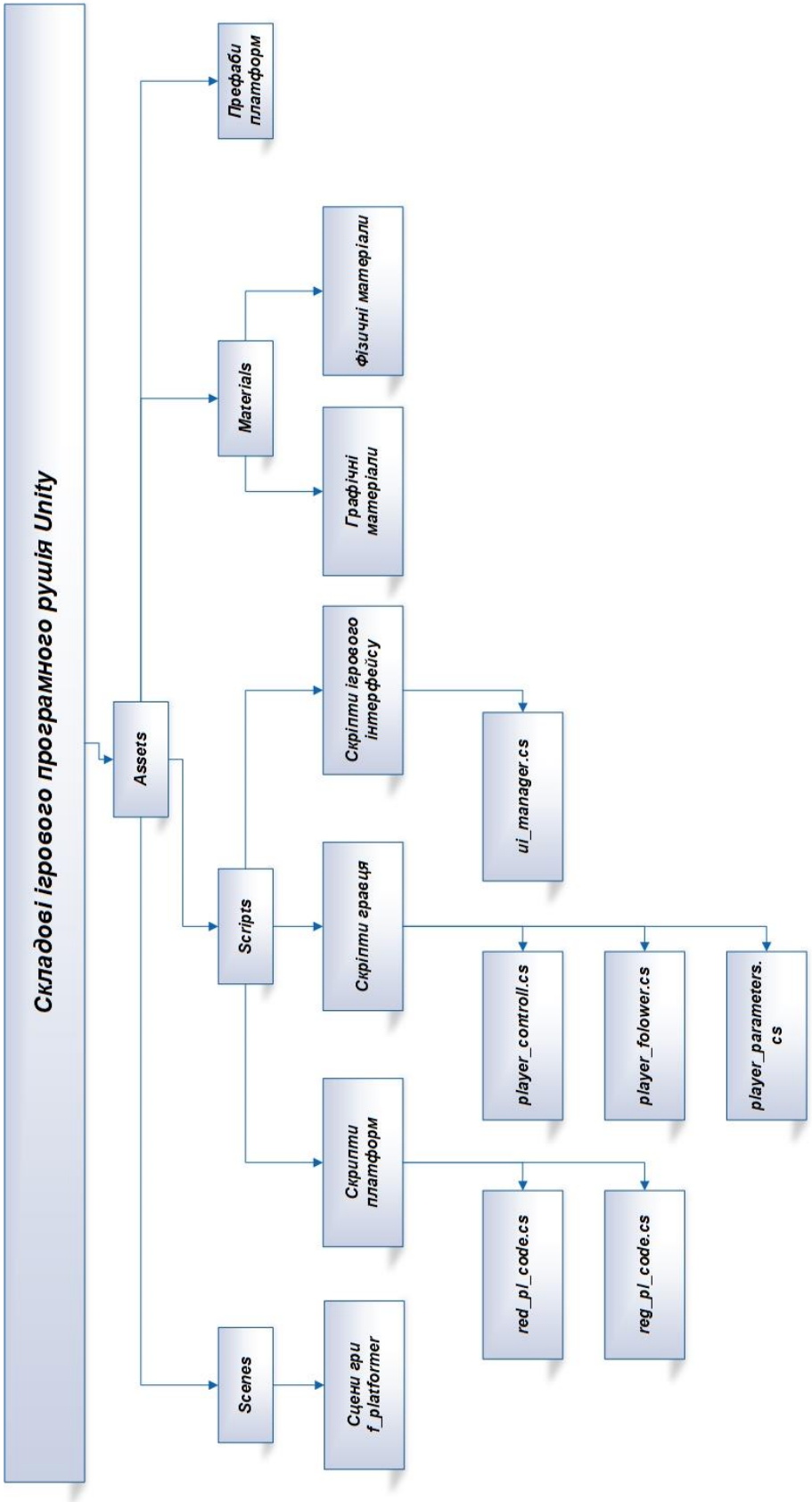


Рисунок 1.31. Розміщення файлів проекту по підкаталогам розділу Assets

1.7 Тестування працездатності ігрових елементів

Для якісної реалізації будь-якої гри, її необхідно тестувати та відлагоджувати не тільки під час розробки, а ще й після закінчення кожного етапу розробки. Багато ігрових продуктів продовжують тестувати навіть тоді, коли гра у фінальній версії вже відправилась до печаті, або у магазини. Це робиться для того, щоби якомога швидше виправляти помилки, які не були знайдені під час розробки.

У випадку з реалізацією ігор на ігровому програмному рушії Unity, цей процес виконується безпосередньо в самому редакторі. Для тестування та відладки гри під час розробки, потрібно встановлювати ігрові об’єкти та сцени у такий стан, який потрібно протестувати та натиснути кнопку «Play». Таким же чином виконується тестування і відладка після закінчення розробки, достатньо виставити гру у її початковий стан и натиснути ту ж саму кнопку. На рисунку 1.32 можна побачити процес тестування гри:

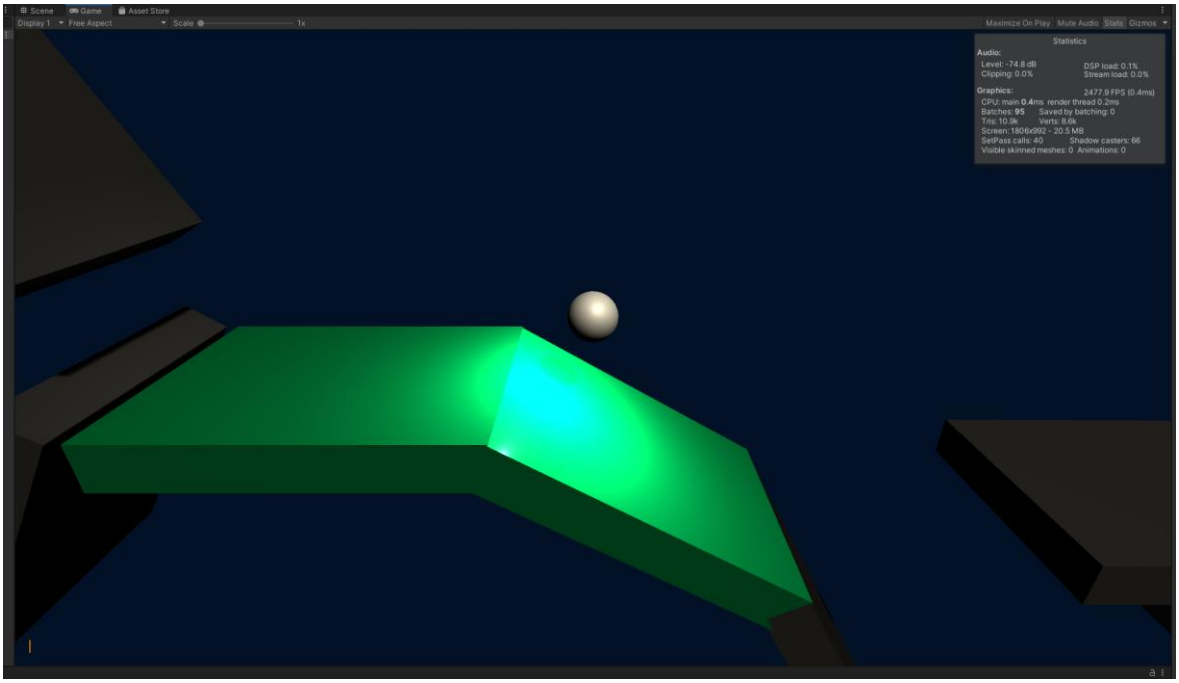


Рисунок 1.32. Процес тестування гри

Основною метою тестування є перевірка працездатності ігрових механік та управління сферою гравця. Також, цей процес відіграє важливу роль у балансуванні ігрового процесу та побудові карти рівня. Оскільки дає змогу швидко та оперативно тестувати ті, чи інші ігрові ситуації.

В результаті тестування було знайдено деякі помилки. Хоча по суті це не є помилками у коді. Деякі платформи не правильно реагували з гравцем в рамках одного типу платформ. Ця проблема виникла через те, що фізичні матеріали не правильно були додані до платформ. Не до префабу цих платформ, а безпосередньо до них, що створювало унікальні «варіанти» префабу платформи.

Була знайдена несправність у роботі коду відновлення енергії сфери гравця. Платформа, що мала відновлювати гравцю енергію, не робила цього. Помилка була виправлена створенням зв’язку між платформою та сферою гравця, яка містила в собі код параметрів гравця.

В цілому гра працює справно, да дає можливість отримати ігровий досвід гри в жанрі платформер, та подивитись на взаємодію сфери гравця із физикою платформ.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

В даному дипломному проекті розроблено та реалізовано 2D-гру в жанрі платформер ігровому програмному рушії Unity. Виготовлений програмний продукт дає змогу зіграти в гру, яку можна запустити на операційний системі Microsoft Windows. Розробка програмного продукту виконувалась за допомогою умовно безкоштовного ігрового програмного рушія Unity із використанням редактора коду Microsoft Visual Studio, систем контролю версій та графічного редактора. Розроблене програмне забезпечення було протестовано на помилки в коді та графічному супроводженні. Виявлені недоліки було усунуто за допомогою інструментів програмного рушія Unity.

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення.

Оцінка якості програмного продукту з точки зору користувача визначається необхідним на стадії функціонування розміром оперативної пам'яті ЕОТ, витратами машинного часу, пропускною спроможністю каналів передачі даних. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Таблиця 2.1. Каталог аналогів

Найменування ПЗ	*Обсяг функції ПП = V_0, ум. машинних команд*
1. ПП СУБД	1300 – 8600
2. ПП введення інформації	1800 – 8800
3. ПП оптим. розрахунків	13000 – 10200

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2 Розрахунок трудомісткості

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар = 244 х 0,8 = 195.20 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \qquad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \qquad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \qquad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.2.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.3.);Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.4.).

Таблиця 2.2 Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.3. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПО	1,75 – 1,2
Б	ПО – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПО маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4 Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПО типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Та*L_1*К_н=195,2 *0,12*0,7= 16,40 \text{ (люд/годин)}$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Та*L_2*К_н =195,2 *0,11*0,7 = 15,04 \text{ (люд/годин)}$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Та*L_3*К_н*К_т=195,2 *0,61*0,7*0,7= 58,35 \text{ (люд/годин)}$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$=2 (стр), розробка ТП $N_{тп}$=15(стр), розробка

робочого проекту $N_{рп}$=25 (стр), пояснювальна записка відповідно $N_{пз}$=50 (стр)

Розрахунок зведений у таблицю 2.5

Таблиця 2.5 Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1	2	3	4
1.ТЗ	$T_{Ртз}$=16,40	$T_{кк}=0,7*N_{тз}$= 0,7*2=1,4	$T_{нк}=0,15*N_{тз}$=0,15*2=0,30
2.Розробка ТП	$T_{Ртп}$=15,04	$T_{кк}=0,7*N_{тп}$=0,7*15=10,50	$T_{нк}=0,15*N_{тп}$=0,15*15=2,25
3.Розробка РП	$T_{Ррп}$=58,35	$T_{кк}=0,7*N_{рп}$=0,7*25=17,5	$T_{нк}=0,15*N_{рп}$=0,15*25=3,75
4.Розробка ПЗ	$T_{пз}=1,5**N_{пз}$=1,5*50 =75	$T_{кк}=0,7*N_{тз}$=0,7*50=35	$T_{нк}=0,15*N_{пз}$=0,15*50 =7.5
Усього, в т.ч.:	230,2		
- на розробку	$\Sigma T_р$=152		
- контроль керівника		$\Sigma T_{кк}$= 64,4	
- нормоконтроль			$\Sigma T_{нк}$=13,8

2.3 Розрахунок ціни програмного продукту

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПЗ. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2024» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2024 року - 6500 гривень; мінімальну погодинну тарифну ставку – 39.26 грн.

Таблиця 2.6 Розрахунок основної заробітної плати виконавців

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	152	39.26	5958,15
2.Контроль керівника	65	38,50	2502,50
3.Нормоконт-роль	14	38,50	539,00
Усього	-	-	ΣЗо= 8999,15

Таблиця 2.7 Розрахунок матеріальних витрат на розробку ПЗ

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	60	2.60	160,0
Разом	-	-	-	$В_{м\,і}$ =160,0
Транспортно– заготівельні витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$= 0,1х160.0=16,0
Усього				$В_м$=$В_{мі}$+$В_{тр\_з=}$ 176,0

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.8.

Таблиця 2.8 Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	176,0	$В_м$ (див. табл. 2.7)
2. Основна заробітна плата	8999,15	$З_о$ (див. табл. 2.6)
3. Додаткова заробітна плата	1349,87	$Зд = 0{,}15 \times Зо$=8999,15х0,15
4. Відрахування до єдиного фонду соціального внеску	2276,78	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$=0,22х (8999,15+ 1349,87)
5. Накладні витрати	2699,75	$Внак. = 0{,}3 \times Зо$=0,3 х 8999,15
6. Повна собівартість	15501,55	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 176,0+8999,15+1349,87+2276,78+ 2699,75

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$П=(С_п*Р)/100$= (15501,55*10)/100 = 1550,15грн (2.4)

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о = С_п+П$ =15501,55+1550,15= 17051,70 грн (2.5)

Податок на додану вартість визначаємо по наступній формулі:

ПДВ = 0.2 * $Ц_о$ = 17051,70 *0,2 = 3410,34грн; (2.6)

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$Ц_р=Ц_о$+ПДВ = 17051,70 + 3410,34 = 20462,04 грн (2.7)

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

3.1 Вступ

Охорона праці, як соціальний чинник, відіграє на підприємстві важливу, роль оскільки, якими б важливими не були трудові здобутки, вони не можуть компенсувати людині втраченого здоров'я, а тим більше життя. Те і інше дається лише один раз. Необхідно пам'ятати, що внаслідок нещасних випадків та аварій гинуть на виробництві не просто робітники та службовці, на підготовку яких держава вкладає значні кошти, а перш за все люди – годувальники сімей, батьки та матері дітей. Незадовільний стан охорони праці відображається на економіці держави. Служба охорони праці створюється на підприємствах незалежно від форми власності та видів діяльності для виконання правових, організаційно-технічних, санітарно-гігієнічних, соціально-економічних і лікувально-профілактичних заходів, спрямованих на запобігання нещасних випадків, професійних захворювань і аваріям в процесі праці. Основна мета всіх цих заходів – створити на підприємстві безпечні та здорові умови праці.

У розділі охорона праці дипломного проекту наведені характеристики приміщень, де експлуатуються ВДТ. До розгляду взято робоче місце програміста (оператора ЕОМ).

3.2 Аналіз небезпечних та шкідливих чинників, що впливають на працівника

Оператори ПК і програмісти зіштовхуються із впливом таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, недостатня освітленість робочої зони, електричний струм та інші. Тому на робочому місці програміста повинні бути створені умови для високопродуктивної праці.

Перетворення і обробка інформації проводиться за допомогою ПК. Робота може кваліфікуватися як робота оператором ЕОМ.

3.3 Розробка заходів з охорони праці

3.3.1 Виробничі приміщення

При плануванні виробничого приміщення врахована санітарна характеристика виробничих процесів, дотримуються норми корисної площі для працюючих, а також нормативи площ для розташування устаткування, що забезпечують безпечну роботу та зручне обслуговування устаткування.

Об'ємно-планувальні рішення будівель та приміщень для роботи з ВДТ мають відповідати вимогам ДСанПіН 3.3.2.007-98. Розміщення робочих місць З ВДТ ЕОМ і ПЕОМ у підвальних приміщеннях, на цокольних поверхах заборонено. Площа на одне робоче місце становить не менше ніж 6,0 м^2, а об'єм – не менше ніж 20,0м^3.

Виробничі приміщення повинні обладнуватися шафами для зберігання документів, полицями, стелажами, тумбами тощо, з урахуванням вимог до плащі приміщення.

У приміщеннях з ВДТ слід щоденно робити вологе прибирання. Приміщення повинні бути оснащені аптечками першої медичної допомоги.

3.3.2 Мікроклімат робочої зони працівників, вентиляція

У виробничих приміщеннях на робочих місцях з ВДТ мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря (ДСанПіН 3.3.2.007-98).

Таблиця 3.1. Норми мікроклімату для приміщень з ВДТ ЕОМ та ПЕМ

Пора року	Категорія робіт	Температура повітря, С, не більше	Відносна вологість повітря %	Швидкість руху повітря, м/с
Холодна	Легка-1а	22-24	40-60	0,1
	Легка-1б	21-23	40-60	0,1
Тепла	Легка-1а	23-25	40-60	0,1
	Легка-1б	22-24	40-60	0,1

Рівні позитивних і негативних іонів у повітрі приміщень з ВДТ мають відповідати санітарно-гігієнічним нормам № 2152-80.

Таблиця 3.2. Рівні позитивних і негативних іонів у повітрі

Рівні	Число іонів в 1 $см^3$ повітря	Число іонів в 1 $см^3$ повітря
	n+	n-
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально допустимі	50000	50000

3.3.3 Освітлення робочого місця, шум, вібрація

Штучне освітлення в приміщеннях з робочими місцями, обладнаними ВДТ має здійснюватись системою загального рівномірного освітлення. У виробничих та адміністративних приміщеннях, у разі переважної роботи з документами, допускається застосування системи комбінованого освітлення – крім системи загального освітлення додатково встановлюються світильники місцевого освітлення.

Значення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300-500лк.

Як джерела світла для штучного освітлення мають застосовуватись переважно люмінесцентні лампи типу ЛД. Допускається застосування ламп розжарювання у світильниках місцевого освітлення.

3.3.4 Організація робочого місця користувача ПК

Робочі місця слід так розташовувати відносно світових прорізів, щоб природне світло падало збоку, переважно зліва. При розміщенні робочих столів з ВДТ слід дотримуватися таких відстаней: між бічними поверхнями ВДТ -1,2м; від тильної поверхні одного ВДТ до екрану іншого – 2,5м.

Екран ВДТ має розташовуватися на оптимальній відстані від очей користувача, що становить 600…700 мм, але не ближче ніж за 600 мм з урахуванням розміру літерно-цифрових знаків і символів.

Клавіатуру розташовують на поверхні столу на відстані 100…300 мм від краю, зверненого до працюючого. У конструкції клавіатури має передбачатися

опорний пристрій, який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5…15^0.

При оснащенні робочого місця лазерним принтером параметри лазерного випромінювання повинні відповідати вимогам СанПіН № 5804-91.

ЕОМ ВДТ і ПК , інше устаткування , електропроводи та кабелі за виконанням і ступенем захисту мають відповідати класу зони за НПАОП 40.1-1.01-97, мати апаратуру захисту від струму короткого замикання та інших аварійних режимів. У приміщеннях, де одночасно експлуатується понад п’ять ЕОМ встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення. Не допускається підключати ЕОМ з ВДТ і ПК до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв

3.3.5 Електробезпека

Це система організаційних і технічних заходів та засобів, що забезпечують захист людей від шкідливої і небезпечної дії електричного струму, електричної дуги, електричного поля і статичної електрики.

Основні технічні засоби і заходи забезпечення електробезпеки при нормальному режимі роботи електроустановок включають:

- ізоляцію струмовідних частин;
- недоступність струмовідних частин;
- блоківки безпеки;
- засоби орієнтації в електроустановках;
- виконання електроустановок, ізольованих від землі;
- захисне розділення електричних мереж;
- компенсацію ємнісних струмів замикання на землю;
- вирівнювання потенціалів.

Із метою підвищення рівня безпеки, залежно від призначення, умов експлуатації і конструкції, в електроустановках застосовується одночасно більшість з перерахованих технічних засобів і заходів.

Особа відповідальна за електрогосподарство призначається з числа працівників, які мають не нижче IV групи з електробезпеки та відповідний стаж роботи для обслуговування електроустановок несе персональну відповідальність за допущення працівника використовувати в роботі електричну енергію

3.4 Пожежна безпека

Пожежна небезпека – це можливість виникнення та розвитку пожежі в будь-якій речовині, процесі, стані.. Коли людина перебуває в зоні впливу пожежі, то вона може потрапити під дію наступних небезпечних та шкідливих факторів: токсичні продукти згорання, вогонь, підвищена температура середовища, дим, недостатність кисню, руйнування будівельних конструкцій, вибухи, паніка.

Усі працівники повинні вміти користуватись наявними вогнегасниками, іншими первинними засобами пожежогасіння, знати місце їх знаходження.

До первинних засобів пожежогасіння відносяться:

- вогнегасники;
- пожежний інвентар (покривала з негорючого теплоізоляційного полотна, грубововняної тканини або повсті;
- ящики з піском;
- бочки з водою, пожежні відра, совкові лопати) та пожежний інструмент (гаки, ломи, сокири тощо).

Пожежні щити (стенди) встановлюються на території об’єкта з розрахунку один щит (стенд) на площу 5000 м2.

Вибір типу та визначення необхідної кількості вогнегасників здійснюється відповідно до Типових норм належності вогнегасників, затверджених наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи. Евакуаційні шляхи і виходи повинні втримуватися відповідно до НАПБ А.01.001 2004, бути вільними, нічим не захаращуватися і у разі виникнення пожежі забезпечувати безпеку під час евакуації всіх людей, які перебувають у приміщеннях будівель та споруд.

ВИСНОВКИ

Протягом виконання дипломного проектування, було проведено аналіз існуючих ігрових проектів в жанрі платформеру. Це дало змогу виділити основні риси притаманні цьому типу ігор. Виділивши їх, було розроблено концепцію ігрового процесу на невеликий дизайн-документ із основними ідеями для проекту.

Процес розробки було поділено на етами проектування, розробки та відладки. Під час проектування, було розроблено основні модулі гри та принципи їх роботи. Закладені основні механізми взаємодії між ними. Було прорoблено принципи роботи із фізикою програмного рушія Unity та способи її імплементації у ігровий процес.

На етапі розробки всі спроектовані модулі було реалізовано у вигляді скриптів, ігрових об’єктів та компонентів для них. Кожний елемент гри було зв’язано із іншим, виконано розбиття директорії гри на підкаталоги, які відповідають змісту для біль простого пакування проекту. Для виконання розробки проекту використовувалась мова програмування C#, як середа розробки VisualStudio, а для роботи з графікою використано Adobe Photoshop.

Етап відладки проходив у декілька етапів, приділяючи основну увагу справного виконання ігрового коду, знаходження помилок. Справної роботи фізики. Перевіряючи один ігровий елемент за одним було знайдено деякі помилки. Вони були виправлені, а ігровий процес доведено до більшого балансу з точки зору складності.

В результаті роботи було реалізовано 2D-гру в жанрі платформер, яка використовує, як основний ігровий механізм, фізику програмного рушія Unity. Це дало змогу ввести в ігровий процес елементи використання фізики для проходження рівнів, та створило трохи інший ігровий процес.

Модульна розробка дала змогу створити основу для подальших розробок ігрових проектів у жанрі платформер. Окрім того, завдяки такому підходу, процес розробки проектів буде більш автоматизований.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Гусєв Б.С. Комп’ютерна схемотехніка [навчальний посібник] // – К.: НУБіП України, 2022. – 264с.
2. Матвієнко М.П., Розен В. П. Комп’ютерна схемотехніка: навчальний посібник. – К.: Видавництво Ліра-К, 2020. – 192 с.
3. Дейбук В.Г. Віртуальний електронний практикум: Навчальний посібник – Чернівці: Чернівецький нац. ун-т, 2021. – 188 с.
4. Трофименко О.Г. С++. Алгоритмізація та програмування: підручник / О.Г. Трофименко, Ю.В. Прокоп, Н.І. Логінова, О.В. Задерейко. 2-ге вид. перероб. і доповн. – Одеса : Фенікс, 2019. – 477 с.
5. Азаров О. Д., Гарнага В. А., Клятченко Я. М., Тарасенко В. П. Комп’ютерна схемотехніка: підручник. – Вінниця: ВНТУ, 2018. – 230 с.
6. Архангельский А.Я. Программирование в C++Builder, 7-е изд. – М.: ООО Бином-Пресс, 2010 г. – 1230с., ил.
7. Нікулін В.С. Перетворювальні пристрої, ведені мережею: Конспект лекцій. –Харків: УкрДАЗТ, 2008. – Ч.4. – 85 с.
8. Мосіюк О.О., Федорчук А.Л. Операційні системи та системне програмування: навчально-методичний посібник. Житомир: Вид-во ЖДУ ім. Івана Франка, 2022. – 76 с.
9. Stroustrup B. A Tour of C++ (Second Edition). – Addison-Wesley, 2018. – 240 p. – ISBN 978-0-13-499783-4.
10. Каганюк О.К. Комп'ютерна схемотехніка: Навчальний посібник. – Луцьк: РРВ Луцького НТУ, 2016. – 236 с.
11. Бібліотека MSDN [Електронний ресурс]. – Режим доступу: URL http://msdn.microsoft.com/ru-ru/library/default.aspx.

ДОДАТОК А. Лістинг коду основних модулів гри мовою C#

Скрипт player_controll.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class player_controll : MonoBehaviour
{
    player_parameters _player_parameters;
    Rigidbody player_rb;
    float player_speed = 10f;

    void Start()
    {
        _player_parameters = GetComponent<player_parameters>();
        player_rb = GetComponent<Rigidbody>();

    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            if(_player_parameters.get_player_energy() > 20f)
            {
                player_rb.AddForce(Vector3.up * 200f);
                _player_parameters.reduce_player_energy(20f);
            }
        }
    }

    private void FixedUpdate()
    {
        if (Input.GetAxis("Horizontal") > 0f)
        {
            player_rb.AddForce(Vector3.right * player_speed);
        }
        if (Input.GetAxis("Horizontal") < 0f)
        {
            player_rb.AddForce(Vector3.right * -player_speed);
        }
```

```
    }
}
```

Скрипт player_folower.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class player_folower : MonoBehaviour
{
    [SerializeField] Transform player_transform;

    private void Update()
    {
        transform.position = new Vector3(player_transform.position.x,
player_transform.position.y+4f, transform.position.z);
    }
}
```

Скрипт player_parameters.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class player_parameters : MonoBehaviour
{
    [SerializeField] ui_manager _ui_manager;
    float player_energy = 200f;

    public void reduce_player_energy(float amount)
    {
        player_energy -= amount;
        if (player_energy < 0)
            player_energy = 0;
        _ui_manager.set_energy_bar_value(player_energy);
    }

    public void regen_player_energy(float amount)
    {
        player_energy += amount;
        if (player_energy > 200f)
            player_energy = 200f;
        _ui_manager.set_energy_bar_value(player_energy);
    }
```

```
    public float get_player_energy()
    {
        return player_energy;
    }

    private void FixedUpdate()
    {
        if(player_energy < 200)
        {
            regen_player_energy(0.1f);
        }
    }

    private void OnTriggerStay(Collider other)
    {
        red_pl_code _red_pl_code;
        reg_pl_code _reg_pl_code;
        _red_pl_code = other.GetComponent<red_pl_code>();
        _reg_pl_code = other.GetComponent<reg_pl_code>();
        if(_red_pl_code)
        {
            reduce_player_energy(_red_pl_code.return_red_rate());
        }
        if (_reg_pl_code)
        {
            regen_player_energy(_reg_pl_code.return_reg_rate());
        }
    }
}
```

Скрипт red_pl_code.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class red_pl_code : MonoBehaviour
{
    [SerializeField] float reduce_rate = 0.3f;
    public float return_red_rate()
    {
        return reduce_rate;
    }
}
```

Скрипт reg_pl_code.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class reg_pl_code : MonoBehaviour
{
    [SerializeField] float regen_rate = 0.3f;
    public float return_reg_rate()
    {
        return regen_rate;
    }
}
```

Скрипт ui_manager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ui_manager : MonoBehaviour
{
    [SerializeField] Slider energy_bar;
    public void set_energy_bar_value(float value)
    {
        energy_bar.value = value;
    }
}
```

ДОДАТОК Б. Слайди мультимедійної презентації

Розробка 2D-гри в жанрі платформер з використанням фізики програмного рушія Unity

ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТА ГРУПИ 4КС-57: АЛЕКСЄЄНКО ДМИТРА СЕРГІЙОВИЧА

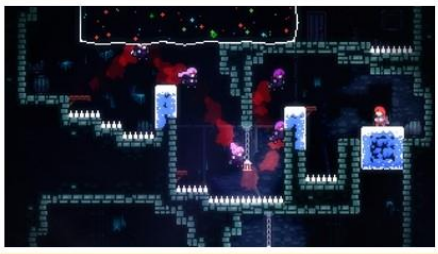
КЕРІВНИК: КІРЄЄВ І.А.

Особливості ігрового жанру платформер

Ігровий жанр платформер, також відомий як платформна гра, це вид відеоігор, в якому ігровий процес в основному пов'язаний зі стрибками по платформах і обходом перешкод.

Основними характеристиками жанру є:

- Ігровий процес акцентований на взаємодії із платформами;
- Вирішення просторових головоломок;
- Використання оточення для досягнення своїх цілей.



Особливості програмного рушія Unity та причини його вибору для розробки проекту

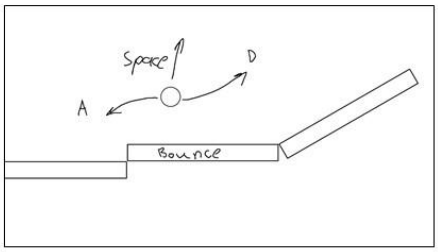
Ігровий двигун Unity є одним із популярніших ігрових двигунів у світі. За його допомогою створюють як мобільні, так і комп'ютерні ігри. Цей двигун досі активно розвивається!

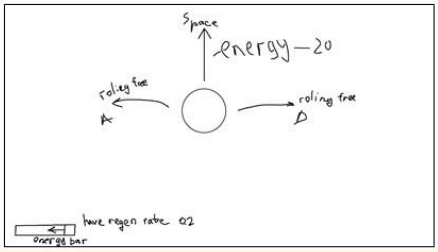
- Має зручну візуальну середу розробки;
- Має можливість міжплатформенної розробки ігор;
- Підтримує модульну систему компонентів;
- Використання мови програмування C# та його можливостей;
- Велике ком'юніті розробників;
- Зрозуміла та вичерпна документація, безліч курсів у інтернеті та літературі;
- Гнучка система ліцензування ігрового двигуна;
- Безплатні та платні ассети для проектів будь-якого рівня.

Особливості ігрових механік розроблюємого проекту

Можна виділити такі особливості розробляємого проекту:

- Гравець має свої параметри – енергію;
- Енергія витрачається на стрибки;
- Рух сфери гравця в сторони безкоштовний по енергії;
- Поверхні платформ мають різні властивості;
- Енергія поступово відновлюється;
- Деякі платформи можуть впливати на енергію гравця.



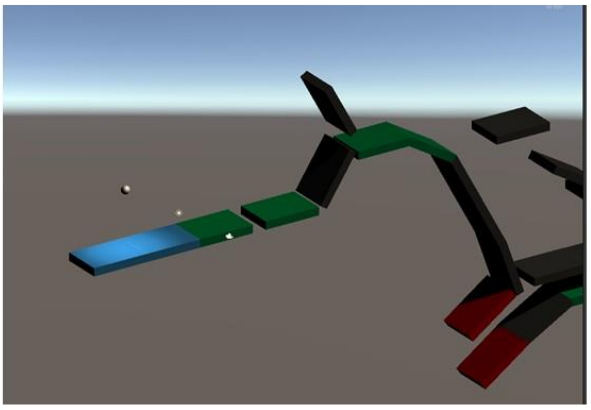


Огляд основних елементів ігрового процесу

Основним елементом ігрового процесу є переміщення по ігровому рівню, побудованому із платформ.

Переміщення виконується стрибками, або рухом по поверхням вліво та вправо.

Гравцю необхідно уважно слідкувати за платформами на рівні, використовуючи їх властивості на свою користь.

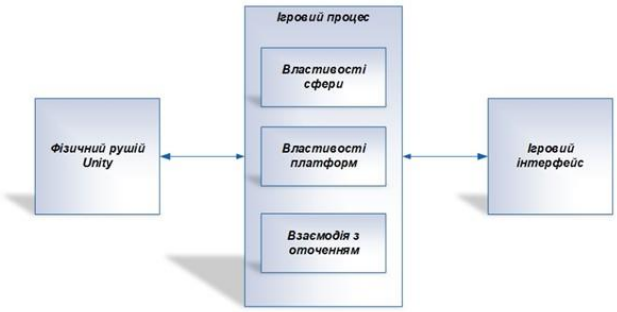


Принцип роботи основних елементів ігрового процесу

Структурно гра поділяється на взаємодію трьох основних складових:

1) Фізичного рушія Unity;
2) Ігрового процесу;
3) Ігрового інтерфейсу

Основною складовою гри є саме ігровий процес, оскільки фізична частина не взаємодії з інтерфейсом гри.

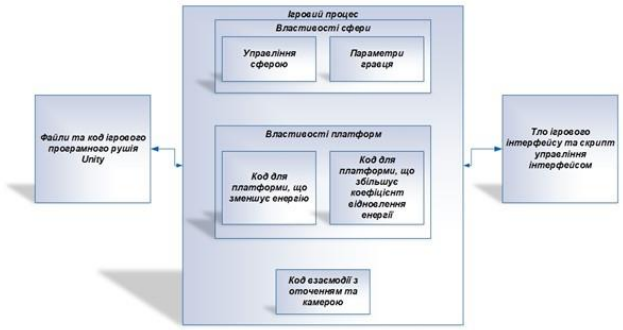


Принцип роботи основних елементів ігрового процесу

Основу ігрового процесу складають взаємодія властивостей сфери гравця та властивостей платформ.

Під час торкання сфери гравця до платформи, виконується розрахунок фізичних законів таких властивостей платформ як стрибучість та сила тертя.

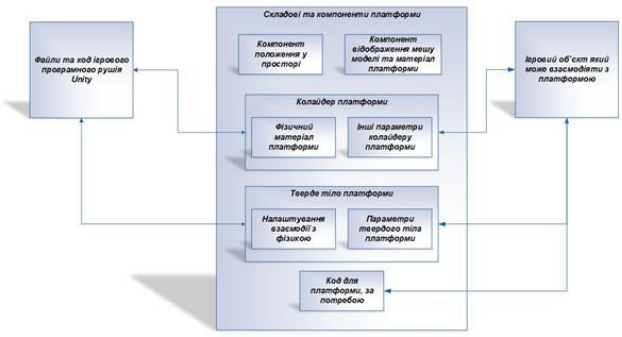
В результаті створюється фізична сила, яка впливає на сферу гравця, змушуючи її стрибати, або швидко гасити свою стрибучість.



Принцип роботи основних елементів ігрового процесу

Кожна ігрова платформа має типовий склад:

- Компонент положення у просторі платформи;
- Компонент графічного відображення платформи;
- Коллайдер платформи для симуляці границі твердого тіла та фізичного матеріалу поверхні;
- Тверде тіло для налаштування фізики твердого тіла;
- Код платформи для створення унікального впливу на сферу гравця.



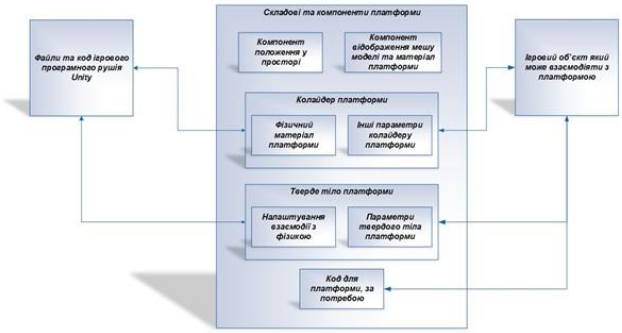
Принцип роботи основних елементів ігрового процесу

Всі платформи існують як копії оригіналів – префабів.

Кожна платформа може бути налаштована індивідуально, або використана як копія префабу.

Властивості фізичного матеріалу платформ можуть бути індивідуально налаштовані для платформ.

Це створює умови для створення різних ситуацій під час гри.



Етапи реалізації основних елементів ігрового процесу

Реалізація основних елементів гри складалась з:

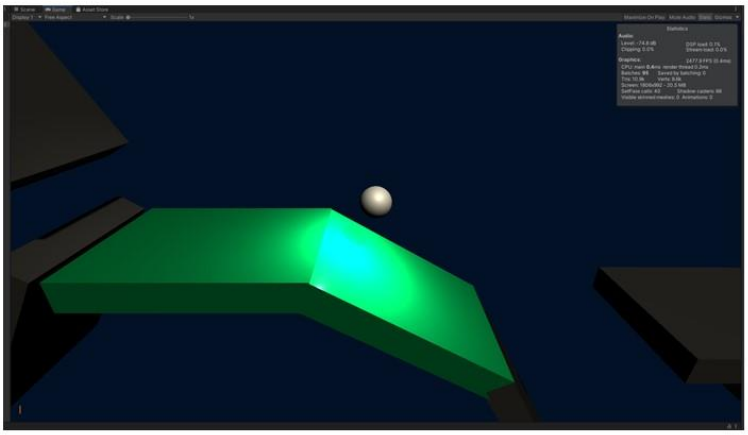
- Реалізації графічних матеріалів платформ;
- Реалізації фізичних матеріалів платформ;
- Реалізації префабів платформ;
- Реалізації сфери гравця;
- Створення рівня гри;
- Реалізації логіки сфери гравця;
- Реалізації логіки платформ.

Хід тестування гри

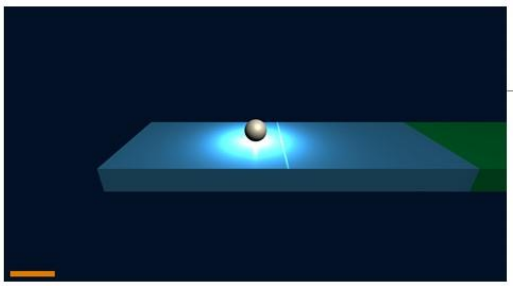
Тестування гри проводилось з допомогою вбудованого у ігровий програмний рушій Unity інструмент.

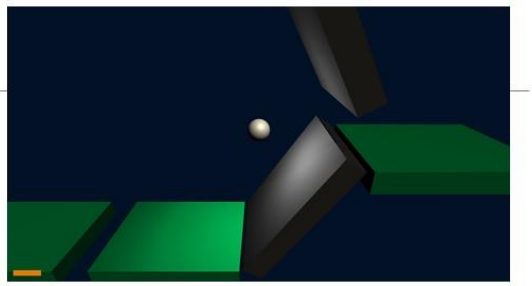
Метою тестування було:

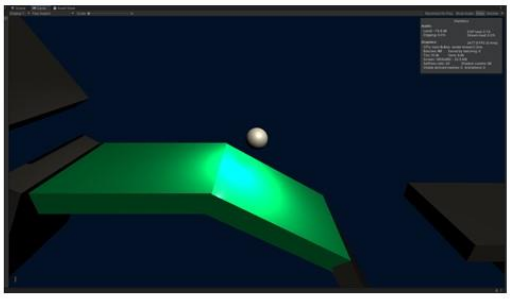
- Визначити коректність управління сферою гравця;
- Визначити коректність роботи фізики платформ;
- Визначити коректність роботи коду унікальних платформ, що впливають на сферу гравця.

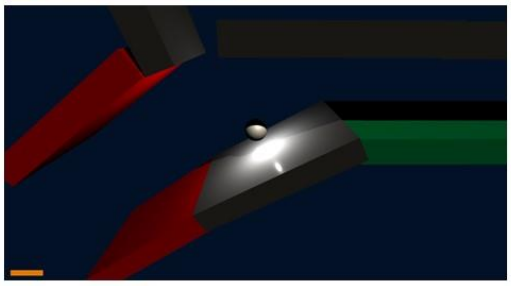


Скріншоти розробленої гри









ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Алексєєнка Дмитра Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність: 123 "Комп'ютерна інженерія"

Освітня програма: «Обслуговування комп'ютерних систем і мереж»

Тема дипломного проекту: Розробка 2D-гри у жанрі платформер з використанням фізики програмного рушія Unity

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 70 сторінок. У пояснювальній записці виконано опис предметної області, способи створення 2D-гри в жанрі топ-платформер. Проведено проектування та реалізація основних елементів гри. Графічна частина складається з 12 слайдів мультимедійної презентації, які передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.

б) самостійність роботи над проектом: Протягом всього строку дипломного проектування та переддипломної практики здобувач освіти Алексєєнко Д.С. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.

в) теоретична підготовка випускника (випускниці): Здобувач освіти Алексєєнко Д.С. під час роботи над дипломним проектом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проекту

г) вміння розв'язувати виробничі та конструкторські питання

Під час дипломного проектування здобувач освіти Алексєєнко Д.С. мав змогу самостійно приймати рішення з реалізації основних елементів гри, та показав вміння організовано працювати над поставленим завданням, складати схеми та проводити розробку коду за допомогою актуальних для теми комп'ютерних програмних засобів.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Добре

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові керівника дипломного проекту

Кірєєв Ігор Анатолійович

Місце роботи і посада керівника дипломного проекту

ВСП "Одеський технічний фаховий коледж ОНТУ", викладач комісії комп'ютерних технологій та програмної інженерії

Підпис

«10» 06 2024 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Алексєєнка Дмитра Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність 123 " Комп'ютерна інженерія "

Освітня програма «Обслуговування комп'ютерних систем і мереж»

Керівник дипломного проекту (роботи) Кірєєв Ігор Анатолійович

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) Розробка 2D-гри в жанрі платформер з використанням фізики програмного рушія Unity

Обсяг розрахунково-пояснювальної записки 70 сторінок

Обсяг графічної (презентаційної) частини 12 аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню

Представлений на рецензію дипломний проект повністю відповідає меті проектування та технічному завданню. Тематика дипломного проекту є актуальною для своєї галузі та присвячена питанням створення ігрових продуктів в цілому та розробці ігор на ігровому програмному рушії Unity, в цілому.

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Дипломний проект складається зі вступу, трьох розділів, висновків, переліку використаних джерел. У технологічному розділі розглянуті питання проблематики розробки гри на ігровому програмному рушії Unity, сформовано вимоги до гри згідно до теми дипломного проекту та завданню, виконано проектування основних аспектів розробляємої гри. За допомогою відповідного програмного забезпечення реалізовані всі намічені зміни до ігрового процесу

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату у роботі не виявлено

г) перелік позитивних якостей дипломного проекту (роботи)

1. Детально описано процес виконання розробки гри;
2. Виконано проектування елементів гри із поясненнями на схемах;
3. Розроблено функціонуючу гру в жанрі платформер за допомогою відповідних іструментів розробки.

д) основні недоліки дипломного проекту (роботи)

1. Гра має мало ігрових механік;
2. Графічна частина гри виконана слабо.

Оцінка розрахункової частини Добре

Оцінка графічної частини Добре

Загальна оцінка Добре

Прізвище, ім'я, по батькові рецензента к.т.н. Селіванова Алла Віталіївна

Місце роботи і посада рецензента Одеський національний технологічний університет, декан факультету комп'ютерної інженерії, програмування та кіберзахисту

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ, ПРОГРАМУВАННЯ ТА КІБЕРЗАХИСТУ

Підпис:

«17» червня 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григоріївна Краснокутська

ID перевірки:
1016325162

Дата перевірки:
05.06.2024 21:56:15 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
06.06.2024 14:11:11 EEST

ID користувача:
100011688

Назва документа: 4КС-57_Алексєєнко

Кількість сторінок: 43 Кількість слів: 8579 Кількість символів: 60047 Розмір файлу: 2.22 MB ID файлу: 1016123931

5.25%
Схожість

Найбільша схожість: 3.53% з Інтернет-джерелом (https://card-file.ontu.edu.ua/server/api/core/bitstreams/6c95086b-bff...

5.25% Джерела з Інтернету 53 ..... Сторінка 45

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи 1

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНГО ДИПЛОМНОГО ПРОЕКТА
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Алексєєнко Дмитро Сергійович,
здобувач освіти гр. 4КС-57, та

Кірєєв Ігор Анатолійович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускного дипломного проекту фахового молодшого бакалавра на тему:

«Розробка 2D-гри у жанрі платформер з використанням фізики програмного рушія Unity» (автор роботи – Алексєєнко Д.С., керівник роботи – Кірєєв І.А.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Алексєєнко Д.С. /

Керівник _____ / Кірєєв І.А./

«10» 06 2024 р.