



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД  
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ  
КАФЕДРА «ПРИКЛАДНА МАТЕМАТИКА ТА ІНФОРМАТИКА»

## НАВЧАЛЬНИЙ МОДУЛЬ

«Розробка ігрових додатків для ОС Android»  
дисципліни «Інструментальна підтримка розробки  
комп'ютерних ігрових додатків»  
підготовки студентів освітнього рівня «магістр» за  
спеціалізацією «Програмне забезпечення мультимедійних  
систем для ігрових додатків» спеціальності 121 «Інженерія  
програмного забезпечення»

**GAMEHUB: UNIVERSITY-ENTERPRISES COOPERATION IN GAME  
INDUSTRY IN UKRAINE”**

**GAMEHUB: СПІВРОБІТНИЦТВО МІЖ УНІВЕРСИТЕТАМИ ТА  
ПІДПРИЄМСТВАМИ В СФЕРІ ГРАЛЬНОЇ ІНДУСТРІЇ В УКРАЇНІ»**

561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP

*The handbooks were performed with support of the Erasmus+ Programme of the European Union (561728-EPP-1-2015-1-ES-EPPKA2-CBHE-JP). The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.*

Покровськ, 2018

УДК 004.032.6 : 004.92

Навчальний модуль «Розробка ігрових додатків для ОС Android» дисципліни «Інструментальна підтримка розробки комп’ютерних ігрових додатків» підготовки магістрів за спеціалізацією «Програмне забезпечення мультимедійних систем для ігрових додатків» спеціальності 121 «Інженерія програмного забезпечення» / Укладачі: С.О. Щололо, Ю.Л. Дікова. – Покровськ: ДонНТУ, 2018. – 93 с.

Навчальний модуль «Розробка ігрових додатків для ОС Android» дисципліни «Інструментальна підтримка розробки комп’ютерних ігрових додатків» входять до дисциплін підготовки студентів освітнього ступеня «магістр» спеціальності 121 «Програмна інженерія» за спеціалізацією «Програмне забезпечення мультимедійних систем для ігрових додатків», яка впроваджена в рамках виконання гранту Еразмус+ 561728-EPP-1-2015-1- ES-EPPKA2-CBHE-JP «GameHub: Співробітництво між університетами та підприємствами в сфері гральної індустрії в Україні»

Укладачі:

**С.О. Щололо**, доцент кафедри КІ, к.т.н., доцент;

**Ю.Л. Дікова**, старший викладач кафедри КІ.

Рецензенти:

**Святний В.А.**, завідувач кафедри КІ, д.т.н., професор;

**Вовна О.В.**, завідувач кафедри ЕТ, д.т.н., доцент.

Відповідальний за випуск: **Дмитрієва О.А.**, завідувач кафедри ПМІ.

Розглянуто на засіданні кафедри «Прикладної математики і інформатики»  
протокол № 11 від 15 травня 2018 року

Затверджено навчально-видавничим відділом ДонНТУ,  
протокол № 14 від 12 червня 2018 року

Рекомендовано до друку Вченю радою ДонНТУ,  
протокол № 10 від 21 червня 2018 року



Цей матеріал ліцензовано на умовах [Ліцензії Creative Commons Із Зазначенням Авторства — Некомерційна — Поширення На Тих Самих Умовах 4.0 Міжнародна](#).

## ЗМІСТ

<b>ЗАГАЛЬНИЙ ВСТУП</b>	<b>5</b>
<b>ДОВІДНИК МОДУЛЯ</b>	<b>7</b>
1    Вступ .....	8
2    Опис модуля .....	9
3    Мета та передбачувані результати вивчення модуля .....	9
4    Місце модуля в структурі дисципліни .....	11
5    Інформаційне наповнення змістового модуля 2 .....	11
6    Форми навчання.....	12
7    Порядок проведення атестації.....	13
8    Зворотній зв'язок.....	15
9    Викладацький склад та допоміжні джерела .....	15
10   Навчальна програма і матеріали.....	16
11   Рекомендована література та інтернет-посилання .....	28
<b>МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ</b>	<b>29</b>
1    Вступ. Загальні положення .....	30
2    Наскірне завдання .....	31
3    Обладнання .....	31
4    Попередні підготовчі кроки до виконання лабораторних робіт .....	32
5    Лабораторна робота № 1. Формування головних персонажів, базової механіки та засобів керування ігровим процесом.....	35
6    Лабораторна робота № 2. Програмування поведінки і взаємодії ігрових об'єктів, фізичних процесів та основних ефектів .....	51

<b>7</b>	<b>Лабораторна робота №3. Анімація персонажів, реалізація предметів і цілей, організація ігрових рівнів.</b>	<b>65</b>
<b>8</b>	<b>Заключний практичний семінар</b>	<b>79</b>
<b>9</b>	<b>Заключний звіт</b>	<b>81</b>
<b>10</b>	<b>Література</b>	<b>84</b>
	<b>Додаток А. Критерії оцінювання лабораторної роботи</b>	<b>85</b>
	<b>Додаток Б. Критерії оцінювання презентації результатів на заключному практичному семінарі</b>	<b>87</b>
	<b>Додаток В. Критерії оцінювання заключного звіту</b>	<b>91</b>
	<b>Додаток Д. Приклад оформлення титульної сторінки заключного звіту</b>	<b>95</b>

## **ЗАГАЛЬНИЙ ВСТУП**

В рамках виконання гранту Еразмус+ 561728-EPP-1-2015-1- ES-EPPKA2-CBNE-JP «GameHub: Співробітництво між університетами та підприємствами в сфері гральної індустрії в Україні» в Донецькому національному технічному університеті впроваджена підготовка студентів спеціальності 121 «Програмна інженерія» за спеціалізацією «Програмне забезпечення мультимедійних систем для ігорних додатків».

Навчальний план підготовки студентів на рівні «магістр» передбачає опрацювання наступних навчальних модулів:

- «Методи теорії ігор в ігрових додатках» (Theory of Games in Game Applications), дисципліна «Математична теорія ігор»;
- «Розробка ігрових додатків на базі движка Unity» (Game Applications Development Based on Unity Engine), дисципліна «Інструментальна підтримка розробки комп’ютерних ігрових додатків»;
- «Розробка ігрових додатків для OS Android» (Game Applications Development for OS Android Platform), дисципліна «Інструментальна підтримка розробки комп’ютерних ігрових додатків»;
- «Розробка ігрових додатків на базі HTML-5 для WEB» (Game Applications development for WEB based on HTML-5) , дисципліна «Інструментальна підтримка розробки комп’ютерних ігрових додатків»;
- «3D графіка в ігрових додатках (на базі графічного редактора Blender)» (3D Graphics in Game Applications (Based on Blender Game Engine), дисципліна «Комп’ютерний синтез та обробка зображень»;
- «Місце ігрових додатків на ринку програмного забезпечення» (Place of Game Applications in Software Market), дисципліна «Сучасні засоби інформатики та комп’ютерний ринок».

Навчальний план підготовки студентів на рівні «бакалавр» передбачає опрацювання наступних навчальних модулів:

- «Архітектура ігрових додатків» (Game Applications Architecture), дисципліна «Архітектура та проектування програмного забезпечення»;

- «Основи створення ігрових додатків (на базі GameMaker)» (Basics of Game Applications Development with GameMaker Studio), дисципліна «Комп’ютерна графіка»;
- «Особливості тестування ігрових додатків» (Features of Game Applications Testing), дисципліна «Якість ПЗ та тестування»;
- «Командна розробка ігрових додатків» (Team Development of Game Applications), дисципліна «Групова динаміка і комунікації».

Для кожного модуля розроблено: довідник модуля, опорні конспекти (презентації) лекцій та методичні вказівки виконання лабораторних робіт і практичних занять за модулем. Методичні матеріали всіх модулів, доступні за посиланням <http://89.185.3.253:9090/login> (*login: guest, pass: gamehub*).

Дане видання містить опис навчального модуля «Розробка ігрових додатків для ОС Android» дисципліни «Інструментальна підтримка розробки комп’ютерних ігрових додатків» входять до дисциплін підготовки студентів освітнього ступеня «магістр» спеціальності 121 «Програмна інженерія».

## **ДОВІДНИК МОДУЛЯ**

**«Розробка ігрових додатків для ОС Android»**

**дисципліни «Інструментальна підтримка розробки комп’ютерних  
ігрових додатків»**

**підготовки студентів освітнього рівня «магістр» за спеціалізацією  
«Програмне забезпечення мультимедійних систем для ігрових  
додатків» спеціальності 121 «Інженерія програмного забезпечення»**

## 1 Вступ

Метою викладання навчальної дисципліни “Інструментальна підтримка розробки комп’ютерних ігрових додатків” є формування практичних знань та вмінь студента в області розробки ігрових додатків для сучасних платформ.

Основними завданнями вивчення дисципліни “Інструментальна підтримка розробки комп’ютерних ігрових додатків” є:

- знайомство із основними інструментальними засобами створення сучасних ігрових додатків для різних апаратних та програмних платформ, порівняльний аналіз особливостей, переваг і недоліків цих платформ, оцінка перспектив їх розвитку;
- огляд сучасних движків створення ігрових додатків, що дозволяють оптимізувати та прискорити процес розробки за допомогою ефективної реалізації модульної розробки додатку та ефективної підтримки кросплатформеності;
- оволодіння повним циклом розробки типового ігрового додатку, організація та контроль процесів планування, розробки, просування та підтримки ігрового додатку, вміння спланувати роботу команди розробників відповідно до спеціалізації кожного з них;
- реалізація типових елементів ігрових додатків у відповідності до платформи реалізації, оцінка ефективності організації ігрової механіки з урахуванням типових сценаріїв використання інтерфейсу користувача та різних способів взаємодії з платформою;
- урахування особливостей платформи реалізації ігрового додатку, вміння використати ці особливості належним чином, оптимізація програмного коду відповідно до можливостей апаратних платформ.

Змістовний модуль «Розробка ігрових додатків для ОС Android» дисципліни «Інструментальна підтримка розробки комп’ютерних ігрових додатків» орієнтовно на оволодіння студентами практичних навичок створення мобільних ігрових додатків для операційної системи Android.

## 2 Опис модуля

Галузь знань: 12 «Інформаційні технології».

Напрям підготовки: 121 «Інженерія програмного забезпечення»

Рівень: Магістр

Назва дисципліни: Інструментальна підтримка розробки комп’ютерних ігрових додатків

Назва змістового модуля: Розробка ігрових додатків для ОС Android

Семестр: 1

Кількість кредитних одиниць: дисципліна – 6,0, модуль – 2,0

Орієнтовна кількість годин: дисципліна – 180, модуль – 60

Викладачі: доцент, к.т.н. Цололо С.О.

ст. викл. Дікова Ю.Л.

## 3 Мета та передбачувані результати вивчення модуля

### 3.1 Мета модуля

Мета модуля – ознайомити студентів з архітектурою та загальними принципами проектування додатків для Android, виконати аналіз властивостей платформи для розробки ігрових додатків, набути навичок створення ігрових додатків Android в середовищі Unity, а також публікації, просування і способи монетизації мобільних ігрових додатків.

#### 3.1.1 Знання та їх використання

У разі успішного оволодіння матеріалами модуля студент буде:

- знати загальні принципи побудови та особливості архітектури ігрових додатків на мобільній платформі Android;
- виконувати класифікацію ігрових мобільних додатків за типами проходження ігрового процесу, вміти виділити особливості розробки додатків відповідно до типу гри та характеристик мобільної платформи реалізації;
- розробляти та оцінювати загальну концепцію мобільної гри, а також створювати персонажів з набором властивостей та характеристик, що максимально відповідають обраній концепції;

- реалізовувати інтерфейс користувача і елементи управління мобільною грою в середовищі Unity з урахуванням особливостей керування процесом гри, які відповідають різним способам взаємодії користувачів з платформою Android;
- створювати анімацію і програмування руху персонажів за допомогою засобів середовища Unity, при цьому підкреслюючи особливості ігрової механіки поведінкою персонажів гри;
- виконувати налагодження, комплексне тестування і публікацію гри з оцінкою можливих шляхів ефективної монетизації та вдосконалення ігрового процесу.

### 3.1.2 Дослідницький навички

За підсумками вивчення модулю студент повинен вміти аналізувати можливості сучасних мобільних платформ для реалізації ігрових додатків та обирати найбільш прийнятні інструменти для вирішення задачі створення додатків для ОС Android.

### 3.1.3 Спеціальні вміння

У разі успішного вивчення модуля студент буде вміти:

- знати та використовувати основні програмні засоби розробки мобільних ігрових додатків для Android;
- оцінювати сучасні ігрові движки та фреймворки, що дозволять найбільш ефективно, просто та швидко реалізувати конкретний різновид мобільного ігрового додатку для Android;
- використовувати сучасний графічний движок Unity для розробки ігрових додатків різних типів та рівнів складності на мобільній платформі Android;
- знати повний цикл розробки Android-додатку та мати змогу підключитися до розробки ігрового мобільного додатку на будь-якому етапі реалізації.

### 3.1.4 Соціальні вміння

У разі успішного вивчення модуля студент буде здатний:

- творчо реалізувати свою діяльність, проявляти пізнавальну активність;
- застосувати раніше отримані знання як основу для засвоєння нових знань;

- обґрунтувати той чи інший спосіб діяльності при виконанні практичних завдань;
- проявляти навички лідерства та вміння, необхідні для виконання групової та командної роботи;
- своєчасно та самостійно виконувати пошук можливих помилок, знаходити способи їх усунення та попередження;
- застосовувати навички комунікативного інтерактивного та перцептивного спілкування, обирати моделі та стилі спілкування.

### 3.1.5 Особисті якості

У разі успішного вивчення модуля студент буде вміти:

- самостійно вирішувати питання, що відносяться усіх етапів розробки мобільного ігрового додатку для Android;
- вивчати та аналізувати онлайн-джерела та сучасну науково-технічну літературу з питань створення мобільних ігрових додатків для Android;
- оцінювати сучасні тенденції у створенні мобільних ігрових додатків, аналізувати перспективність розробки того чи іншого ігрового додатку для платформи Android.

## 4 Місце модуля в структурі дисципліни

<i>Номер</i>	<i>Змістовний модуль</i>	<i>Тиждень вивчення</i>
1	Розробка ігрових додатків на базі движка Unity	1-6
2	<b>Розробка ігрових додатків для ОС Android</b>	7-11
3	Розробка кросплатформених ігрових додатків на HTML5	12-16

## 5 Інформаційне наповнення змістового модуля 2

<i>Номер тижня</i>	<i>Зміст</i>
7	Архітектура і базові принципи проектування додатків для платформи операційної системи Android.

<i>Номер тижня</i>	<i>Зміст</i>
	Розробка загальної концепції мобільного ігрового додатку і формування головних персонажів.
8	Огляд основних властивостей платформи Android для розробки ігрових додатків. Проектування основних елементів інтерфейсу ігрового додатку та реалізація засобів управління ігровим процесом.
9	Особливості створення мобільних ігрових додатків для платформи Android в середовищі Unity Засоби анімації, програмування руху та реалізація освітлення персонажів і основних ігрових елементів
10	Публікація та засоби просування ігрового додатку, соціальна складова, оцінка шляхів ефективної монетизації Оцінка та тестування ігрового додатку з іншими студентами, балансування ігрової механіки на основі відгуків гравців, підготовка додатку до релізу та публікації, підготовка залікового звіту

## 6 Форми навчання

Навчальне навантаження модуля складається з аудиторної та самостійної роботи. Аудиторна робота включає 4 лекції, 3 лабораторні роботи (наскрізна структура завдань) та заключний практичний семінар із обговоренням розробленої гри.

Самостійна робота студентів передбачає підготовку до аудиторних занять і лабораторних робіт, а також оформлення залікового звіту. Підготовка до поточних аудиторних занять є аналіз літератури, електронних матеріалів з інтернету по темам лекцій і лабораторних робот, підготовка до проміжних та підсумкового тестів.

Лабораторні роботи 1-3 модуля орієнтовані на послідовне виконання наскрізного завдання та мають на меті відпрацювання навичок і вмінь розробки мобільного ігрового додатку для ОС Android.

Практичний семінар – заняття, на якому студент (або група студентів) презентує гру, що була розроблена. Інші студенти оцінюють результат,

виступаючи у ролі тестувальників та перших гравців. За результатами обговорення на семінарі до ігрової механіки вносяться зміни, що дозволять грі стати більш привабливою для широкої аудиторії.

Заліковий звіт – докладний опис та презентація графічного інтерфейсу мобільного ігрового додатку для Android, розробленого в середовищі Unity при виконанні лабораторних робіт 1-3 з урахуванням виправлення зауважень, що були надані аудиторією на практичному семінарі.

## 7 Порядок проведення атестації

Загальний принцип оцінювання підсумкових знань студента з курсу «Інструментальна підтримка розробки комп’ютерних ігрових додатків» полягає в тестуванні студентів на лекціях, оцінці поточної практичної роботи студента у навчальному семестрі на лабораторних роботах та оцінки контрольного заходу у формі іспиту, за результатами яких студент має сумарну оцінку в балах.

За результатами вивчення кожного змістового модуля передбачено оцінка виконання залікового завдання. Оцінка включає: результати тестування студентів на лекціях, результати поточного опитування при виконання лабораторних робіт модуля, оцінку за виконання залікового завдання. Сумарно оцінка кожного змістового модуля не може бути більш ніж 20 балів. Максимальна оцінка іспиту 40 балів.

Оцінка результатів вивчення дисципліни в цілому:

Поточне тестування	Балів
Змістовний модуль 1	до 20
<b>Змістовний модуль 2</b>	<b>до 20</b>
Змістовний модуль 3	до 20
Іспит	до 40
Разом	до 100

Графік проведення поточного оцінювання модуля 2:

Номер тижня	Оцінювання
7	Оцінка виконання лабораторної роботи 1
8	Оцінка виконання лабораторної роботи 2
9	Оцінка виконання лабораторної роботи 3
10	Оцінка участі у практичному семінарі
11	Оцінка залікового звіту і захисту проекту

## **Представлення звіту щодо виконання лабораторних робіт.**

При представленні звіту з лабораторних робіт необхідно виконувати наступне. При умові виконання кожної окремої лабораторної роботи єдиний звіт надається студентом на 11 тижні семестру. Продовження терміну можливе лише при наявності поважної причини, передбаченої порядком навчання студентів у вищій школі. При цьому:

- електронна версія єдиного звіту надається викладачеві через інтернет на початку 11 тижня;
- під час практичного семінару на 10 тижні студент демонструє закінчений варіант гри.
- за кожний день прострочки представлення та здачі єдиного звіту по модулю 2 знімається 1 бал (не більш ніж 5 днів – 5 балів)

## **Метод оцінки змістового модуля**

Кількість балів в загальній оцінці змістового модулю відповідає наступному:

Виконання лабораторної роботи 1	максимально 4 бали.
Виконання лабораторної роботи 2	максимально 4 бали.
Виконання лабораторної роботи 3	максимально 4 бали.
Результати участі в практичному семінарі	максимально 4 бали.
Оцінка залікового звіту і захисту проекту	максимально 4 бали.

Усі набрані бали підсумовуються (максимально 20 балів), штрафні бали за запізнення в представленні єдиного звіту (максимально мінус 5 балів) віднімаються. Сумарна оцінка (від 0 до 20 балів) є індивідуальною оцінкою студента із засвоєння змістового модуля 2.

## **Метод оцінки дисципліни в цілому**

Оцінки студентів за результатами вивчення змістовних модулів 1-3 підсумовуються. Далі додається оцінка іспиту (максимально 40 балів) та розраховується сумарна оцінка студента в балах за дисципліною. Сумарна оцінка в балах, переводиться за нижченнаведеною шкалою оцінювання в національну:

<i>Сума балів за всі види навчальної діяльності</i>	<i>Оцінка за національною шкалою</i>
90-100	Відмінно
82-89	Добре
74-81	Добре
64-73	Задовільно
60-63	Задовільно
35-59	не зараховано з можливістю повторного складання
0-34	не зараховано з обов'язковим повторним вивченням дисципліни

## 8 Зворотній зв'язок

Інформація щодо результатів тестування, виконання лабораторних робіт та загальна оцінка змістового модулю надається кожному студенту як індивідуально, так і всієї групи в цілому.

Інформація щодо результатів тестування надається студентам на 11-му тижні навчання. Інформація щодо оцінки виконання лабораторної роботи надається студентові під час заняття. Інформація щодо оцінки змістового модуля в цілому надається студентам на 11 тижні навчання.

Контактні дані для online-допомоги та консультування:

Викладачі:

доцент, к.т.н. Цололо С.О., [sergii.tsololo@donntu.edu.ua](mailto:sergii.tsololo@donntu.edu.ua),  
ст. викл. Дікова Ю.Л., [yuliia.dikova@donntu.edu.ua](mailto:yuliia.dikova@donntu.edu.ua)

## 9 Викладацький склад та допоміжні джерела

**Обов'язки викладачів:**

- подача матеріалів модуля згідно з програмою;
- оцінка результатів тестування та виконання лабораторних робіт;
- оцінка підсумкового звіту.

**Обов'язки координатора дисципліни:**

- планування та внесення змін до модуль;
- координація і управління професорсько-викладацьким складом;
- координація проведення тестування, лабораторних робіт та іспиту.

### **Обов'язки допоміжного персоналу:**

Допоміжний персонал здійснює підготовку комп'ютерної техніки до виконання лабораторних робіт студентами та надає технічну підтримку студентів під час виконання лабораторних робіт.

### **Контактні дані викладачів:**

доцент, к.т.н. Цололо С.О., [sergii.tsololo@donntu.edu.ua](mailto:sergii.tsololo@donntu.edu.ua),  
ст. викл. Дікова Ю.Л., [yuliia.dikova@donntu.edu.ua](mailto:yuliia.dikova@donntu.edu.ua)

### **Контактні дані куратора:**

професор, д.т.н. Башков Є.О., [eabashkov@i.ua](mailto:eabashkov@i.ua)

## **10 Навчальна програма і матеріали**

**10.1 Тема 1. Архітектура і базові принципи проектування додатків для платформи операційної системи Android. Формування головних персонажів, базової механіки та засобів управління ігровим процесом**

### **10.1.1 Мета та очікувані результати**

Ознайомити студентів з архітектурою та основними етапами створення додатків для платформи Android, а також розглянути процес створення загальної концепції мобільної гри, формування основних персонажів та елементів ігрової механіки.

Очікуваними результатами є:

- розробка простого тестового додатку для платформи Android;
- концепція мобільного ігрового додатку із проробкою ідеї, основних персонажів та елементів гри.

### **10.1.2 Лекція**

Лекція знайомить із загальною архітектурою мобільних додатків та їх компонентів, а також пропонує аналіз переваг, особливостей та можливостей операційної системи Android та сучасних підходів до розробки додатків для цієї платформи.

### **Мета лекції:**

- ознайомитися із загальною архітектурою мобільних додатків Android та їх компонентів;

- проаналізувати переваги, особливості та можливості різних версій платформи Android;
- розглянути інтегроване середовище розробки Android Studio;
- ознайомитися з файловою структурою додатку та процесом побудування проекту;
- розглянути файл маніфесту та можливості системи автоматичної зборки Gradle;
- проаналізувати життєвий цикл додатку, а також розглянути стеки діяльностей та засоби відстеження змін стану діяльності;
- ознайомитися з можливостями інструменту Logcat, що призначений для відлагодження Android-проектів.

Основні результати лекції відповідають вище передбаченим цілям.

#### **10.1.3 Лабораторна робота 1. Формування головних персонажів, базової механіки та засобів управління ігровим процесом**

Лабораторна робота орієнтована на практичне ознайомлення студентів з виділенням ідеї, основних характеристик та етапів створення мобільного ігрового додатку.

##### **Мета лабораторної роботи:**

- ознайомити студентів із особливостями створення Unity-проекту, що буде орієнтований на мобільну платформу ОС Android;
- розглянути процес додавання та налаштування спрайтів до проекту, принципів побудування на їх основі основних ігрових елементів;
- ознайомитися із принципами створення скриптів на мові програмування C#, їх структурою та засобами звернення до ігрових об'єктів;
- набути навичок із налаштування фізичної моделі поведінки ігрових об'єктів у вигляді колайдерів різного типу;
- ознайомитися із принципами побудування сенсорного інтерфейсу користувача у вигляді екранних кнопок та завдання їх поведінки за допомогою скриптів;
- навчитися керувати камерою та правильно організовувати ігрові об'єкти в групі;
- розглянути особливості створення APK-файлу ігрового додатку для його тестування на пристройі;

Відповідно до наскрізного завдання студент створює основного персонажа та поверхню (платформу) на основі спрайтів, реалізує їх поведінку у рамках базової механіки 2D-платформеру та додає до гри можливість сенсорного керування за допомогою екранних кнопок.

У разі успішного виконання лабораторної роботи студент буде здатен розробити першу найпростішу версію мобільного ігрового додатку у жанрі 2D-платформеру із базовою фізичною моделлю та сенсорним керуванням.

Успішне засвоєння матеріалів та виконання лабораторної роботи сприяє формування у студента наступних соціальних навичок та вмінь:

- дотримуватися регламентних рамок та строків виконання робіт.
- вміння роботи у команді, на досягнення загального результату;
- обирати моделі та стилі спілкування;
- обґрунтувати спосіб діяльності при виконанні практичних завдань, що мають такі форми прояву:
- вміння планувати свій час та діяльність групи;
- дотримання регламентних рамок;
- дисциплінованість та зібраність;
- вміння формулювати та доносити до співрозмовника свої думки;
- дотримуватися мотиваційних принципів в роботі та навчанні;
- дотримуватись етичних норм ділового та партнерського спілкування.

#### 10.1.4 Методичні матеріали та вказівки

Методичні матеріали та вказівки доступні за посиланням <http://89.185.3.253:9090/login> (login: *guest*, pass: *gamehub*), директорія *03\_M\_Game Applications Development for OS Android Platform*.

Опорний конспект лекції № 1 дивись

*03\_M\_Lec\_1 Game Applications Development for OS Android Platform.pdf*

Методичні вказівки щодо виконання лабораторної роботи дивись

*03\_M\_Lab Game Applications Development for OS Android Platform.pdf*

#### 10.1.5 Питання, що виносяться на іспит.

1. Архітектура платформи Android з точки зору розробника.
2. Особливості версій платформи Android.
3. Інтегроване середовище розробки Android Studio.
4. Життєвий цикл додатку Android.

5. Файлова структура додатку.
6. Файл маніфесту та процес побудування проекту.
7. Системи автоматичної зборки Gradle.
8. Стеки діяльностей та відстеження змін стану діяльності.
9. Відлагодження Android-проектів за допомогою Logcat.
10. Аналіз жанру, тематики та цільової аудиторії ігрового додатку.
11. Формулювання головних складових та відмінних рис гри.
12. Опис послідовності дій у грі («core loop»).
13. Оцінка економіки та соціальних складових гри.
14. Аналіз ігрового процесу та підтримка гри після випуску.

## 10.2 Тема 2. Огляд основних властивостей платформи Android для розробки ігрових додатків. Програмування поведінки і взаємодії ігрових об'єктів, фізичних процесів та основних ефектів.

### 10.2.1 Мета та очікувані результати

Ознайомити студентів з особливостями платформи Android, які використовуються при створенні ігрових додатків. Сформувати у студентів знання щодо проектування основних елементів інтерфейсу ігрового додатку та реалізації засобів контролю над ігровим процесом у відповідності до стандартних схем управління додатками на платформі Android.

### 10.2.2 Лекція

Лекція знайомить студентів з основними засобами управління ресурсами мобільного додатку. Виконується огляд основних засобів операційної системи для інтерфейсу користувача, виклик стандартних діалогів і сервісів та створення власних елементів управління. Розглядається навігація між різними діяльностями (activity) додатку. Наводяться приклади організації зберігання даних у вигляді внутрішніх сховищ операційної системи з використанням постачальників даних, а також створення власних сховищ даних.

#### Мета лекції:

- розглянути основні елементи інтерфейсу додатку та їх розміщення в макетах відображення;

- ознайомитися з програмним створенням елементів інтерфейсу та використанням ідентифікаторів в коді програми для маніпуляції з елементами інтерфейсу;
- розглянути способи використання зображень, меню, макетів відображення; класів відображення та його підкласів, елементів Toast і Snackbar;
- використання різних кольорів, розмірів, одиниць вимірювання, стилів та тем, атрибутів розміщення елементів та фрагментів.
- розглянути взаємодію між діяльностям та фрагментами, а також використання моделей та адаптерів для взаємодії з даними;
- використання контент-провайдерів до баз даних та файлів для створювання постійних сховищ даних додатку.
- розглянути основні способи реалізації декількох процесів і потоків (threads), а також життєвий цикл процесу і потоків і використання AsyncTask.

Основні результати лекції відповідають вище передбаченим цілям.

#### 10.2.3 Лабораторна робота 2. Програмування поведінки і взаємодії ігрових об'єктів, фізичних процесів та основних ефектів

Лабораторна робота орієнтована на оволодіння студентами навичок проектування інтерфейсів ігрових додатків на платформі Android.

##### Мета лабораторної роботи:

- навчитися контролювати положення об'єктів відповідно до поверхні (платформи);
- ознайомитися із головними принципами реалізації базових трансформацій та використання Physics2D у скриптах;
- навчитися працювати із різними ігровими шарами та групувати об'єкти за ними відповідно до призначення;
- навчитися створювати шаблони (prefabs) та ефектори (effectors) для більш швидкого та зручного створення ігрових рівнів;
- ознайомитися із створенням різних типів перепон і ворогів, що відповідають жанру гри, та навчитися програмувати їх поведінку;
- навчитися додавати до основних ігрових подій ефекти, побудовані на основі системи часток.

Відповідно до наскрізного завдання студент створює більш досконалу версію ігрового додатку із підтримкою фізики, ігрових перепон та ворогів різних типів, а також ефектів, що супроводжують ключові ігрові події.

У разі успішного виконання лабораторної роботи студент буде вміти програмувати поведінку та взаємодію основних ігрових об'єктів, створювати ігрові шаблони та використовувати систему часто для створення ігрових ефектів.

Успішне засвоєння матеріалів та виконання лабораторної роботи сприяє формування у студента наступних соціальних навичок та вмінь:

- реалізувати свою діяльність творчо, проявляти пізнавальну активність;
- критично ставитися до результатів групової та особистої роботи, вміти проводити самооцінку та групове оцінювання виконаних робіт;
- своєчасно та самостійно виконувати пошук можливих помилок, знаходити способи їх усунення та попередження, що мають такі форми прояву:
- прагнення до самореалізації та нових форм діяльності;
- висловлення нестандартних ідей;
- прояви винахідливості, кмітливості, допитливості;
- бажання пробувати нове, йти на ризик при прийнятті рішень;
- бути здатним до прийняття та узгодження колективного рішення;
- вміння створювати творчу атмосферу;
- критичне ставлення до результатів групової та особистої роботи;
- використовувати аналіз помилок, допущених у особистій чи груповій роботі, для знаходження правильного вирішення завдання;
- використовувати прийоми ділового спілкування, раніше отримані знання як основу для засвоєння нових знань.

#### 10.2.4 Методичні матеріали та вказівки

Методичні матеріали та вказівки доступні за посиланням <http://89.185.3.253:9090/login> (login: guest, pass: gamehub), директорія *03\_M\_Game Applications Development for OS Android Platform*.

Опорний конспект лекції № 2 дивись

*03\_M\_Lec\_2 Game Applications Development for OS Android Platform.pdf*

Методичні вказівки щодо виконання лабораторної роботи дивись

*03\_M\_Lab Game Applications Development for OS Android Platform.pdf*

#### **10.2.5 Питання, що виносяться на іспит.**

1. Основні елементи інтерфейсу Android-додатку.
2. Програмне створення елементів інтерфейс.
3. Способи реалізації меню та макетів відображення.
4. Класи і підкласи відображення.
5. Використання елементів Toast і Snackbar.
6. Стилі, теми, атрибути розміщення елементів та фрагментів.
7. Взаємодія між діяльностям та фрагментами.
8. Використання моделей та адаптерів для взаємодії з даними.
9. Використання контент-провайдерів до баз даних та файлів.
10. Основні способи реалізації декількох процесів і потоків.
11. Життєвий цикл процесу і потоків, використання AsyncTask.
12. Огляд сучасних засобів проектування інтерфейсів
13. Способи взаємодії користувача з Android-додатками.
14. Основні режими типового ігрового додатку та їх призначення.

#### **10.3 Тема 3. Особливості створення мобільних ігрових додатків для платформи Android в середовищі Unity. Анімація персонажів, реалізація предметів і цілей, організація рівнів та переходу між ними**

##### **10.3.1 Мета та очікувані результати**

Ознайомити студентів з особливостями використання середовища Unity для створення мобільних ігрових додатків для платформи Android. Проаналізувати відмінності у створенні мобільного ігрового додатку, що мають бути враховані при заздалегідь відомій цільовій платформі розробки.

Сформувати знання щодо основних засобів анімації, програмування руху та реалізації освітлення персонажів і основних ігрових елементів. Розглянути типові анімації та рухи, що є стандартним при реалізації того чи іншого типового персонажу або ігрового елементу.

##### **10.3.2 Лекція**

Лекція знайомить студентів з процесом створення мобільного ігрового додатку для платформи Android в середовищі Unity. Розглядаються основи механізму скриптової анімації, інструменти дизайну рівнів, проблеми освітлення та використання набору камер, контроль та керування

персонажами на основі подій, особливості використання плагінів, типові проблеми Android-додатків на Unity.

**Мета лекції:**

Ознайомити студентів з особливостями Unity для мобільних платформ:

- початок роботи з Android SDK;
- імпорт та робота моделями і текстурами;
- основи механізму анімації Macanim;
- базові інструменти дизайну рівнів;
- освітлення в реальному часі та карти освітлення;
- особливості використання декількох камер;
- введення в події та делегати;
- розробка і настройка персонажів;
- використання сумісної програми з зупинкою (coroutines);
- контроль та керування діями персонажів;
- скрипти в анімації Macanim;
- основи створення ефектів часток;
- огляд та особливості використання плагінів;
- процес зведення і компіляція гри;
- специфічні для Android проблеми.

Основні результати лекції відповідають вище передбаченим цілям.

### 10.3.3 Лабораторна робота 3. Анімація персонажів, реалізація предметів і цілей, організація рівнів та переходу між ними

Лабораторна робота орієнтована на оволодіння студентами навичок програмування руху персонажів і основних ігрових елементів та реалізації різних способів освітлення в середовищі Unity.

**Мета лабораторної роботи:**

- навчитися додавати до гри предмети, що збираються (collectibles), та реалізовувати відповідні скрипти підрахунку таких предметів;
- ознайомитися із засобами виведення інформації екран пристрою за допомогою елементів Unity.UI;
- навчитися створювати таймлайні анімації на основі декількох спрайтів;

- навчитися розробляти діаграми переходів між анімаціями за значенням окремого параметра, який контролюється відповідним скриптом;
- ознайомитися із механізмом створення декількох рівнів із використанням однакових шаблонів;
- навчитися організовувати переходи між рівнями при виконанні ігрової умови (досягнення певної позиції або збирання певної кількості предметів);
- ознайомитися із додатковими налаштуваннями при генерації мобільного додатку, що пов’язані із параметрами екрану.

Відповідно до наскрізного завдання студент реалізує анімацію основних дій головного персонажу, додає до гри предмети, що можна збирати, та організовує переход між різними ігровими рівнями за умови виконання заданих ігрових досягнень.

У разі успішного виконання лабораторної роботи студент оволодіє знаннями щодо створення анімації персонажів та ігрових об’єктів, реалізації функції збирання предметів, розбудови декількох ігрових рівнів та переходів між ними.

Успішне засвоєння матеріалів та виконання лабораторної роботи сприяє формування у студента наступних соціальних навичок та вмінь:

- критично ставитися до результатів групової та особистої роботи, вміти проводити самооцінку та групове оцінювання виконаних робіт;
- проявляти навички лідерства та вміння, необхідні для виконання групової та командної роботи;

що мають такі форми прояву:

- критичне ставлення до результатів групової та особистої роботи;
- мати навички в усуненні та запобіганні конфліктів;
- демонструвати доброзичливе та поважливе відношення до пропозицій учасників спільног проєкту;
- формування елементів самоконтролю і самооцінки діяльності
- прагнення вирішувати поточні проблеми самостійно, або із залученням членів команди;
- глибокий аналіз проблем і пошук нових можливостей;
- націленість на успіх, прагнення отримати найкращі результати;
- прагнення до нових форм діяльності, підприємливість;

- керівна роль в групових діях.

#### 10.3.4 Методичні матеріали та вказівки

Методичні матеріали та вказівки доступні за посиланням <http://89.185.3.253:9090/login> (login: *guest*, pass: *gamehub*), директорія *03\_M\_Game Applications Development for OS Android Platform*.

Опорний конспект лекції № 3 дивись

*03\_M\_Lec\_3 Game Applications Development for OS Android Platform.pdf*

Методичні вказівки щодо виконання лабораторної роботи дивись

*03\_M\_Lab Game Applications Development for OS Android Platform.pdf*

#### 10.3.5 Питання, що виносяться на іспит.

1. Основи роботи моделями і текстурами.
2. Базові інструменти дизайну рівнів.
3. Розробка, настройка і керування персонажів.
4. Механізми анімації та скрипти.
5. Освітлення в реальному часі та карти освітлення.
6. Особливості використання камер.
7. Використання подій, делегатів та співпрограми.
8. Зведення, компіляція та тестування гри.
9. Специфічні для Android проблеми Unity-додатків.
10. Класифікація типових анімацій.
11. Використання фізичних движків.

### 10.4 Тема 4. Публікація та засоби просування ігрового додатку, соціальна складова, оцінка шляхів ефективної монетизації. Тестування ігрового додатку, балансування ігрової механіки, підготовка додатку до релізу та публікації.

#### 10.4.1 Мета та очікувані результати

Ознайомити студентів з основними етапами публікації ігрового додатку та різними шляхами його реклами та просування. Навчити студентів правильно оцінювати та виділяти найбільш прийнятні шляхи монетизації ігрового додатку в залежності від та особливостей реалізації ігрового процесу.

Ознайомити студентів з особливостями поточного та кінцевого тестування ігрового додатку, способами балансування ігрової механіки та оптимізації ігрового процесу на основі досвіду користувачів.

#### 10.4.2 Лекція

Лекція знайомить з основними елементами організації магазину додатків в Google Play та процесом підготовки та публікації додатку в магазині. Розглядаються різні за підходом та часом способи отримання прибутку з додатку (монетизація), в тому числі завдяки реалізації соціальної складової додатку. Наводяться основні етапи процесу реклами та просування ігрового додатку із урахуванням платформи реалізації.

##### **Мета лекції:**

Ознайомити студентів з:

- основними категоріями додатків в Google Play;
- експортом додатку та особливостями створення сертифіката;
- процесом публікації додатку в Google Play;
- способами оновлення існуючого додатку;
- типовими способами монетизації додатків;
- реалізацією соціальної складової додатку;
- способами організації багатокористувальських онлайн-додатків;
- етапами організації реклами та просування додатку.

Основні результати лекції відповідають вище передбаченим цілям.

#### 10.4.3 Заключний практичний семінар. Тестування ігрового додатку, балансування ігрової механіки, підготовка додатку до релізу та публікації.

Заключний практичний семінар орієнтований на обговорення створених один одним додатків з метою корегування основних параметрів ігрової механіки, а також підготовки додатку до публікації.

##### **Мета практичного семінару:**

- оцінити правильність визначення концепції ігрового додатку, відповідності обраному жанри та коректності відображення основних графічних елементів;
- оцінити правильність визначення поведінки та взаємодії ігрових об'єктів відповідно до відомих принципів ігрової механіки;

- оцінити як саме були реалізовані особливості реалізації ігрових об'єктів та механіки відповідно до того, що проект є саме мобільним ігровим додатком;
- оцінити привабливість проекту та надати рекомендації щодо способів подальшого розвитку розробленого мобільного ігрового додатку.

За результатами семінару студенти навчаються роботи ефективно та конструктивно оцінювати ігрові додатки, виявляти недоліки та помилки, що були допущені на усіх етапах розробки від ідеї до кінцевого тестування та публікації, балансувати ігрову механіку відповідно до зауважень гравців.

Успішне засвоєння матеріалів та виконання лабораторної роботи сприяє формування у студента наступних соціальних навичок та вмінь:

- критично ставитися до результатів групової та особистої роботи, вміти проводити самооцінку та групове оцінювання виконаних робіт.
- реалізовувати вербальні та невербальні способи спілкування;
- приймати та підтримувати правила ділової поведінки, усної комунікації, партнерських відносин

що мають такі форми прояву:

- ефективна робота в команді та вміння застосувати відповідну моменту схему спілкування з партнерами;
- здатність узгоджувати внутрішні бажання з колективними потребами;
- вміння формулювати та доносити до співрозмовника свої думки;
- критичне ставлення до результатів групової та особистої роботи;
- мати навички в усуненні та запобіганні конфліктів;
- демонструвати доброзичливе та поважливе відношення до пропозицій учасників спільного проекту;
- формування елементів самоконтролю і самооцінки при виконанні діяльності
- вміти побудувати довірче, взаємовигідне спілкування, підтримувати реалізацію групових планів та дій;
- взаєморозуміння, побудова щиріх відносин з партнерами;
- вміння ясно і конкретно висловлювати думки, слухати та розуміти співрозмовника, спонукати партнерів на подальші ефективні дії та досягнення запланованих результатів.

#### 10.4.4 Методичні матеріали та вказівки

Методичні матеріали та вказівки доступні за посиланням <http://89.185.3.253:9090/login> (login: *guest*, pass: *gamehub*), директорія *03\_M\_Game Applications Development for OS Android Platform*.

Опорний конспект лекції № 4 дивись

*03\_M\_Lec\_4 Game Applications Development for OS Android Platform.pdf*

Методичні вказівки щодо підготовки до практичного семінару дивись

*03\_M\_Lab Game Applications Development for OS Android Platform.pdf*

#### 10.4.5 Питання, що виносяться на іспит.

1. Основні типи тестування додатків.
2. Інструменти ручного та автоматизованого тестування.
3. Основи роботи з системами версій.
4. Категорії та вимоги до додатків в Google Play.
5. Особливості процесу публікації додатку в Google Play.
6. Типові способами монетизації додатків.
7. Реалізація соціальної складової додатку.
8. Організація реклами та просування додатку.

### 11 Рекомендована література та інтернет-посилання

1. John Horton. *Android Game Programming by Example*. Packt Publishing, 2015. – 360 p.
2. James S. Cho. *The Beginner's Guide to Android Game Development*. Glasncvin Publishing, 2014. – 430 p.
3. Thomas Finnegan. *Learning Unity Android Game Development*. Packt Publishing, 2015. – 316 p.
4. Valera Cogut. *Unity 5 for Android Essentials*. Packt Publishing, 2015. – 180 p.
5. Unity, начало разработки под Android –  
<https://docs.unity3d.com/ru/530/Manual/android-GettingStarted.html>
6. Unity Android Tutorials – <https://nevzatarman.com/category/unity-android-tutorials/>.

**МЕТОДИЧНІ ВКАЗІВКИ  
ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ**

**Дисципліна: Інструментальна підтримка розробки комп'ютерних  
ігрових додатків**

**Модуль: Розробка ігрових додатків для ОС Android**

## 1 Вступ. Загальні положення

Комплекс лабораторних робот модуля «Розробка ігрових додатків для ОС Android» виконується відповідно [1], передбачує послідовне (step by step) виконання наскрізного завдання та має на меті відпрацювання навичок розробки мобільних ігрових додатків для ОС Android за допомогою пакета Unity.

До комплексу лабораторних робіт входить заключний практичний семінар – заняття, на якому студент презентує ігровий додаток, що було розроблено. При цьому інші студенти групи оцінюють результат, виконуючи роль тестувальників ігрового додатку. За результатами обговорення на семінарі сформулюються зміни, що мають бути внесені до остаточної версії ігрового додатку.

Заліковий звіт – надання докладного письмового звіту про виконання наскрізного завдання лабораторних робіт 1-3.

Самостійна робота студентів передбачає вивчення додаткової літератури [3, 5], підготовку до лабораторних робіт, а також підготовку та оформлення залікового звіту та презентації за результатами виконання лабораторних робіт.

### 1.1 Графік виконання лабораторних робіт

Номер тижня	Виконання	Оцінювання (максимальний бал)
7	Лабораторна робота №1	4
8	Лабораторна робота №2	4
9	Лабораторна робота №3	4
10	Презентація та обговорення отриманих результатів на заключному практичному семінарі	4
11	Захист залікового звіту	4
Максимальний сумарний бал		20

### 1.2 Оцінка виконання комплексу лабораторних робіт.

Критерії оцінювання виконанняожної лабораторної роботи наведені в додатку А.

Критерії оцінювання обговорення результатів на заключному практичному семінарі наведені в додатку Б.

Критерії оцінювання заключного звіту наведені в додатку В.

Інформація щодо оцінки змістового модуля в цілому надається студентам на 11 тижні навчання.

### 1.3 Контактні дані для on-line допомоги та консультування:

Викладачі :

- доцент, к.т.н. Ідололо С.О., [sergii.tsololo@donntu.edu.ua](mailto:sergii.tsololo@donntu.edu.ua)
- ст. викл. Дікова Ю.Л., [yuliia.dikova@donntu.edu.ua](mailto:yuliia.dikova@donntu.edu.ua)

## 2 Наскрізне завдання

Лабораторні роботи за модулем присвячені розробці мобільного ігрового додатку в жанрі 2D-платформер для ОС Android. Завдання до лабораторних робіт є наскрізним та відповідає усім ключовим етапам проектування, розробки, тестування та підготовки до публікації ігрового додатку.

Отже, в даному модулі процес розробки ігрового додатку розбивається на три основних етапи:

1. Формування головних персонажів, базової механіки та засобів управління ігровим процесом.
2. Програмування поведінки і взаємодії ігрових об'єктів, фізичних процесів та основних ефектів
3. Анімація персонажів, реалізація предметів і цілей, організація рівнів та переходів між ними.

Кожен етап розробки ігрового додатку відповідає одній лабораторній роботі, після виконання яких студент представляє свій додаток на підсумковому практичному семінарі.

Таким чином, головною метою і результатом виконання даного модуля є реалізація повного циклу розробки мобільного ігрового додатку.

## 3 Обладнання

Виконання лабораторних робіт передбачає створення мобільного ігрового додатку для ОС Android за допомогою середовища розробки Unity 2017.

Мінімальні вимоги до технічних характеристик персонального комп’ютера, що буде використаний при розробці:

- процесор Intel Dual Core, 64-bit;

- оперативна пам'ять 2GB;
- не менш ніж 5GB вільного простору на HDD;
- графічна карта сумісна з DX11;
- інтернет-з'єднання.

Вимоги до операційної системи

- Microsoft Windows 7 (Service Pack 1) або вище;
- 64-bit версія.

Вимоги до платформи:

- середа розробки Unity 2017;
- інтегроване середовище розробки програмного забезпечення Visual Studio 2015+;
- комплект розробника Java Development Kit (JDK) 8+;
- комплект розробника Android SDK (входить до Android Studio або встановлюється окремо).

## 4 Попередні підготовчі кроки до виконання лабораторних робіт

Для виконання лабораторних робіт необхідно оснастити робоче місце шляхом встановлення набору інструментальних засобів розробки програмного забезпечення.

Перш за все необхідно встановити середовище розробки Unity 2017, бо саме в ньому буде проходити основний цикл розробки мобільного ігрового додатку. Unity дозволяє створювати додатки, що працюють під різними операційними системами, ігровими консолями, мобільними пристроями, веб-додатками. Основними перевагами Unity є наявність візуальної середовища розробки, міжплатформеної підтримки і модульної системи компонентів.

Unity можна безкоштовно завантажити на офіційному сайті за посиланням <https://store.unity.com/download> (рис. 4.1). Точна версія середовища не має значення, але у методичних вказівках використано версію Unity 2017, тому саме вона є рекомендованою для встановлення.

Для редагування C#-скриптів в середовищі Unity 2017 необхідний редактор, у ролі якого найчастіше використовується Microsoft Visual Studio (MS VS). Якщо на робочому місці не встановлено жодної версії MS VS вище 2012, то необхідно завантажити і встановити з офіційного сайту версію Visual Studio Community за посиланням <https://www.visualstudio.com/ru/thank-you-downloading-visual-studio/?sku=Community&rel=15> (рис. 4.2).

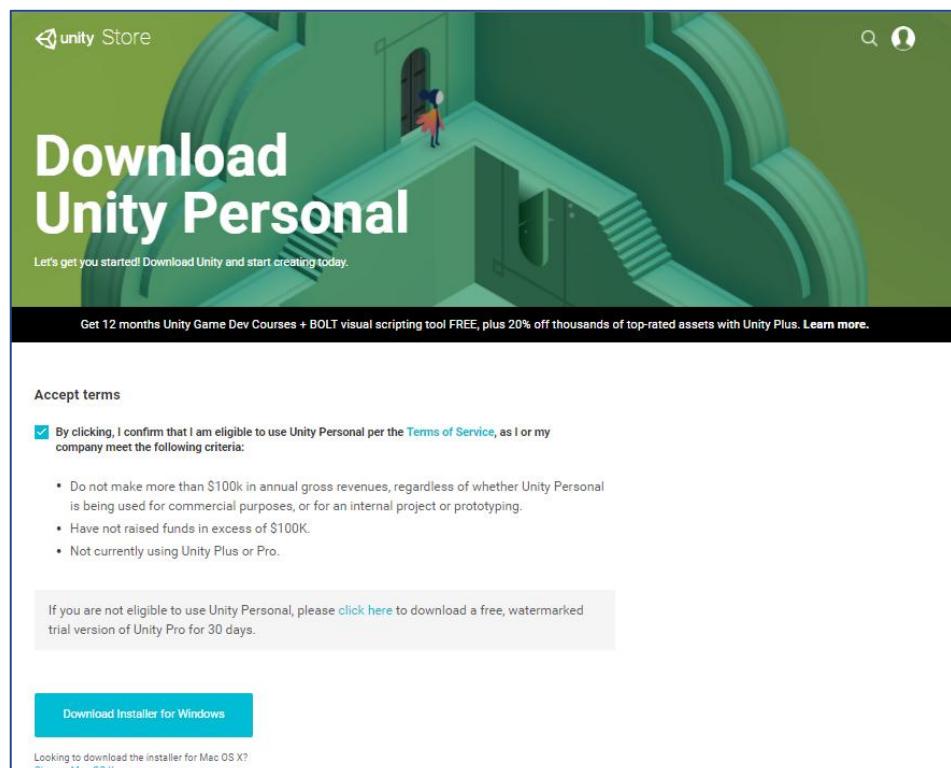


Рисунок 4.1 – Завантаження середовища Unity 2017

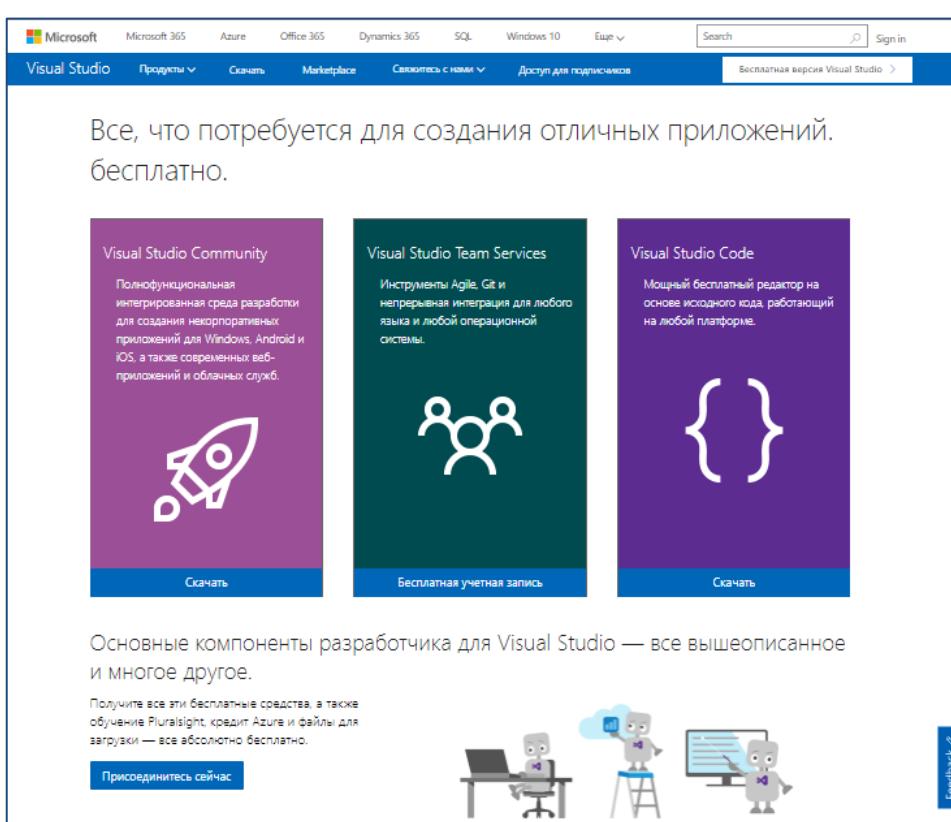


Рисунок 4.2 – Завантаження Visual Studio Community

Для компіляції проекту в мобільний додаток для ОС Android середовище Unity 2017 вимагає встановлення Java Development Kit (JDK) та Android SDK.

JDK – безкоштовно поширюваний комплекст розробника додатків на мові Java, що включає в себе компілятор Java (javac), стандартні бібліотеки класів Java, приклади, документацію, різні утиліти і виконавчу систему Java (JRE). Версія JDK 8 для операційної системи Windows може бути встановлена із

офіційного сайту за посиланням <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html> (рис. 4.3).

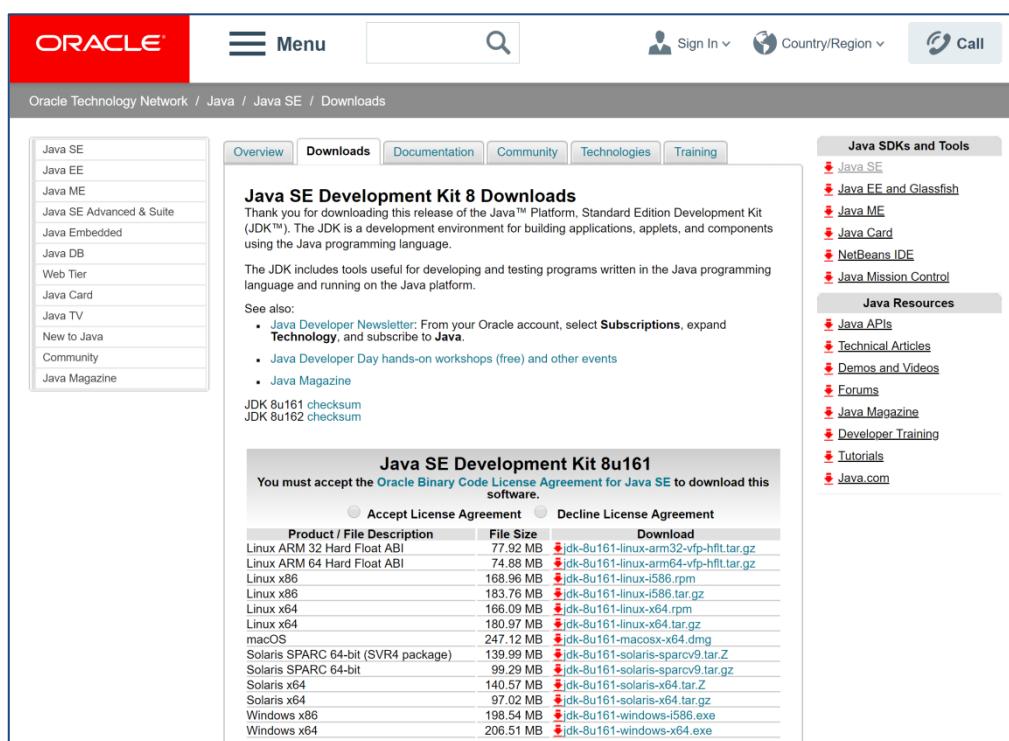


Рисунок 4.3 – Завантаження JDK

Android SDK – універсальний засіб розробки мобільних додатків для операційної системи Android. Відмінною рисою від звичайних редакторів для є наявність широких функціональних можливостей, що дозволяють запускати тестування і налагодження програм, оцінювати роботу програми в режимі сумісності з різними версіями ОС Андроїд і спостерігати результат в реальному часі. На даний час Android SDK є частиною інтегрованої системи розробки (IDE) Android Studio. Ця IDE заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains, та є офіційним засобом розробки Android-додатків.

Завантажити актуальну версію Android Studio можна із офіційного сайту за посиланням <https://developer.android.com/studio/index.html> (рис. 4.4).

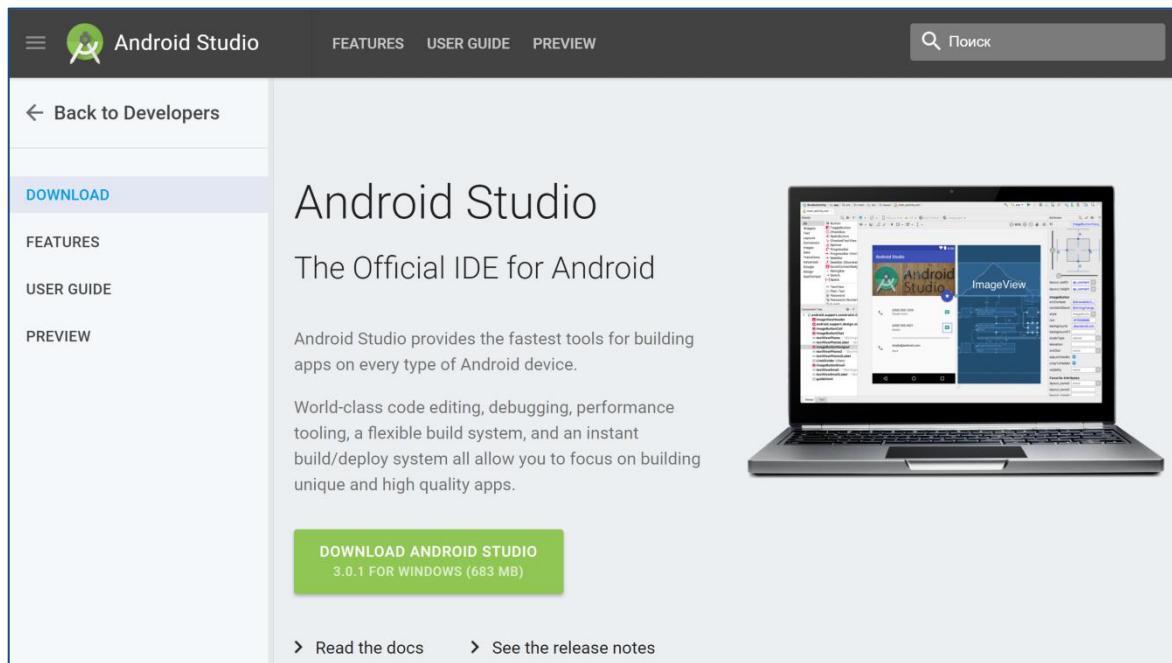


Рисунок 4.4 – Завантаження Android Studio

## 5 Лабораторна робота № 1. Формування головних персонажів, базової механіки та засобів керування ігровим процесом

Перша лабораторна робота орієнтована на ознайомлення студентів із основними принципами виділення та формування головних персонажів гри у жанрі 2D-платформеру та реалізації базової механіки відповідно до запланованих засобів керування ігровим процесом.

Мета лабораторної роботи:

- ознайомити студентів із особливостями створення Unity-проекту, що буде орієнтований на мобільну платформу ОС Android;
- розглянути процес додавання та налаштування спрайтів до проекту, принципів побудування на їх основі основних ігрових елементів;
- ознайомитися із принципами створення скриптів на мові програмування C#, їх структурою та засобами звернення до ігрових об’єктів;
- набути навичок із налаштування фізичної моделі поведінки ігрових об’єктів у вигляді колайдерів різного типу;

- ознайомитися із принципами побудування сенсорного інтерфейсу користувача у вигляді екранних кнопок та завдання їх поведінки за допомогою скриптів;
- навчитися керувати камерою та правильно організовувати ігрові об'єкти в групі;
- розглянути особливості створення APK-файлу ігрового додатку для його тестування на пристрой.

Відповідно до наскрізного завдання студент створює основного персонажа та поверхню (платформу) на основі спрайтів, реалізує їх поведінку у рамках базової механіки 2D-платформеру та додає до гри можливість сенсорного керування за допомогою екранних кнопок.

У разі успішного виконання лабораторної роботи студент буде здатен розробити першу найпростішу версію мобільного ігрового додатку у жанрі 2D-платформеру із базовою фізичною моделлю та сенсорним керуванням.

Успішне засвоєння матеріалів та виконання лабораторної роботи сприяє формування у студента наступних соціальних навичок та вмінь: вміння дотримуватися регламентних рамок, бажання постійно вчитися, оцінювати строки проведення та виконання роботи. Форми прояву: бажання пробувати щось нове, прагнення вирішувати поточні проблеми самостійно, йти на розумний ризик при прийнятті рішень, елементи самоконтролю і самооцінки при виконанні та оцінки роботи, вміння планувати свій час.

## 5.1 Завдання до лабораторної роботи

1. Створити новий проект та додати до нього усі спрайти, що необхідні на початковому етапі.
2. На базі спрайтів створити основні ігрові об'єкти на задати їх поведінку за допомогою колайдерів із блоку «Physics 2D».
3. Розробити скрипт керування персонажем за допомогою кнопок клавіатури
4. Створити шар із фоновим зображенням та налаштувати камеру, прив'язавши її до персонажа.
5. Додати до проекту блок сенсорних кнопок та налаштувати їх поведінку за допомогою скриптів.
6. Створити APK-файл та протестувати додаток на реальному пристрой.

## 5.2 Підготовка до лабораторної роботи №1

Підготовка до виконання лабораторної роботи № 1 включає етапи:

1. Ознайомитися із головними принципами побудування ігрових додатків жанру 2D-платформеру.
2. Ознайомитись з методичними вказівками до виконання лабораторної роботи (див. 5.4).

## 5.3 Контрольні питання і попередні матеріали для допуску до лабораторної роботи № 1

### 5.3.1 Контрольні питання

1. Вкажіть основні особливості ігрових додатків, що відносяться до жанру 2D-платформеру?
2. Вкажіть основні інструментальні засоби, які необхідні для створення додатків у середовищі Unity?
3. Які додаткові компоненти мають бути встановлені для розробки саме мобільних ігрових додатків у середовищі Unity?
4. Наведіть основні вимоги до спрайтів, що призначені для створення ігрових об'єктів (персонаж, поверхня тощо).
5. Вкажіть структуру в основні архітектурні елементи додатків для ОС Android
6. Що таке Android Runtime і в чому його відмінність від Dalvik?
7. Назвіть основні компоненти Android-додатку.
8. Навіщо використовується файл AndroidManifest.xml?
9. Які три основні стани є в Операції (Activity), як реалізований життєвий цикл та перехід між станами?
10. Вкажіть призначення Служб (Services) та опишіть їх життєвий цикл.
11. Для чого використовуються постачальники контенту (Content Provider) та які з системних провайдерів вам відомі?
12. Вкажіть призначення та особливості налаштування приймачей широкомовних повідомлень (Broadcast receiver).

### 5.3.2 Попередні матеріали

1. Спрайт головного персонажу.
2. Спрайт поверхні (платформи).
3. Зображення екранних кнопок (вліво, вправо).
4. Зображення для фонового шару.

## 5.4 Методичні вказівки до виконання лабораторної роботи № 1

### 5.4.1 Створення проекту

Після завантаження та встановлення Unity, Visual Studio, JDK та Android Studio необхідно запустити Unity та натиснути «New» для створення нового проекту. Після цього відкривається нова сторінка, на якій вказується назва нового проекту та каталог для збереження файлів (рис. 5.1). Okрім цього, розробнику необхідно обрати тип проекту – 3D або 2D. У нашому випадку буде розроблятися 2D-платформер, отже оберемо «2D» та натиснемо «Create Project».

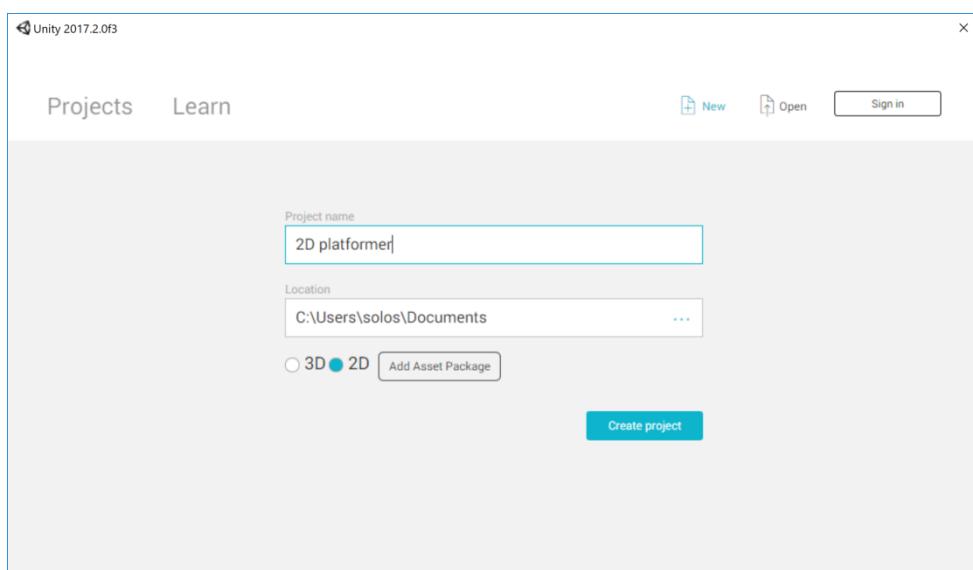


Рисунок 5.1 – Вікно створення створення проекту

Після цього на екрані буде відображене пусте робоче середовище Unity (рис. 5.2), яке складається із декількох вікон. Їх положення залежить від налаштувань користувача. Для керування проектом використовується відповідно вікно «Project», що містить усі файли проекту із групуванням по папках.

### 5.4.2 Додавання спрайтів

Оберіть каталог «Assets» – його вміст відобразиться праворуч. За допомогою правої кнопки миші оберіть пункт контекстного меню «Create > Folder» та створіть папку «Sprites». Вона буде містити спрайти (зображення), необхідні для гри.

Для побудови найпростішого платформера нам знадобляться два основних спрайти – поверхня («земля») та власне персонаж (рис. 5.3). Спрайти можна розробити у будь-якому графічному редакторі, після чого необхідно помістити їх до папки «Sprites».

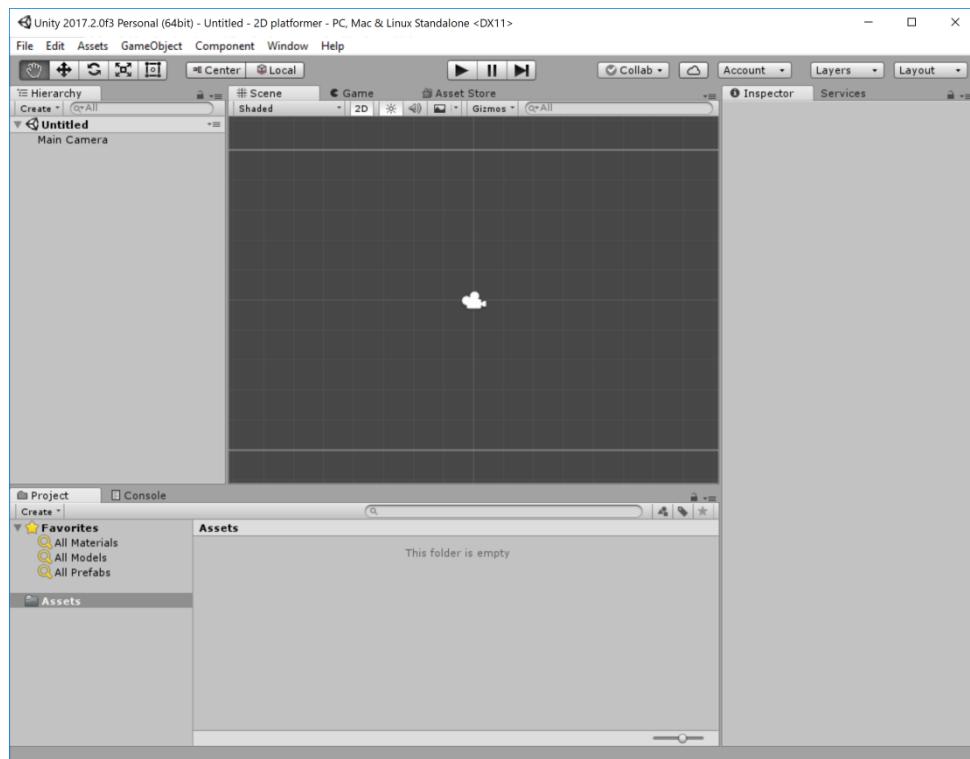


Рисунок 5.2 – Вікно пустого проекту



Рисунок 5.3 – Спрайти для проекту

Далі, для того щоб спрайти стали частиною гри, необхідно перетягнути їх до панелі «Scene» (Сцена). Ця панель призначена для розміщення та організації усіх елементів гри. Її також можна прокручувати, утримуючи «Alt» та переміщаючи мишею. Таким чином, по суті "сцена" – це рівень гри,

хоча в майбутньому це може також бути сторінкою меню або іншим екраном у грі. При цьому панель "Гра" дозволяє дізнатися, яке представлення гри камера відобразить на початку ігрового рівня. Для виклику достатньо натиснути кнопку «Play» (рис. 5.4).

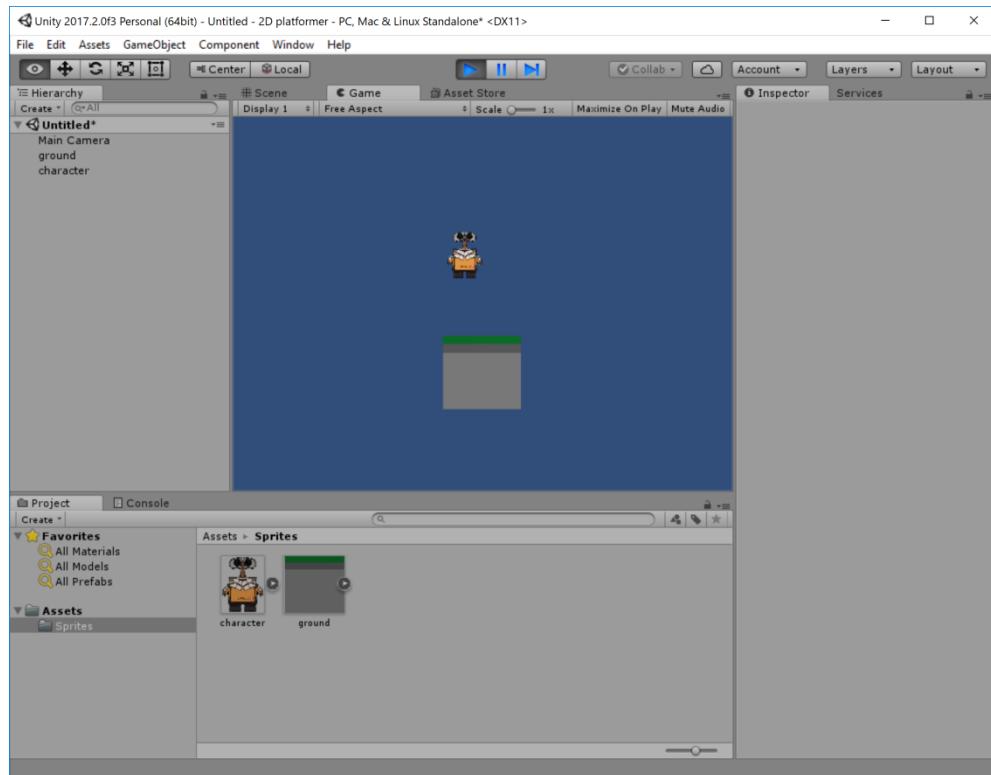


Рисунок 5.4 – Розміщення спрайтів у проекті

### 5.4.3 Додавання фізики

На панелі «Scene» оберіть за допомогою миші елемент (далі будемо називати його тайлом), що представляє поверхню («землю»). Це приведе до того, що атрибути об'єкту відобразяться у вікні «Inspector». Воно призначено саме для налаштування різноманітних параметрів об'єкту, таких як розмір, кут тощо (рис. 5.5).

Далі натисніть кнопку «Add Component» та оберіть «Physics 2D > Box Collider 2D». Візуально це добавить до об'єкту тонку зелену рамку, функціонально об'єкт почне визначати колізії – саме у тих межах, що задаються рамкою. Якщо в вас у якості поверхні використовується не прямокутний, а більш складний об'єкт, то можна використати «Edge Collider» для задання більш складної форми.

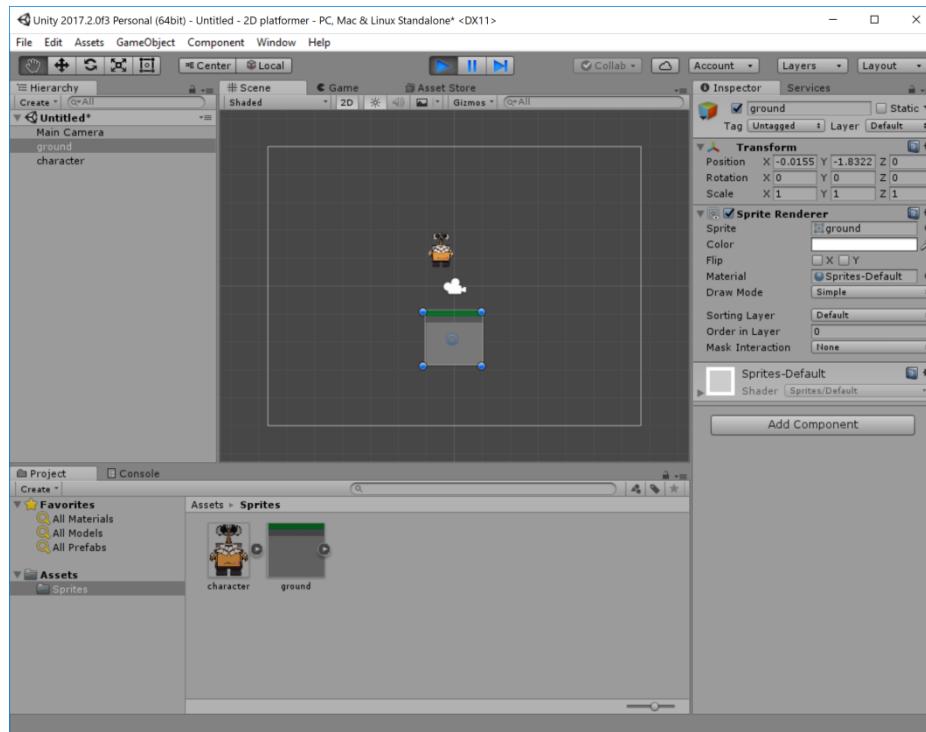


Рисунок 5.5 – Зміна атрибутів об’єкту

Далі теж саме необхідно зробити для об’єкту персонажу, причому вибір типу колайдеру заснований на формі спрайта персонажа. Якщо він має прямокутну форму, то обираємо «Box Collider 2D», в іншому випадку – «Edge Collider» або «Polygon Collider».

Отже після проведених дій обидва об’єкти стають «твірдими» та контролюють зіткнення, але гравітація є повністю відсутньою. Для виправлення цього необхідно обрати об’єкт персонажу, натиснути «Add Component» та додати «Physics 2D > Rigidbody 2D» – це надасть об’єкту елементи гравітації. Щоб перевірити це достатньо натиснути «Play» – і при цьому персонаж «впаде» із свого поточного місця на поверхню.

#### 5.4.4 Додавання керування

Наступним етапом роботи над платформером є додавання елементів керування. Це вимагає використання деяких навичок нескладного програмування.

Перш за все необхідно створити нову дочірню папку «Scripts» в папці «Assets». Далі в цій папці необхідно за допомогою правої кнопки миші обрати «Create > C# Script» та назвати новий скрипт «Controls». Подвійний клік на скрипті призведе до відкриття вікна редактування в IDE Visual Studio (рис. 5.6).

Базова структура скрипту, що був згенерований, є дуже простою. Увесь код, описаний в методі “Start”, буде виконано в момент створення об’єкту – наприклад, при запуску гри чи окремого рівня. Код з методу “Update” виконується періодично під час виконання ігрового додатку кожен раз при оновлені сцени.

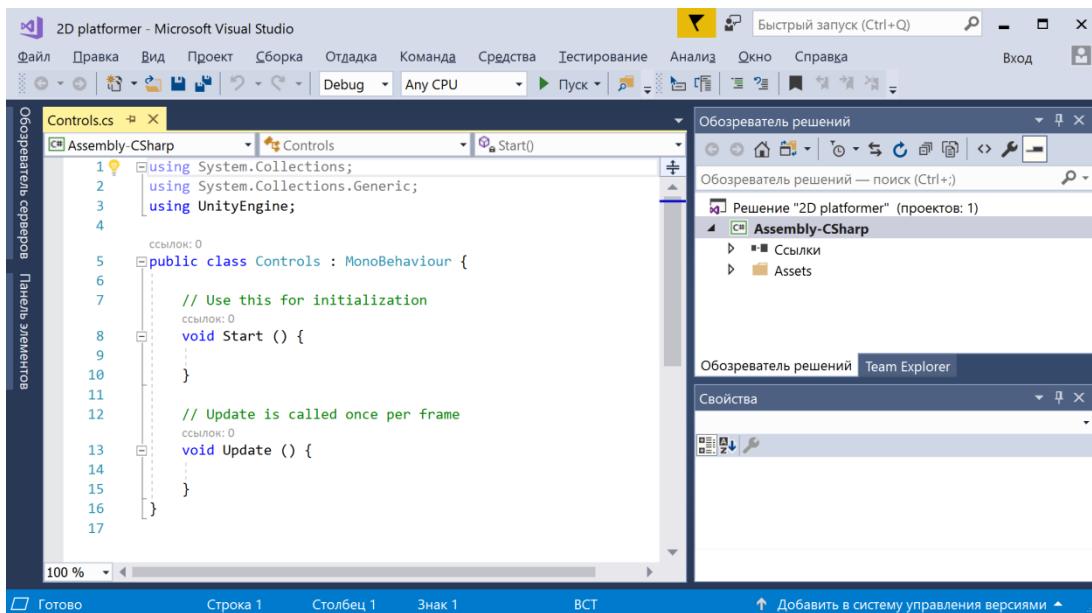


Рисунок 5.6 – Редагування скрипта Controls.cs

В загальному випадку увесь код можна як писати самому, так і використовувати зовнішні джерела. Наприклад, існує цілий магазин «Asset Store», де можна знайти найрізноманітніші ресурси (скрипти, моделі, звуки тощо). При чому складні елементи та великі набори зазвичай є платними. Але для нашого платформера код є дуже простим, тому заливати інші джерела немає ніякої потреби.

Отже, за допомогою напишемо такий код

```
public class Controls : MonoBehaviour {
    public Rigidbody2D rb;
    public float movespeed;

    void Start () {
        rb = GetComponent<Rigidbody2D>();
    }

    void Update () {

        if (Input.GetKey(KeyCode.LeftArrow))
        {
            rb.velocity = new Vector2(-movespeed, rb.velocity.y);
        }
    }
}
```

```
if (Input.GetKey(KeyCode.RightArrow))
{
    rb.velocity = new Vector2(movespeed, rb.velocity.y);
}
}
```

В коді створюється змінна із плаваючою точкою, яка називається movespeed. Вона зроблена загальнодоступною, щоб до неї могли отримати доступ за межами цього сценарію.

Також створимо посилання rb на Rigidbody2D, що був доданий до персонажа. Звернімо увагу, що значення для загальнодоступних змінних можна змінювати за допомогою інспектора об'єкта гри, до якого додається скрипт.

У методі "Start" виконується зв'язування rb з компонентом Rigidbody2D, що є прикріплений до персонажу гри. У методі "Update" виконується прослуховування натискання стрілки вліво або стрілки праворуч, а потім змінюємо положення об'єкту відповідно до натиснутих кнопок. Таким чином до персонажу додається поведінка, що надає йому певної динаміки (рис. 5.7).

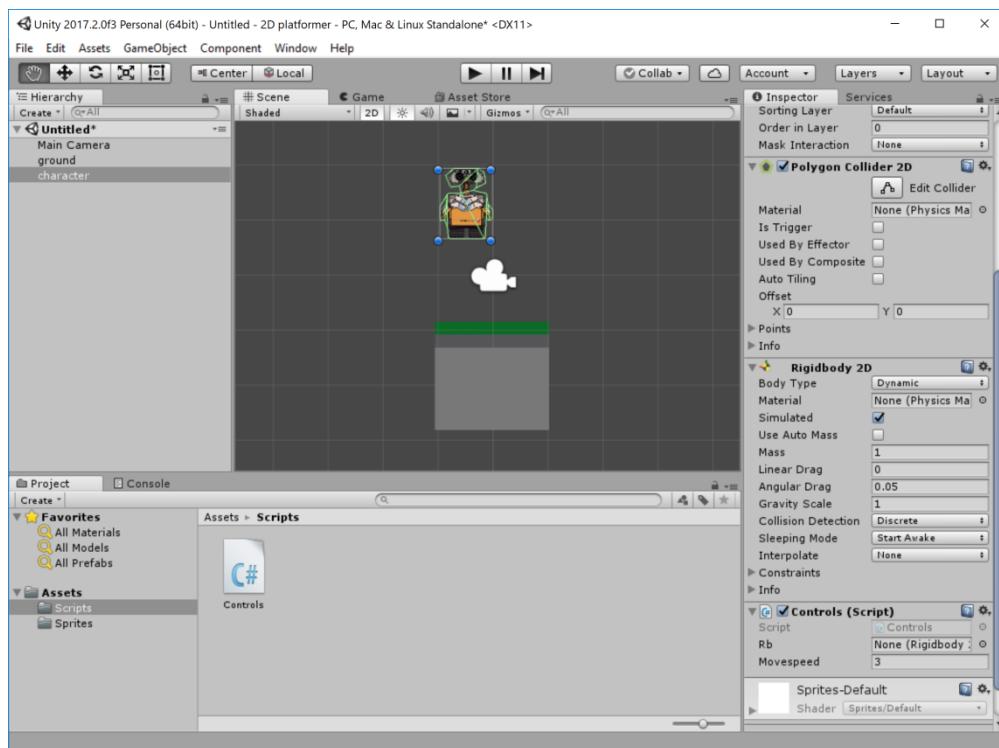


Рисунок 5.7 – Прив'язка скрипту до об'єкта та встановлення швидкості

Далі необхідно повернутися до Unity та перетягнути скрипт Controls.cs на персонажа. Це прив'яже скрипт до конкретного об'єкту та надасть

можливість редагувати його параметри. Також необхідно не забути встановити швидкість руху персонажу – це можна зробити в нижній частині панелі «Inspector». Встановіть будь-яке бажане значення, наприклад, «2».

Результати усіх дій можна побачити, натиснувши «Play». Тепер персонаж по натисканню кнопок переміщується вправо та вліво. Як для мобільного додатку оптимально було б реалізувати сенсорне керування, але це буде розглянуто пізніше.

#### 5.4.5 Додаткові налаштування

Далі внесемо до проекту деякі додаткові зміни.

По-перше, змінемо розмір поверхні-платформи – для цього розтягнемо її по горизонталі, вхопивши за один з країв або скориставшись полями в інспекторі. Зверніть увагу, що початковий спрайт поверхні повинен мати такий рисунок, щоб «компактний» та «розтянутий» варіанти мали одинаковий вигляд.

По-друге, прив'яжемо головну камеру до персонажу. Для цього в панелі ієрархії у лівій частині вікна перетягнемо об'єкт камери на об'єкт персонажу. Таким чином, камера стане дочірнім об'єктом відповідно до персонажу та буде рухатися із ним. Більш того, оберемо об'єкт камери та за допомогою відповідного інструменту переміщення (у верхній лівій частині вікна) змістимо його трохи праворуч від персонажу. Це дозволить переміщуватися більш комфортно і бачити, які об'єкти будуть з'являтися далі на ігровому рівні.

Після прив'язки камери до персонажу виникає одна проблема – при досягненні краю платформи персонаж повинен з неї впасти. Так й виходить, але із урахуванням того, що камера прив'язана до персонажа, падає не персонаж, а платформа. Для виправлення цього прикrogenого ефекту оберіть персонажа, знайдіть на панелі інспектора «RigidBody 2D > Constraints» пропорець «Freeze Position Z» та встановіть його. Тепер персонаж буде вести себе адекватно і звалюватися з платформи.

Третім покращенням буде встановлення деякого фону у вигляді картинки. Для цього необхідно підібрати відповідну картинку та додати її до спрайтів. Після додавання картинку можна збільшити, встановивши значення «Scale X,Y» на 5x5, та перемістити її на фон, задавши «Order in Layer = -1» – це означає, що цей шар буде розміщено за іншими.

Для додання додаткової глибини сцені можна у фонового об'єкту встановити позицію  $Z = 20$ , а тип основної камери змінити з «Projection» на «Perspective». Підсумковий результат наведений на рис. 5.8.

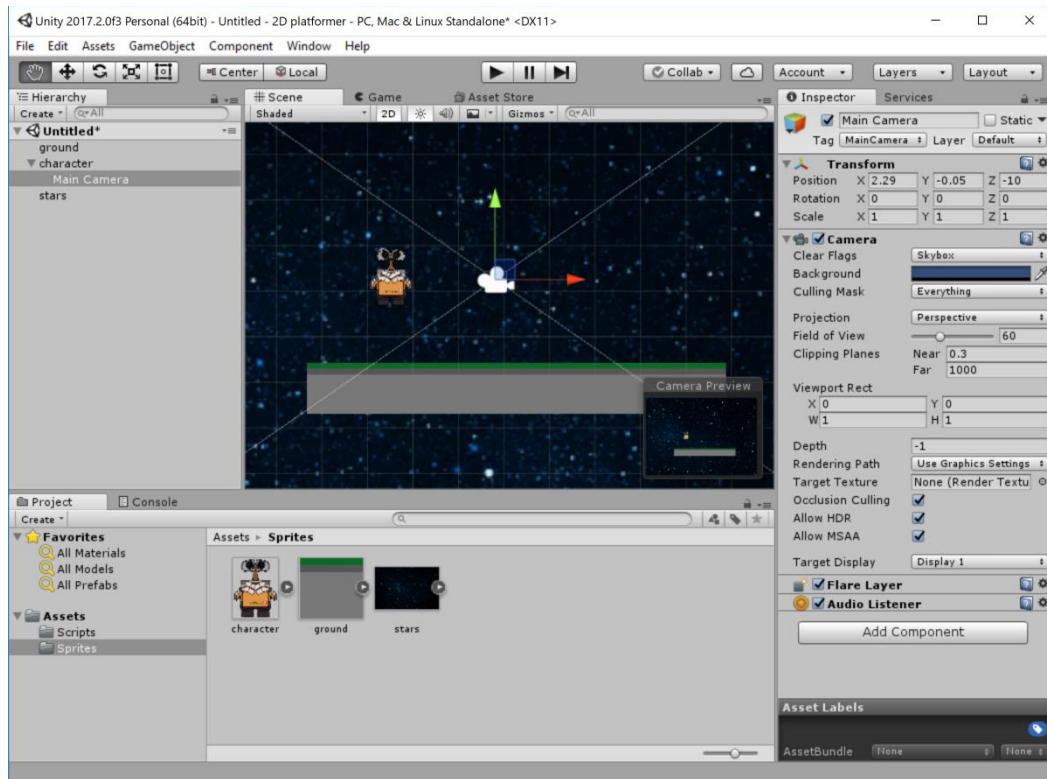


Рисунок 5.8 – Додавання фонового об’єкту та налаштування камери

#### 5.4.6 Додавання сенсорного керування

Наступним кроком розробки додатку є додавання сенсорних кнопок, що будуть використатися для сенсорного керування. Для додавання цих елементів необхідно в меню «GameObject» вибрати «UI > Image». При цьому буде створено Canvas («полотно») – плаваючий шар, що розміщується над ігровою сценою та містить різні UI-елементи (кнопки, життя, здоров'я тощо). Всі ці елементи повинні бути дочірніми відносно Canvas.

В якості кнопок використаємо стрілки, для чого підготуємо відповідний спрайт. Далі створимо Canvas, а в ньому – пустий контейнер («TouchController») із двома зображеннями. Для цього за правою кнопкою миші на Canvas оберемо «Create Empty», і в ньому створимо два зображення. Для прив’язки спрайту зі стрілкою до зображення використаємо поле «Source Image» в інспекторі. При цьому обидві кнопки можуть бути створені з одного

спрайту, просто в одній із кнопок треба встановити «Scale X =-1» для відзеркалювання спрайту (рис. 5.9).

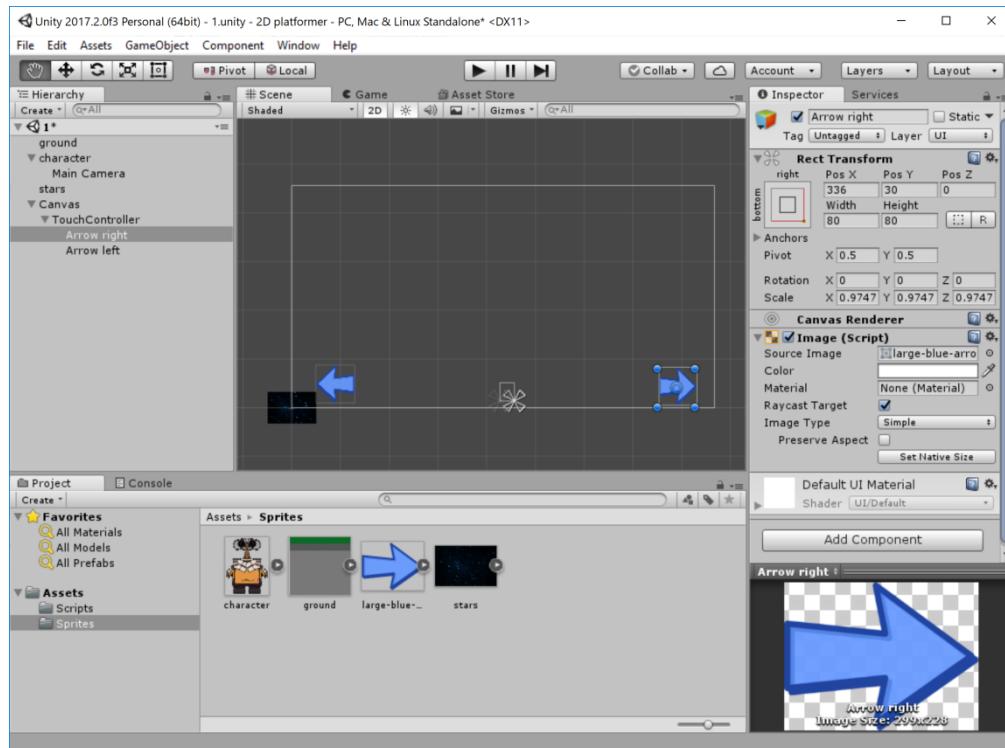


Рисунок 5.9 – Створення сенсорних кнопок

Особливу увагу треба приділити вирівнюванню елементів. Так, контейнер TouchController повинен бути прив'язаний до нижньої границі Canvas та витягнутий за ширину (stretch). А кнопки стрілок в самому контролері прив'язуються відповідно до лівого та правого нижніх кутів.

Далі треба запрограмувати поведінку кнопок, які були щойно додані. Для цього введемо до сценарію Controls.cs дві змінних булевого типу moveright та moveleft та додамо до методу Update такий код:

```
if (moveright)
{
    rb.velocity = new Vector2(movespeed, rb.velocity.y);
}

if (moveleft)
{
    rb.velocity = new Vector2(-movespeed, rb.velocity.y);
}
```

Тепер кожен раз, коли сцена оновлюється, персонаж буде рухатися вліво або вправо відповідно до тих пір, поки відповідна змінна буде "true". Це робиться таким чином тому, що ми можемо виявити, що кнопки

натискаються або випускаються, – але ми не можемо перевірити, чи в даний час вони утримуються.

Наступним кроком буде створення ще одного скрипту Touch.cs з таким кодом:

```
public class Touch : MonoBehaviour
{
    private Controls player;

    void Start()
    {
        player = FindObjectOfType<Controls>();
    }

    public void LeftArrow()
    {
        player.moveright = false;
        player.moveleft = true;
    }
    public void RightArrow()
    {
        player.moveright = true;
        player.moveleft = false;
    }
    public void ReleaseLeftArrow()
    {
        player.moveleft = false;
    }
    public void ReleaseRightArrow()
    {
        player.moveright = false;
    }
}
```

Далі необхідно виконати прив'язку коду до кнопок. Для цього необхідно перетягнути скрипт Touch.cs на елемент «TouchController» у панелі Hierarchy, тим самим виконавши прив'язку. Після цього треба обрати праву кнопку-стрілку в інспекторі, натиснути «Add Component > Event > Event Trigger» та створити два тригери «Pointer Down» та «Pointer Up» за допомогою «Add New Event Type». Ці тригери представляють події натискання та відпускання кнопки.

Далі за допомогою кнопки «+» додамо Runtime-події та перетягнемо TouchController до елементу «None (Object)». Тепер з випадаючого меню можна обрати Touch > RightArrow для Pointer Down та Touch > ReleaseRightArrow для Pointer Up. Це дозволить запускати код корегування змінних moveright та moveleft (рис. 5.10).

Аналогічні дії потрібно виконати для лівої кнопки. У підсумку після проведення усіх налаштувань можна запустити гру та переконатися, що

тепер персонаж реагує як на натискання екранних кнопок, так і на клавіатуру.

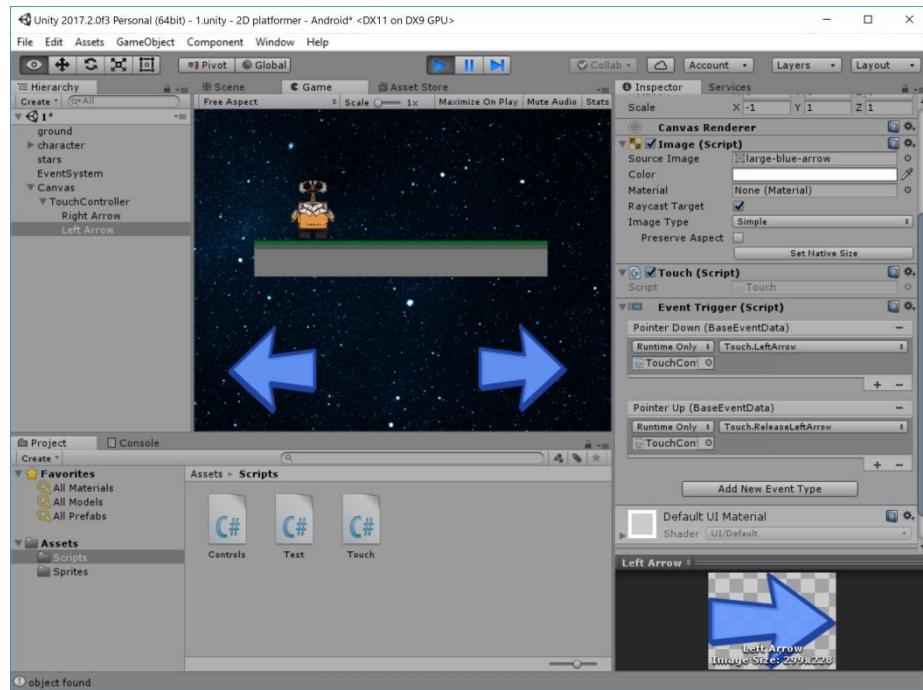


Рисунок 5.10 – Прив’язка скриптів до екранних кнопок

#### 5.4.7 Створення APK-файлу

Перед створенням файла необхідно упевнитися, що сцену було збережено. Для цього потрібно натиснути «File > Save Scene», що призведе до збереження сцени у папці «Assets», але при необхідності, і в залежності від організації проекту, потрібно створити окрему папку «Scenes» та перетягнути сцену туди.

Тепер оберемо «File > Build Settings» та упевнимося, що необхідну сцену додано до списку «Scenes In Build» та встановлений відповідний прапорець. У випадку, коли проект має декілька сцен, порядок має значення – перша сцена у списку буде початковою у додатку. Далі необхідно обрати платформу Android та натиснути «Switch Platform» (рис. 5.11).

При натисканні «Player Settings» в інспекторі відкривається дуже багато різноманітних налаштувань. Наприклад, ви можете створити ваш власний ключ або назву пакету – аналогічно до Android Studio (рис. 5.12).

Крім того, може знадобитися вказати Unity місцезнаходження Java JDK та Android SDK. Це можна зробити в меню Edit > Preferences > External Tools.

Крім того, можна перевірити коректність версії платформи Android, що є встановленою в системі.

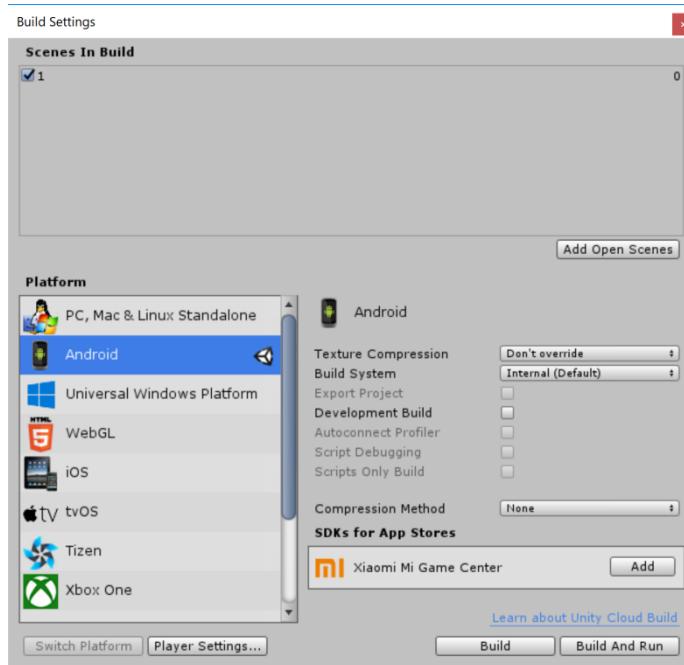


Рисунок 5.11 – Параметри побудови сцени

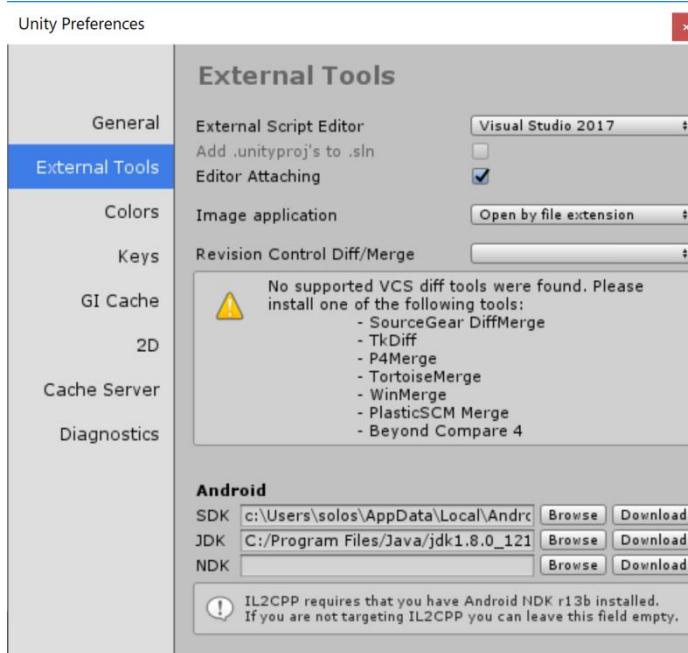


Рисунок 5.12 – Налаштування зовнішніх компонентів

Очікувані практичні результати роботи:

- перша версія мобільного ігрового додатку у жанрі 2D-платформеру із базовою фізичною моделлю та сенсорним керуванням.

## **6 Лабораторна робота № 2. Програмування поведінки і взаємодії ігрових об'єктів, фізичних процесів та основних ефектів**

Лабораторна робота орієнтована на оволодіння студентами навичок із описання та програмування поведінки ігрових об'єктів та їх взаємодії, а також створення ефектів для різних ігрових ситуацій.

Мета лабораторної роботи:

- навчитися контролювати положення об'єктів відповідно до поверхні (платформи);
- ознайомитися із головними принципами реалізації базових трансформацій та використання Physics2D у скриптах;
- навчитися працювати із різними ігровими шарами та групувати об'єкти за ними відповідно до призначення;
- навчитися створювати шаблони (prefabs) та ефектори (effectors) для більш швидкого та зручного створення ігрових рівнів;
- ознайомитися із створенням різних типів перепон і ворогів, що відповідають жанру гри, та навчитися програмувати їх поведінку;
- навчитися додавати до основних ігрових подій ефекти, побудовані на основі системи часток.

Відповідно до наскрізного завдання студент вдосконалює ігровий додаток з лабораторної роботи №1 підтримкою фізики, ігрових перепон та ворогів різних типів, а також додаванням ефектів, що супроводжують ключові ігрові події.

У разі успішного виконання лабораторної роботи студент буде вміти програмувати поведінку та взаємодію основних ігрових об'єктів, створювати ігрові шаблони та використовувати систему часто для створення ігрових ефектів.

Успішне засвоєння матеріалів та виконання лабораторної роботи сприяє формування у студента наступних соціальних навичок та вмінь: оцінювати строки проведення та виконання роботи, обґрунтувати той чи інший спосіб діяльності при виконанні практичних завдань, бажання постійно вчитися, вміння дотримуватися задекларованих регламентних рамок, вільність оволодіння сучасними технологіями та їх використання в усіх сферах життя. Форми прояву: прагнення вирішувати поточні проблеми самостійно, бажання пробувати щось нове, йти на розумний ризик при прийнятті рішень,

елементи самоконтролю і самооцінки при виконанні роботи, вміння планувати свій час.

## 6.1 Завдання до лабораторної роботи № 2

1. Додати до головного персонажу можливість стрибати із платформи на платформу.
2. Реалізувати контроль знаходження персонажу на платформі для запобігання стрибків на безкінечну висоту.
3. Створити шаблони (prefabs) на основі об'єктів персонажу та платформи, додавши до платформи ефектори із розділу Physics 2D.
4. За допомогою шаблону створити декілька нових платформ та таким чином ускладнити наповнення рівню.
5. Розташувати на платформах декілька об'єктів-перепон та запрограмувати контроль зіткнення персонажу із ними.
6. Розташувати на платформах ворогів декількох типів та запрограмувати їх поведінку.
7. За допомогою системи часток створити ефект вибуху та прив'язати його до моменту загибелі персонажу.
8. Створити невидимий об'єкт із шириною на весь ігровий рівень задля забезпечення загибелі персонажу при звалюванні із платформи.

## 6.2 Підготовка до лабораторної роботи № 2

При підготовці до виконання лабораторної роботи необхідно:

1. Протестувати першу версію мобільного додатку, який був створений в ЛР 1, та виправити знайдені недоліки.
2. Ознайомитися із методичними вказівками щодо виконання лабораторної роботи (див. 6.4).

## 6.3 Контрольні питання і попередні матеріали для допуску до лабораторної роботи № 2

### 6.3.1 Контрольні питання

1. Як створити новий проект в Unity?
2. Як додати спрайти до проекту та створити ігрові об'єкти на їх основі?
3. Як за допомогою колайдерів задати базову поведінку об'єкту?
4. Як створити скрипт в Unity та прив'язати його до соб'єкту?
5. Яку основні методи містяться у скрипті і для чого вони призначені?
6. Як звернутися до ігрового об'єкту із скрипта?
7. Як контролювати натиснення клавіатурних кнопок у скрипті?
8. Як реалізувати різну швидкість переміщення персонажу?

9. Якими є принципи групування ігрових об'єктів у ієрархію?
10. На якому ігровому об'єкті необхідно розташовувати сенсорні кнопки?
11. Як реалізувати методи натиснення та відпускання сенсорних кнопок?
12. Як створити APK-файл із ігровим додатком?
13. Які додаткові параметри необхідно налаштувати при створенні APK-файлу?

### 6.3.2 Попередні матеріали

Для допуску к виконанню лабораторної роботи необхідно представити:

1. Перша версія ігрового додатку, що була розроблена в ЛР-1.
2. Спрайти перепони та ворогів (два типи).
3. Звуковий файл із ефектом вибуху.

## 6.4 Методичні вказівки до виконання лабораторної роботи № 2

### 6.4.1 Реалізація стрибків

Будь-які гра жанру платформер не обходить без реалізації стрибків головного персонажу. Тому далі розглянемо реалізацію цієї функції.

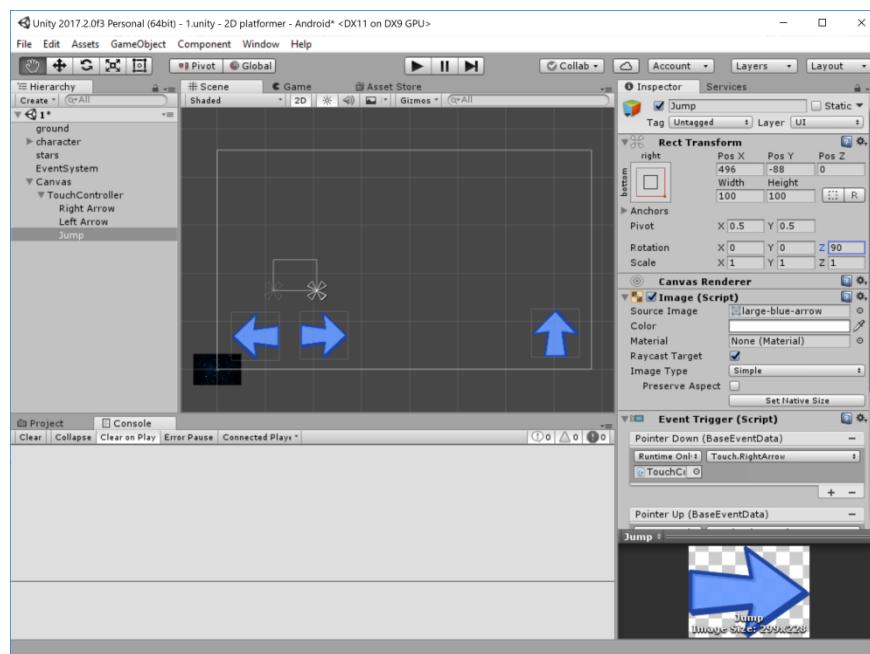


Рисунок 6.1 – Створення кнопки Jump

Перш за все необхідно зменшити елемент керування із сенсорними кнопками та перетягнути його до лівої частини, обравши точкою прив'язки лівий нижній кут. Це дозволить вивільнити місце для кнопки стрибків. Сама

кнопка може бути створена як з нового спрайту, так і з вже існуючого у проекті, але повернутого на 90. Створимо відповідну кнопку та назовемо її Jump (рис. 6.1).

Тепер треба запрограмувати кнопку, що була щойно створена. Для цього відкриємо скрипт Control.cs та задекларуємо дві змінні – jump як public bool та jumpheight як public float. При цьому початкове значення для jumpheight встановлюється за допомогою інспектора і може бути у межах 8-10 одиниць. Далі додамо до методу Update такий код:

```
if (Input.GetKeyDown(KeyCode.Space))
{
    rb.velocity = new Vector2(rb.velocity.x, jumpheight);
}

if (jump)
{
    rb.velocity = new Vector2(rb.velocity.x, jumpheight);
    jump = false;
}
```

Цей код додасть можливість персонажу стрибати при натисненні пробілу на клавіатурі – суть стрибку зводиться до переміщення персонажу за віссю у. А завдяки змінній jump ми можемо прив'язати відповідну дію й до сенсорної кнопки на екрані. Далі залишилось лише прив'язати булеву змінну до кнопки, для чого потрібно додати до скрипту Touch.cs дуже простий код:

```
public void Jump()
{
    player.jump = true;
}
```

При цьому кнопка jump буде мати лише один тригер Pointer Down – послідовність потрібних дій докладно описано в лабораторній роботі 1. Зверніть увагу, що в методі jumpRelease немає сенсу, бо при відпусканні кнопки не має бути жодної дії.

Далі необхідно протестувати коректність стрибання та переміщення персонажу. Для того, щоб сенсорні кнопки не заважали процесу тестування, їх можна скрити. Це робиться за допомогою правої кнопки миші на TouchController в ієрархічному дереві об'єктів або при натисканні відповідного прапорця в інспекторі. Після тестування та налаштування гри все можна повернути назад.

#### 6.4.2 Реалізація гравітації

За результатами тестування в поточному стані гри можна виявити грубий недолік – якщо утримувати пробіл або відповідну сенсорну кнопку, то стрибання може виконуватися безперервно, тобто персонаж за фактом не стрибає, а літає.

Один із способів зробити це – процидання променів (ray casts). Проте, найпростішим способом є аналіз точки поверхні під гравцем у даний момент часу. Для цього потрібно буде створити нове «перетворення» (трансформацію) у сценарії керування. Трансформація – це просто точка в просторі з власними координатами та обертанням. Назовемо трансформацію groundCheck і додамо її таким же чином, як додається будь-яка інша змінна. Крім цього, необхідно додати до точки деякий радіус, булеву змінну onGround та маску шару, яку ми розглянемо пізніше. Отже, сумарно додамо до скрипта Control.cs такий код:

```
public Transform groundCheck;
public float groundCheckRadius;
public LayerMask whatIsGround;
private bool onGround;

...
void FixedUpdate()
{
    onGround = Physics2D.OverlapCircle(groundCheck.position,
    groundCheckRadius,
    whatIsGround);
}
```

FixedUpdate функціонує дуже схоже до Update, але тоді як Update прив'язаний до частоти оновлення екрану, FixedUpdate має більш передбачувану поведінку, що краще підходить для коду, пов'язаного з фізикою. Наведений код встановлює логічне значення onGround на «true» лише тоді, коли нове коло перекриває шар «ground».

Але звичайно, для того щоб це працювало, необхідно встановити координати трансформації groundCheck. Для цього за допомогою панелі ієархії створимо новий пустий ігрової об'єкт Check Ground як дочірній відносно до головного персонажу (натисніть праву кнопку та оберіть «Create Empty»). Тепер оберіть персонажа та перетягніть «Ground Check» до відповідного поля у інспекторі (рис. 6.2). Також треба встановити значення «Ground Check Radius» в «0.1».



Рисунок 6.2 – Прив’язка об’єкту Check Ground до персонажу

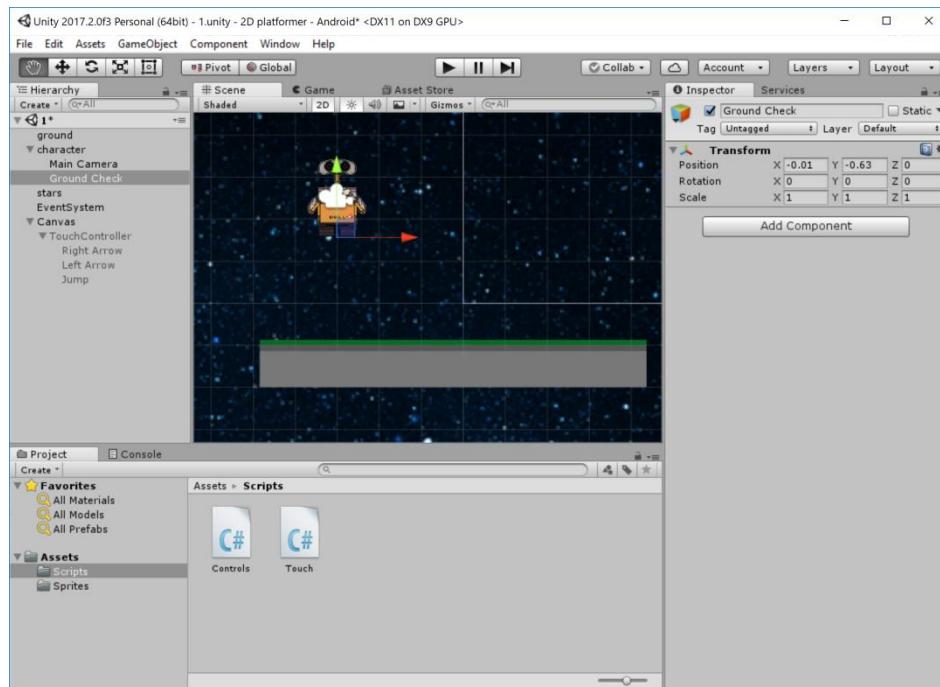


Рисунок 6.3 – Встановлення координат об’єкту Check Ground

Крім того, необхідно упевнитися, що новий пустий об’єкт позиціонований правильно відносно персонажу. Для цього треба подвійним кліком натиснути на об’єкт Check Ground в ієархії і використати переміщення для встановлення об’єкту трохи нижче персонажу із незначним перетинанням (рис. 6.3).

Далі необхідно створити шар «землі». Для цього оберемо об’єкт персонажу, знайдемо випадаючий список Layers (Шари) та натиснемо Add Layer (Додати шар). В списку шарів знайдемо перший вільний рядок та введемо «Ground» (рис. 6.4).

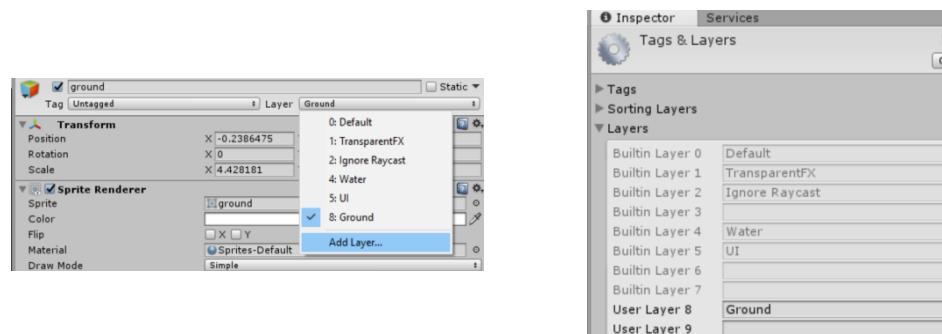


Рисунок 6.4 – Додавання новогу шару

Тепер поверніться до об'єкту платформи та встановіть відповідну галочку у випадаючому меню рядом із шаром Ground (який з'явився в списку). Далі виберіть об'єкт персонажу в ієрархії і встановіть той самий шар для параметру whatIsGround – це прив'яже шар до доданого раніше у коді.

Тепер нарешті можна провести кінцеве тестування переміщення персонажу – він повинен стрибати тільки з платформи та тільки один раз, тобто в грі з'явилася гравітація.

#### 6.4.3 Робота із шаблонами та модифікаторами

Після реалізації базової функціональності треба зайнятися ускладненням ігрового рівня. І найлогічніший крок – створити більше платформ. Найбільш простим способом є копіювання, вставка та зміна розмірів, але перед тим доцільно розглянути такі елементи як шаблони (prefabs).

Шаблон – це базовий об'єкт, який має задану кількість параметрів. Після створення об'єктів на основі шаблону, змінення параметрів у шаблоні змінює ці параметри й у всіх об'єктів, що обули породжені. Це позбавляє необхідності змінювати параметри у кожного з великої кількості об'єктів.

Для створення шаблону достатньо в Assets створити нову папку Prefabs і перетягнути до неї елемент поверхні з вікна ієрархії. Після цього можна створювати нові платформи перетягуванням із папки, що була щойно створена. Отже, за допомогою шаблонів створіть декілька платформ таким чином, щоб їх проходження складало цільний рівень (рис. 6.5).

Якщо зараз спробувати пограти на щойно створених платформах, то можна побачити, що гра має значний недолік. Персонаж, переміщаючись платформами, в деяких випадках прилипає до боків платформи. Причиною є тертя, якого необхідно позбавитися для всіх платформ разом. А це дуже слушна нагода продемонструвати усю зручність користування шаблонами.

Отже, оберіть шаблон ground із папки Prefabs, встановіть прaporець Used by effector та за допомогою інспектора додайте до нього компонент Platform Effector 2D із розділу Physics 2D. Далі скиньте прaporець Use one way, якщо не бажаєте щоб персонаж стрибати крізь платформи знизу. Крім того, треба упевнитися прaporець Use side friction знятий за замовчуванням. Таким чином, цей модифікатор робить платформу дійсно платформою з відповідною поведінкою. Аналогічно функціонують і багато інших модифікаторів, що призначенні для спрощення організації та уніфікації ігрового процесу.

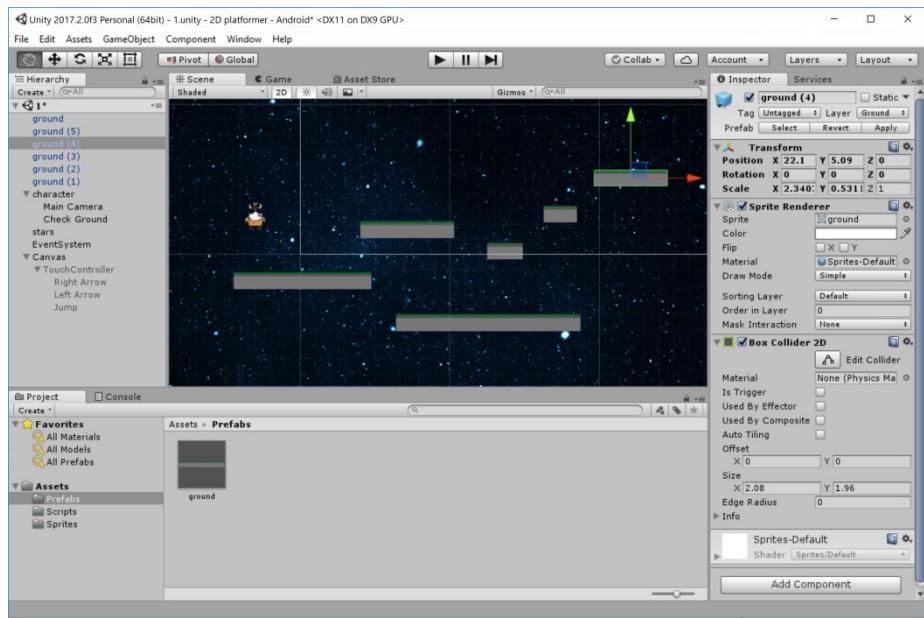


Рисунок 6.5 – Створення декількох платформ з шаблонів

#### 6.4.4 Додавання перепон та ворогів

Жанр платформеру передбачає наявність різних перепон то ворогів. У загальному випадку перепони відрізняються від ворогів тим, що перші є нерухомі, а другі – рухаються.

Почнемо з перепон. Перш за все необхідно створити чи знайти відповідний спрайт (рис. 6.6) та додати його до проекту.

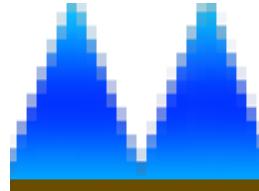


Рисунок 6.6 – Спрайт перепони

Далі потрібно створити об'єкт на основі спрайту (перетягнути його на сцену), додати до нього полігональний колайдер та встановити пропорець Is trigger. Це буде означати, що об'єкт не буде вести себе як платформа, і можна буде описати його поведінку в коді. Для цього створимо новий скрипт та назовемо його Hazard.cs:

```
public class Hazard : MonoBehaviour
{
    private Controls player;
    public Transform start;
```

```
void Start()
{
    player = FindObjectOfType<Controls>();
}

void Update()
{
}

void OnTriggerEnter2D(Collider2D other)
{
    if (other.tag == "Player")
    {
        player.transform.position = start.position;
    }
}
```

Метод `OnTriggerEnter2D` перевіряє, чи є торкання (пересічення) колайдера із іншими об'єктами. В коді перевіряється, чи це торкання є з об'єктом, який має тег «`Player`». Якщо так, то виконується переміщення до стартової позиції. Зверніть увагу, що для об'єкту персонажа за допомогою інспектора необхідно встанови значення тега в «`Player`».

Далі прив'яжемо новостворений скрипт до об'єкту перепони за допомогою `Add Component > Scripts > Hazard`. Крім того, необхідно створити пустий ігровий об'єкт `Start`, що буде виконувати роль початкової позиції. Його треба розмістити там, де повинна починатися (перезавантажуватися) гра після зіткнення персонажа із перепоною.

Тепер можна додати перепону до шаблонів, та на основі шаблону створити декілька об'єктів, розміщуючи їх на платформах у найбільш небезпечних місцях. Але треба не забувати прив'язувати об'єкт `Start` до скрипту, бо інакше не буде виконана початкова ініціалізація. Для цього достатньо перетягнути об'єкт `Start` до відповідного поля скрипту у інспекторі.

Як відомо, до кожного об'єкта в Unity можна прив'язувати декілька скриптів, що дозволяє створювати різні типи перепон і ворогів. Так, додамо новий скрипт `ObjectMoves.cs` із таким кодом:

```
public class ObjectMove : MonoBehaviour
{
    public float amounttomovex;
    public float speed;
    private float currentposx;
    private float currentposy;
    private int facing;

    void Start()
    {
        currentposx = gameObject.transform.position.x;
        facing = 0;
    }

    void Update()
```

```
{  
    if (facing == 1 && gameObject.transform.position.x <  
        currentposx - amounttomovex)  
    {  
        facing = 0;  
    }  
  
    if (facing == 0 && gameObject.transform.position.x > currentposx)  
    {  
        facing = 1;  
    }  
  
    if (facing == 0)  
    {  
        transform.Translate(Vector2.right * speed * Time.deltaTime);  
    }  
    else if (facing == 1)  
    {  
        transform.Translate(-Vector2.right * speed * Time.deltaTime);  
    }  
}  
}
```

Цей скрипт забезпечує переміщення об'єкту по горизонталі у заданих межах. Швидкість та діапазон переміщення вказуються у інспекторі за допомогою зовнішніх параметрів speed та amounttomovex. Тепер за допомогою двох схожих спрайтів (рис. 6.7) можна створити два типи ворогів.



Рисунок 6.7 – Два типи ворогів

Аналогічно до вже створеної перепони, зіткнення персонажа із ворогом першого типу перезапускає рівень. Але на відміну від перепони об'єкт ворога ще й переміщається. Така поведінка досягається додаванням до об'єкту двох скриптів – Hazard.cs та ObjectMove.cs. Ворог іншого типу не знищує персонажа, а лише зіштовхує його із платформи. Це досягається встановленням на об'єкт ворога колайдеру Circle Collider, скидання пропорції «Is trigger» та прив'язкою скрипту ObjectMove.cs.

Після усіх налаштувань на базі ігрових об'єктів можна створити шаблони та розмножити їх по різних платформах (рис. 6.8).

#### 6.4.5 Ефекти та система часток

На даному етапі гри, загибель персонажу гри виглядає не досить переконливо – він просто переміщується на початок рівня та все починається

заново. Для виправлення ситуації доцільно буде розібратися із використати систему часток.

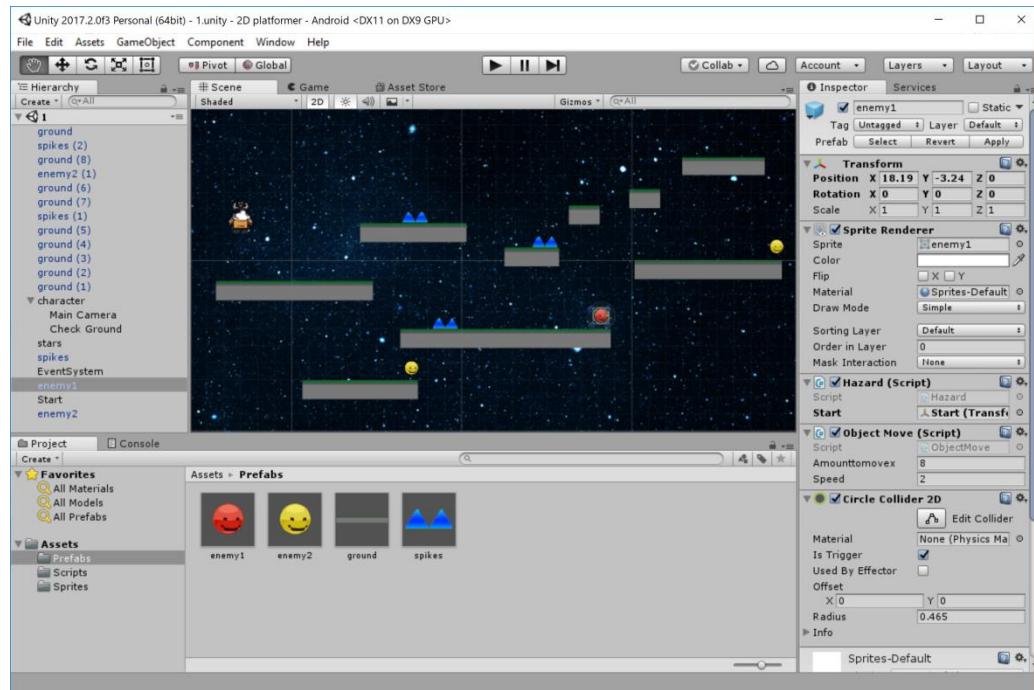


Рисунок 6.8 – Розстановка ворогів по рівню

Створення об'єкту системи часток створюється за допомогою пункту меню «Game Object > Particle System». В початковому стані такий об'єкт являє собою фонтан з мілких частинок, які безкінечно вилітають із зазначеного місця. Обравши об'єкт системи часток в інспекторі можна знайти безліч налаштувань, що контролюють усі аспекти відображення системи часток (рис. 6.9).

Граючись із налаштуваннями, можна підібрати такі параметри, що найбільш відповідають невеличкому вибуху при загибелі персонажу. Колір та вид частинок підберіть за смаком, а в Renderer -> Default Particle вкажіть Default Sprite.

Далі необхідно створити новий скрипт DestroyParticle, який буде виконувати видалення об'єкту системи часток після програвання анімації. Якщо не реалізувати це видалення, Unity кожен раз буде створювати новий об'єкт системи часток, що призведе до неефективного використання ресурсів системи. Код скрипта:

```
public class DestroyParticle : MonoBehaviour
{
    private ParticleSystem thisParticleSystem;
```

```
void Start()
{
    thisParticleSystem = GetComponent<ParticleSystem>();
}

void Update()
{
    if (thisParticleSystem.isPlaying)
    {
        return;
    }
    Destroy(gameObject);
}
```

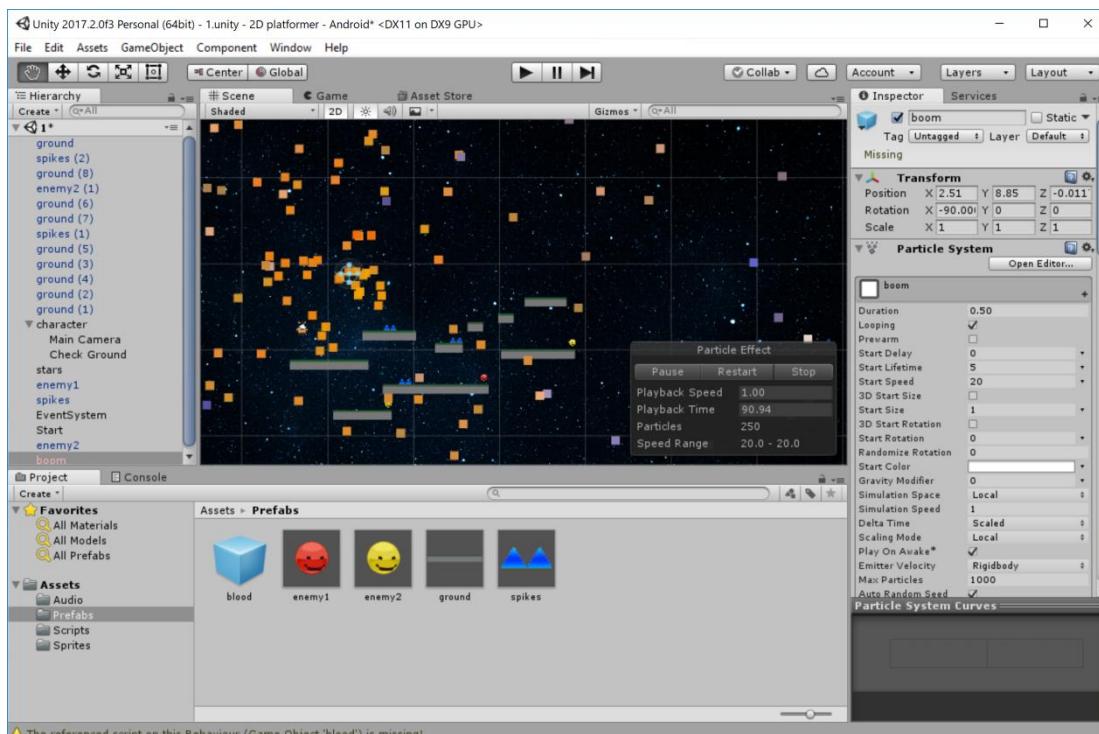


Рисунок 6.9 – Створення об’єкту системи часток

Далі потрібно прив’язати створений скрипт до об’єкту системи часток. Тепер можна дати об’єкту назву, скопіювати його до шаблонів (папка Prefabs) та видалити початковий об’єкт.

Наступний кроком є декларація нового об’єкту Explode та додання до методу onTriggerEnter скрипта Hazard.cs такого кода:

```
public GameObject Explode;
...
Instantiate(Explode, player.transform.position, player.transform.rotation);
```

Цей код створює об’єкт Explode, що задає анімацію вибуху, та виконує початкову ініціалізацію. Відповідно, в ієархії потрібно створити пустий об’єкт, назвати його Explode та прив’язати його до кожного з ворогів та

перепон – бо саме до них прив’язаний скрипт Hazard.cs, в якому з’явився новий параметр Explode.

Для посилення ефекту вибуху можна додати звук. Створимо нову папку Audio та додамо туди файл із звуковим ефектом вибуху. Далі в шаблоні системи часток додамо компонент Audio Source та оберемо файл із папки Audio, а також встановим пропорець Play On Awake. Тепер вибух буде супроводжуватися відповідним звуковим ефектом.

І нарешті, підсумковим кроком є вдосконалення коду, що відповідає за перезавантаження гри в скрипті Hazard.cs:

```
void OnTriggerEnter2D(Collider2D other)
{
    if (other.tag == "Player")
    {
        StartCoroutine("respawndelay");
    }
}

public IEnumerator respawndelay()
{
    Instantiate(Explode, player.transform.position,
player.transform.rotation);
    player.enabled = false;
    player.GetComponent<Rigidbody2D>().velocity = Vector3.zero;
    player.GetComponent<Renderer>().enabled = false;
    yield return new WaitForSeconds(1);
    player.transform.position = start.position;
    player.GetComponent<Renderer>().enabled = true;
    player.enabled = true;
}
```

Цей код розміщує анімацію загибелі персонажа в корутину – підпрограму, що виконується паралельно із основним потоком. Це дозволяє додати деяку затримку, що встановлюється за допомогою WaitForSeconds. На час цієї затримки персонаж зникає і блокується від жодних переміщень перед наступним завантаженням. При цьому увесь імпульс переміщення теж убирається – для того щоб він не переносився на пересування персонажа вже після перезавантаження.

Наприкінці треба зазначити, що за логікою гри персонаж повинен загинути не лише при зіткненні з перепонами і ворогами, а й просто звалившись із платформи. Реалізується це дуже просто. Під усіма платформами створюється пустий об’єкт, який за ширину перекриває увесь рівень (рис. 6.10) – назвемо його BorderBottom. Далі додаємо до нього стандартний Box Collider та скрипт Hazard.cs. Тепер при звалюванні з платформи персонаж рано чи пізно перетнеться із цим об’єктом і рівень перезавантажиться.

Отже, тепер можна повернути видимість блоку сенсорних кнопок, згенерувати APK-файл та протестувати роботу гри безпосередньо на пристройі.

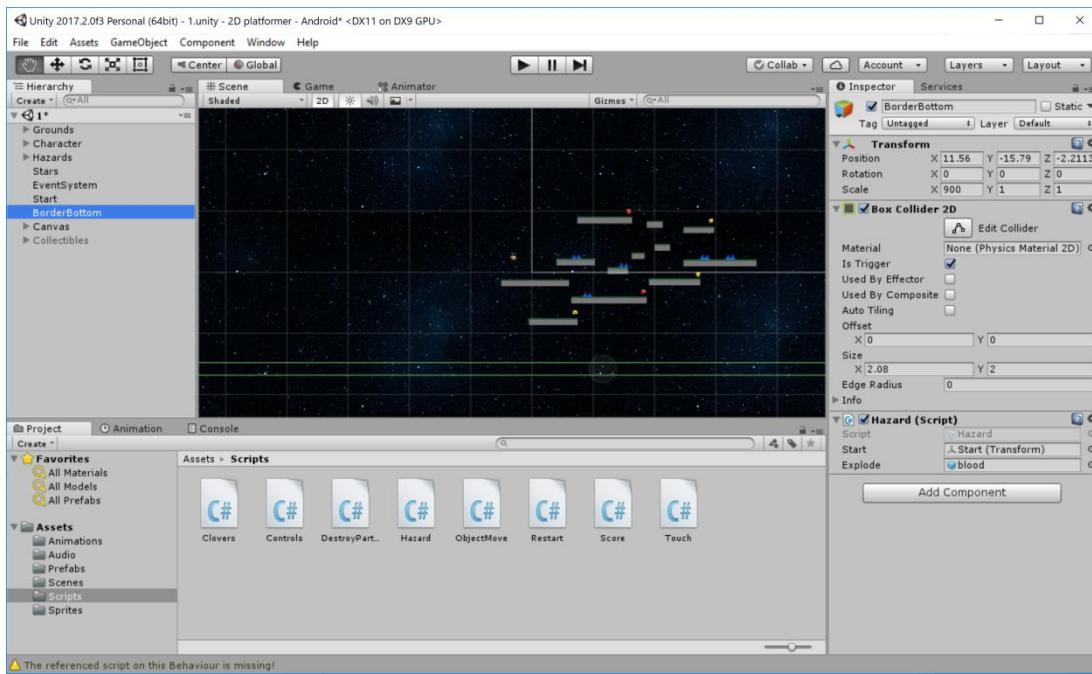


Рисунок 6.10 – Об’єкт BorderBottom

Очікувані практичні результати роботи:

- друга версія мобільного ігрового додатку із розвиненою побудовою рівня, перепонами і ворогами різних типів та ефектами, що супроводжують основні ігрові події.

## 7 Лабораторна робота №3. Анімація персонажів, реалізація предметів і цілей, організація ігрових рівнів.

Лабораторна робота орієнтована на оволодіння студентами навичок створення анімації ігрових об’єктів за допомогою таймлайнів та діаграм переходів, а також реалізації предметів, що збираються (collectibles) та організації ігрових декількох рівнів.

Мета лабораторної роботи:

- навчитися додавати до гри предмети, що збираються (collectibles), та реалізовувати відповідні скрипти підрахунку таких предметів;

- ознайомитися із засобами виведення інформації екран пристрою за допомогою елементів Unity.UI;
- навчитися створювати таймлайні анімації на основі декількох спрайтів;
- навчитися розробляти діаграми переходів між анімаціями за значенням окремого параметра, який контролюється відповідним скриптом;
- ознайомитися із механізмом створення декількох рівнів із використанням однакових шаблонів;
- навчитися організовувати переходи між рівнями при виконанні ігрової умови (досягнення певної позиції або збирання певної кількості предметів);
- ознайомитися із додатковими налаштуванням при генерації мобільного додатку, що пов’язані із параметрами екрану.

Відповідно до наскрізного завдання студент реалізує анімацію основних дій головного персонажу, додає до гри предмети, що можна збирати, та організовує переход між різними ігровими рівнями за умови виконання заданих ігрових досягнень.

У разі успішного виконання лабораторної роботи студент оволодіє знаннями щодо створення анімації персонажів та ігрових об’єктів, реалізації функції збирання предметів, розбудови декількох ігрових рівнів та переходів між ними.

Успішне засвоєння матеріалів та виконання лабораторної роботи сприяє формування у студента наступних соціальних навичок та вмінь: бажання постійно вчитися, вміння дотримуватися регламентних рамок, оцінювати строки проведення та виконання роботи, обґрунтовувати той чи інший спосіб діяльності при виконанні практичних завдань. Форми прояву: прагнення вирішувати поточні проблеми самостійно, бажання пробувати щось нове, йти на розумний ризик при прийнятті рішень, елементи самоконтролю і самооцінки при виконанні роботи, вміння планувати свій час.

## 7.1 Завдання до лабораторної роботи

1. Додати до ігрового додатку фонову музику.
2. Налаштувати основні параметри персонажа (швидкість, висота стрибків тощо) для забезпечення цікавого ігрового процесу.
3. Згрупувати усі елементи по окремих папках в ієрархії.

4. Додати механізм збирання предметів та розташувати предмета на платформах.
5. Забезпечити відображення поточної кількості назбираних предметів за допомогою текстового елементу.
6. Створити три анімації персонажу – переміщення вліво, переміщення вправо та стан покою.
7. Організувати переход між анімаціями шляхом створення діаграми переходів.
8. Додати до основного скрипту код керування параметрами анімації.
9. Створити ще один рівень (сцену), що відповідає за індикацію завершення гри та можливість перезапуску.
10. Забезпечити переход від ігрової сцени до фінального екрану при досягненні певної кількості назбираних об'єктів або за умови досягнення персонажем певної позиції.
11. Створити APK-файл підсумкового варіанту ігрового додатку та протестувати його роботу.

## 7.2 Підготовка до лабораторної роботи № 3

При підготовці до виконання лабораторної роботи необхідно:

1. Протестувати другу версію мобільного додатку, що був створений в ЛР 2, та виправити знайдені недоліки.
2. Ознайомитися із методичними вказівками щодо виконання лабораторної роботи (див. 7.4).

## 7.3 Контрольні питання і попередні матеріали для допуску до лабораторної роботи № 3

### 7.3.1 Контрольні питання

3. Як реалізувати «гравітацію» для приземлення персонажу?
4. Чим метод FixedUpdate відрізняється від Update?
5. Що таке шар та як в Unity реалізована робота із шарами?
6. Як створити новий шар та помістити на нього об'єкти?
7. Навіщо створюються шаблони (prefabs) та модифікатори (effectors) і як з ними працювати?
8. Як позбутися ефекту «прилипання» персонажу до боковин платформи?
9. Як зафіксувати зіткнення персонажу із перепоною?

10. Як реалізувати різну поведінку декількох типів ворогів?
11. Навіщо додавати до об'єктів ворогів різні типи колайдерів?
12. Що таке «система частинок» та для чого вона може бути використана?
13. Які основні параметри є у об'єкту системи частинок?
14. Чому потрібно вручну видаляти об'єкти системи часток?
15. Що таке «корутіна» (coroutine) і для чого вона потрібна?
16. Як додати звуковий ефект до будь-якої події?
17. Як в скрипті реалізувати очікування?

### 7.3.2 Попередні матеріали

Для допуску к виконанню лабораторної роботи необхідно представити:

1. Друга версія ігрового додатку, що була розроблена в ЛР 2.
2. Спрайт предмету для збирання.
3. Спрайти для створення анімації головного персонажу (рух вліво, рух вправо, стояння на місці).
4. Спрайт кнопки перезапуску гри.
5. Звуковий файл із фоновою музикою.

## 7.4 Методичні вказівки до виконання лабораторної роботи № 3

### 7.4.1 Звуки та інші вдосконалення

Жодна гра не обходитья без звукового супроводження. Отже, в попередній лабораторній роботі ми вже додали звук до вибуху, що стається у випадку загибелі персонажу. Для додавання є фонової музики достатньо добавити відповідний файл до папки Audio та прив'язати його у вигляді компоненту до об'єкту персонажу (рис. 7.1).

Серед додаткових параметрів необхідно встановити пропорці «Play On Awake» (програвання починається разом із початком активності персонажа) та «Loop» (зациклює програвання).

Далі зробимо деякі зміни у поведінці персонажу, змінивши такі параметри:

- Gravity Scale: 2;
- Angular Drag: 13;
- Jumpheight: 12;
- Movespeed: 4.

Це зробить гру більш азартною та інтересною. Наступним вдосконаленням може бути збільшення сенсорних кнопок на екрані, але це потрібно коректувати вже в кожному окремому випадку.

Крім того, треба запобігти запуску гри на пристрої в портретному режимі. Для цього перейдемо до File > Build Settings > Player Settings > Resolution and Presentation та встановим відповідні пропорці (рис. 7.2).

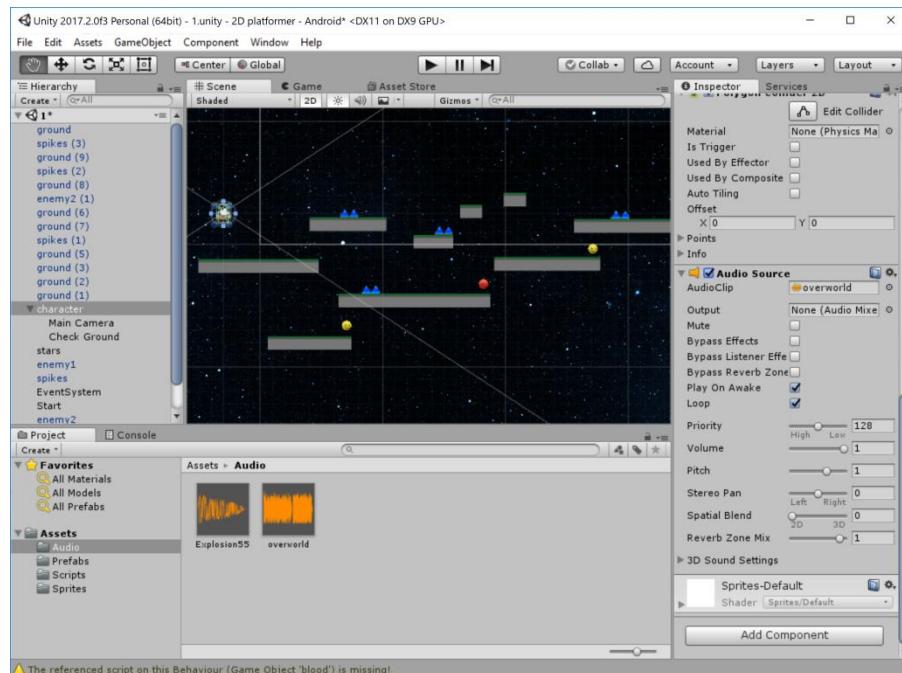


Рисунок 7.1 – Додавання фонової музики

Тепер необхідно навести порядок у ієрархії, згрупувавши однакові елементи у папки, які можна створити як пусті ігрові об'єкти (Create Empty) та перетягнути до них ігрові об'єкти. Це дозволить швидше дістатися до потрібного елементу (рис. 7.2).

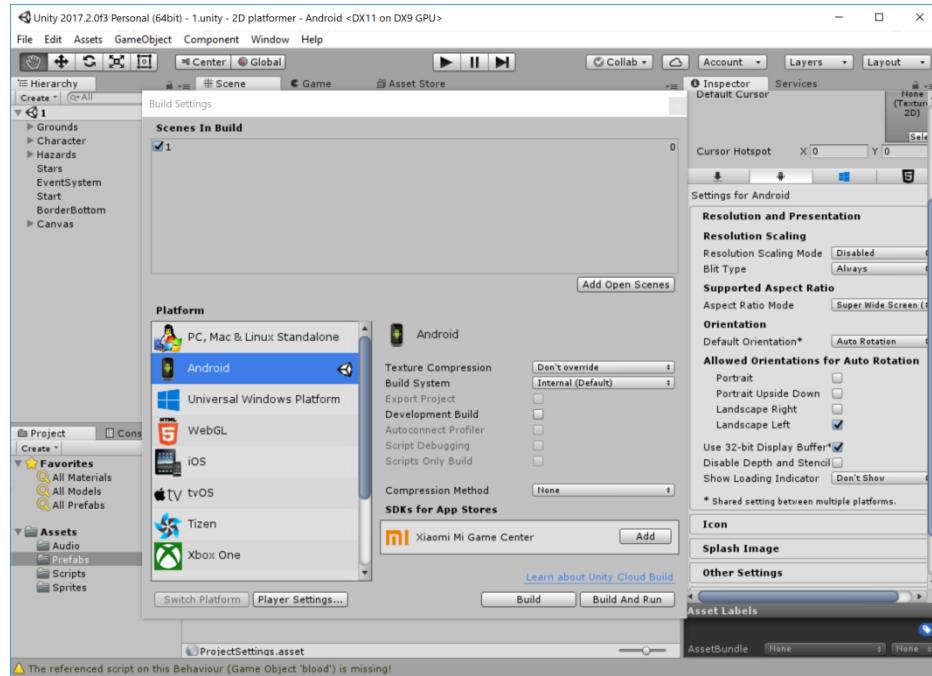


Рисунок 7.2 – Налаштування орієнтації дісплею

#### 7.4.2 Збирання предметів

Кожний добрий платформер має мати винагороди (предмети, collectibles), що потрібно збирати протягом проходження рівня, – монети, кристали або щось таке інше. Отже додамо їх до гри. Як завжди, почнемо із спрайту, який буде представляти винагороду (рис. 7.3).



Рисунок 7.3 – Спрайт предмету для збирання

В основному скрипті (Controls.cs) створимо нову змінну, яка буде зберігати поточну кількість предметів, що називав гравець. Назовемо її clovers, оберемо тип int та зробимо public – щоб інші скрипти мали доступ до неї:

Далі створимо на базі спрайта ігрової об’єкт, додамо полігональний колайдер (polygon collider) та вкажемо, що керувати об’єктом буде скрипт (isTrigger = true). Також треба упевнитися, що об’єкт-предмет знаходиться

завжди попереду – це регулюється за допомогою Order in Layer. Після усіх налаштувань на основі об'єкту можна створити шаблон та помістити його в папку Prefabs.

Для впорядкування усіх предметів, що будуть створені пізніше, створімо в ієрархії сцени новий пустий об'єкт Collectibles. Перетягніть на нього звуковий файл, що буде програватися кожен раз при збиранні предмету. В автоматично створеному компоненті Audio Source пропорець Play on awake повинен бути скинутий.

Тепер створімо новий скрипт Score.cs, який за механізмом роботи дуже схожий на Hazard.cs. Головною відмінністю є те, що зіткнення персонажа з предметом викликає знищення самого предмету. Також треба не забути про звуковий ефект зіткнення та нарощування кількості збираных предметів. Код скрипта:

```
public class Crystals : MonoBehaviour {  
  
    private Controls player;  
    public AudioSource bling;  
  
    void Start () {  
        player = FindObjectOfType<Controls>();  
    }  
  
    void Update () {}  
  
    void OnTriggerEnter2D(Collider2D other)  
    {  
        if (other.tag == "Player")  
        {  
            Destroy(gameObject);  
            bling.Play();  
            player.clovers++;  
        }  
    }  
}
```

Для того, що скрипт працював, необхідно перетягнути щойно створений контейнер Collectibles до поля Audio source предмета – тобто виконати його ініціалізацію звуком, що прикріплений до контейнеру. Недоліком є те, що це необхідно робити для кожного предмету, а не початкового шаблону – так, це не дуже зручно, але ініціалізація може бути виконана лише для конкретного об'єкту, а не шаблону. Але для більш швидкого наповнення рівня предметами можна просто копіювати їх та вставляти – так зв'язок із звуковим файлом буде збережено.

Далі необхідно реалізувати відображення поточної кількості назбираних предметів. Для цього доцільно створити UI-елемент, що буде дочірнім об'єктом Canvas – аналогічно сенсорним кнопкам. Тому оберемо Canvas в

ієрархії та створимо новий текстовий елемент за допомогою GameObject > UI > Text. Для наочності зразу ж на пишемо в ньому «Clevers : 0», і налаштуємо колір та розмір шрифту в елементі, а також вирівняємо його за лівим верхнім кутом (рис. 7.4).

Для керування елементом необхідний відповідний скрипт Score.cs, прив'язаний до текстового елементу:

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

public class Score : MonoBehaviour {
    Text text;
    private Controls player;

    // Use this for initialization
    void Start() {
    }

    void Update () {
        text.text = "Crystals: " + player.clovers;
    }
}
```

Зверніть увагу, що доступ до UI-елементів вимагає підключення UnityEngine.UI за допомогою using.

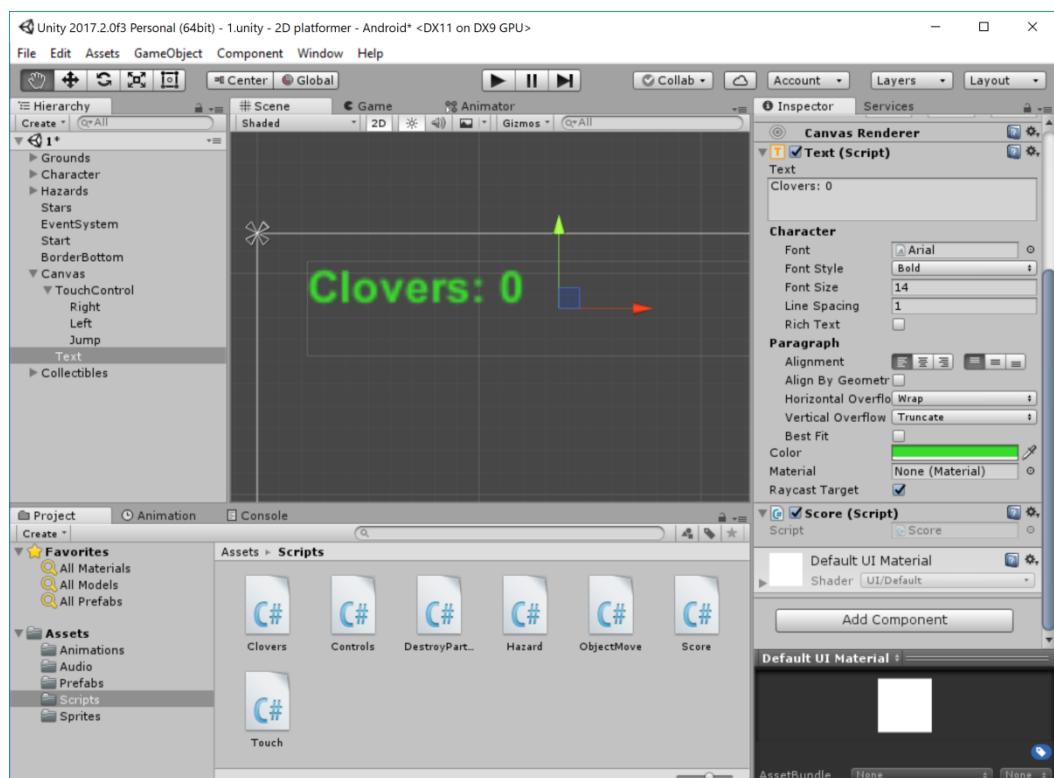


Рисунок 7.4 – Додавання тексту

### 7.4.3 Анимація персонажу

Наступним кроком вдосконалення гри є анимація головного персонажу. Це зробить гру більш динамічною та ознайомить із механізмом анимації в Unity.

Все як завжди починається із спрайтів. Припустимо, що нам необхідно реалізувати три основні види анимації:

1. «Покій» (Idle) – анимація у випадку, коли персонаж стоїть на місці і не рухається.
2. «Вліво» (WalkingLeft) – анимація персонажу при русі вліво.
3. «Вправо» (WalkingRight) – анимація персонажу при русі вправо.

Відповідні спрайти наведені на рис. 7.5. Треба зазначити, що їх може бути і більше, бо чим більше спрайтів – тим краще анимація. Але для демонстрації роботи механізмів анимації достатньо й цих.

Далі за допомогою головного меню необхідно відкрити два вікна – Animation and Animator, та обрати для них найзручніше місце. Після цього оберіть головного персонажа у вкладці Scene (або в ієрархії), залишаючи при цьому видимим вікно Animation. В ньому з'явиться кнопка Create – натисніть її та створіть першу анимацію Idle. Для збереження усіх анимацій буде доречно створити окрему папку Animations в папці Assets.

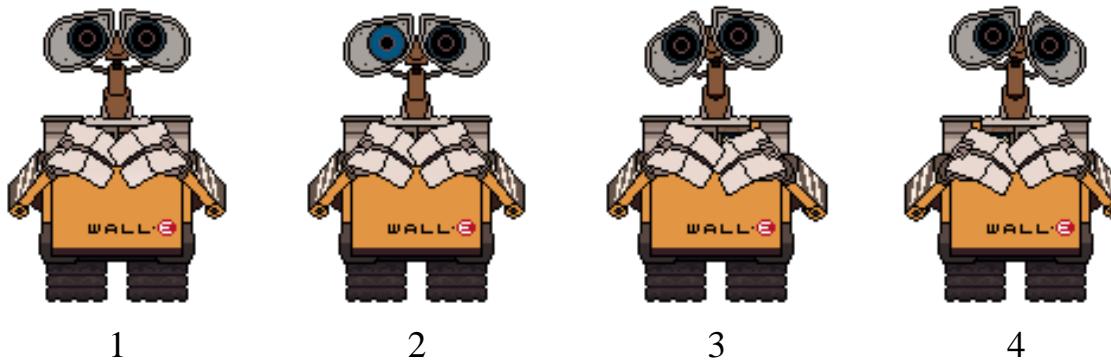


Рисунок 7.5 – Спрайти для анимації

Після створення першої анимації в лівому верхньому куті вікна з'являється надпис Idle (назва анимації) із двома невеличкими стрілками. Якщо натиснути на назву, то відкривається меню, за допомогою якого можна створити інші необхідні анимації – WalkingLeft та WalkingRight (рис. 7.6). За допомогою цього ж меню виконується і переключення між усіма наявними анимаціями персонажу.

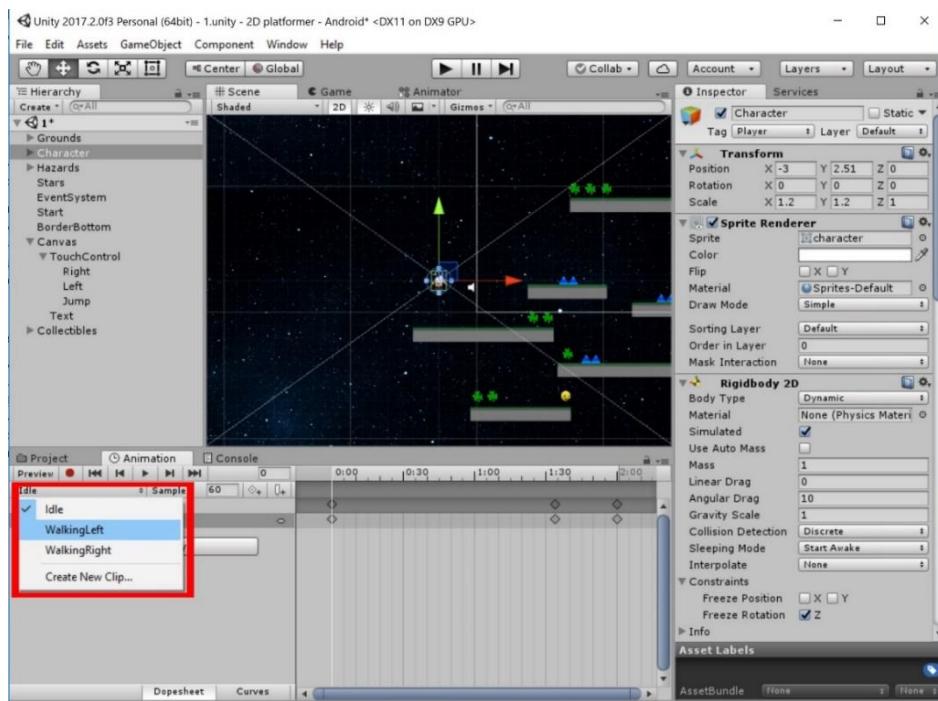


Рисунок 7.6 – Створення анімацій

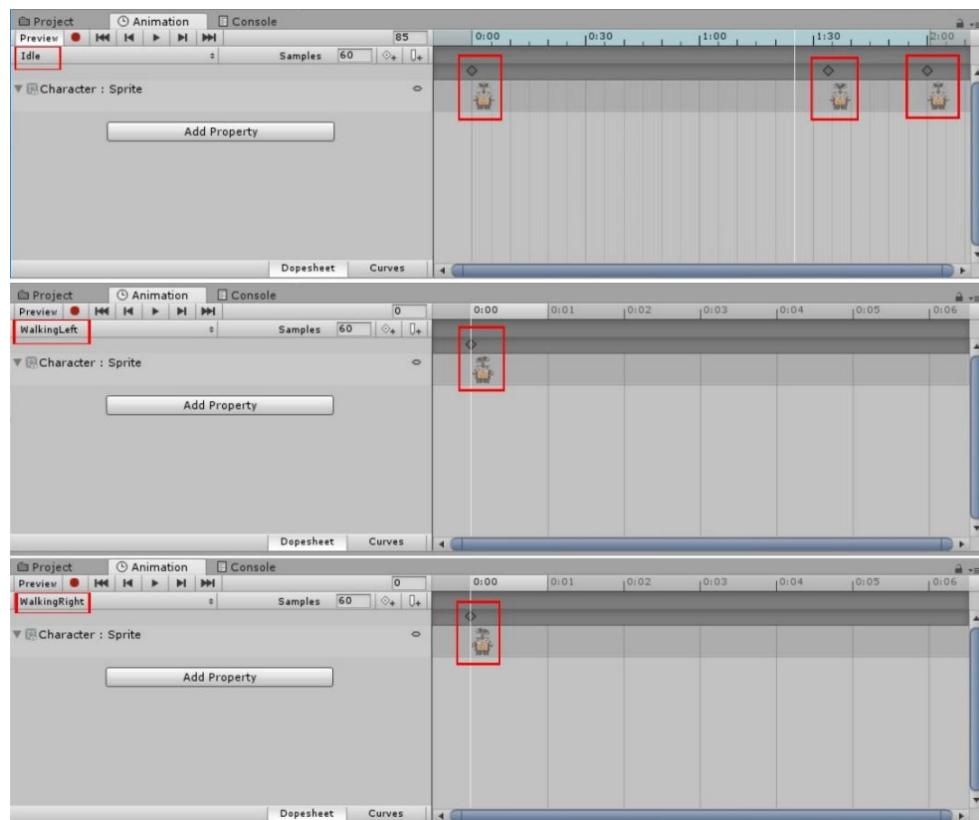


Рисунок 7.7 – Наповнення анімацій

Анімації в Unity створюються за допомогою звичайного таймлайну (timeline), на який в необхідних місцях треба перетягнути кадри-спрайти. Наприклад, для анімації Idle розмістимо на початку спрайт №1 (див. рис. 7.6), далі на відмітці 1:40 – спрайт №2, а потім на 2:00 – знову спрайт №1. Таким чином ми отримаємо ефект підморгування під час відсутності руху персонажу. Аналогічно створюємо анімації для рухів вліво та вправо – в цьому випадку буде достатнім просто розмістити відповідні спрайти на початку анімацій (рис. 7.7).

Тепер виконаємо налаштування переходів між анімаціями. Для цього відкриємо вікно Animator, де побачимо щось схоже на діаграму переходів. Вона містить усі анімації, що є у персонажа, у вигляді станів, між якими можна переходити (рис. 7.8).

Перше, що кидається в очі – перехід між Entry та Idle, тобто Idle зараз є анімацією за замовчуванням і буде програватися постійно під час гри. Крім того є стан Any State, але він використовується для більш складних послідовностей.

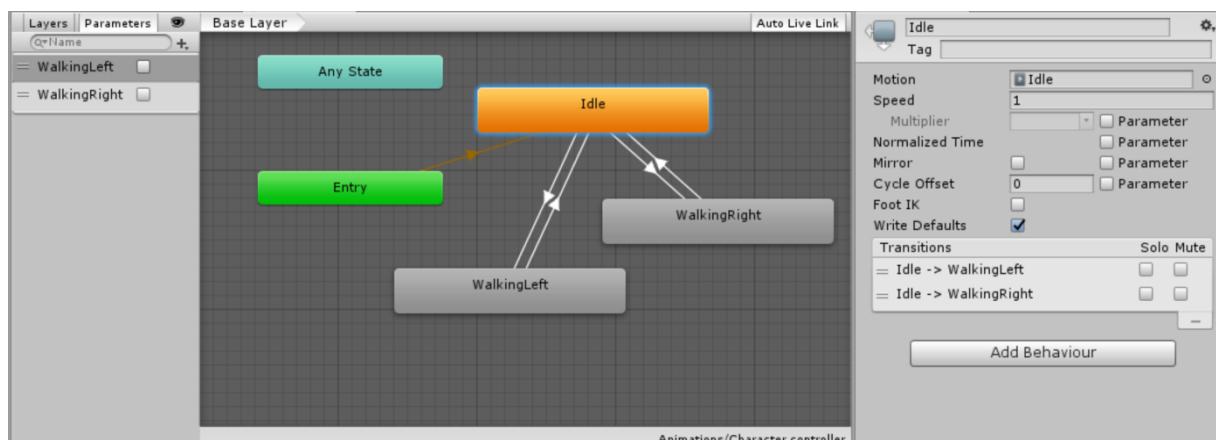


Рисунок 7.8 – Стани анімації

Натиснемо правою кнопкою на Idle, оберемо New Transition (новий переход) та потягнемо його на WalkingLeft. Далі, не знімаючи виділення з щойно створеного переходу, перейдемо до вкладки Parameters в лівій частині вікна Animator. Натиснемо «+» та створимо новий булевий параметр із назвою WalkingLeft – це буде флаг переходу до відповідного стану.

Далі трохи модифікуємо скрипт Control.cs (відображені тільки зміни):

```
public class Controls : MonoBehaviour
{
    ...
}
```

```
private Animator anim;
private bool lf, rf;

void Start()
{
    ...
    anim = GetComponent<Animator>();
}

void Update()
{
    lf = rf = false;

    if (Input.GetKey(KeyCode.LeftArrow))
    {
        ...
        lf = true;
    }
    if (Input.GetKey(KeyCode.RightArrow))
    {
        ...
        rf = true;
    }

    if (moveright)
    {
        ...
        lf = true;
    }

    if (moveleft)
    {
        ...
        rf = true;
    }

    if (lf) {
        anim.SetBool("WalkingLeft", true);
        anim.SetBool("WalkingRight", false);
    }
    else
    if (rf) {
        anim.SetBool("WalkingLeft", false);
        anim.SetBool("WalkingRight", true);
    }
    else {
        anim.SetBool("WalkingLeft", false);
        anim.SetBool("WalkingRight", false);
    }
}
}
```

Ініціалізація аніматора anim в методі Start необхідна для того, щоб можна було з коду дістатися до параметрів анімації за допомогою SetBool. Дві допоміжні змінні-прапорці lf та rf використовуються для фіксації факту переміщення персонажу вправо чи вліво – за допомогою сенсорних чи звичайних кнопок. Скидання прапорців здійснюється на початку методу Update, аналіз та встановлення параметрів анімації відповідно до прапорців –

наприкінці. Цей код можна було реалізувати більш ефективно, але метою була саме наочність демонстрації.

Отже після правки скрипта повернемося до вікна і оберемо перехід з Idle до WalkingLeft. В інспекторі у розділі Conditions натиснемо «+» та оберемо WalkingLeft та true. Тобто цей перехід відбудеться при WalkingLeft == true. Також треба скинути пропорець Has Exit Time – це означає, що аніматор не буде чекати закінчення анімації перед її переключенням (рис. 7.9).

Далі аналогічно створимо зворотній перехід з WalkingLeft до Idle, з тією лише різницею, що умовою переходу буде WalkingLeft == false.

Очевидно, що для стану WalkingRight треба повторити все те ж саме – звичайно, із створенням і налаштуванням параметру WalkingRight. Після цього можна запустити гру та оцінити роботу анімації персонажу.

Звичайно ж анімації можна роботи більш складними, плавними та красивими. Для цього існують так звані таблиці спрайтів, що дозволяють зробити багато проміжних кадрів то досягнути плавності. Крім того, можна створити більше анімацій та прив'язати їх до майже кожної події, що може статися з персонажем – збирання предметів, зустріч із стіною, загибель тощо. Крім того, анімацію можна добавити до будь-яких об'єктів гри, таких як вороги, предмети, платформи. Це значно збільшує час розробки, але й робить гру більш привабливою.

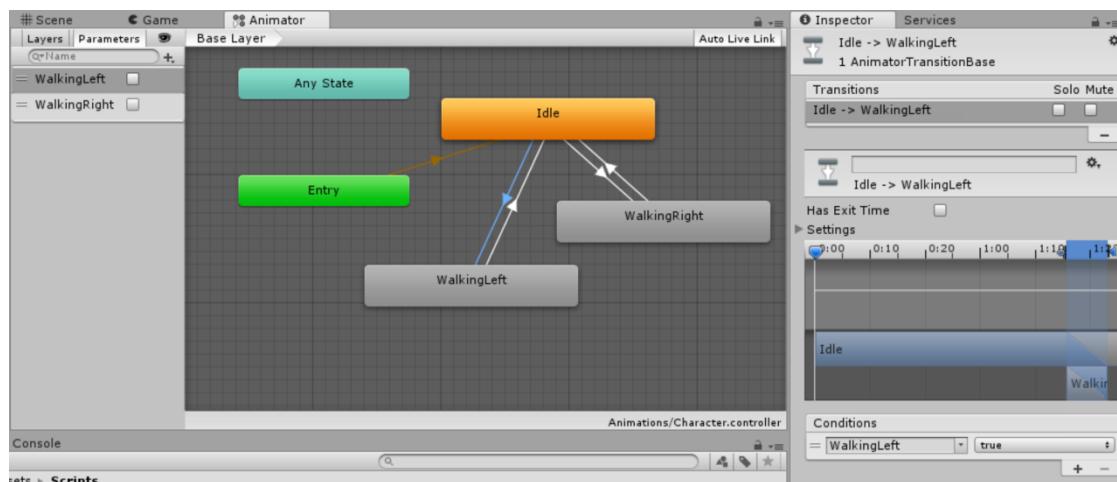


Рисунок 7.9 – Налаштування переходів анімації

#### 7.4.4 Перехід між рівнями гри

На даному етапі розробки у грі явно бракує головної мети. Для платформера це може бути як досягнення відповідної (фінішної) позицій на рівні, так і збирання заданої кількості предметів.

Спробуємо реалізувати другий варіант та припустимо, що для переходу на наступний рівень гри персонажу необхідно набирати 10 предметів. Отже, змінюємо текст на «Crystals: 0 / 41» для наочності та розробляємо нову сцену, до якої буде здійснено перехід після виконання умови.

Збережіть поточну сцену та оберіть File > New Scene. Нова пуста сцена буде створена, а до попередньої завжди можна буде повернутися, натиснувши на неї в папці Scenes. Взагалі, нова сцена може бути чим завгодно – навіть грою у іншому жанрі, але ми обмежимося поздоровленням гравця із виграшом. Цю сцену можна зібрати на власний розсуд з елементів, що вже є в шаблонах, – і це ще одна перевага їх використання в своїх проектах. Ще один спосіб – зробити Save As для першої сцена та видалити усе зайве та додати щось нове – наприклад, кнопку рестарту (рис. 7.10).

Тепер треба виконати перехід до цієї сцени за виконанням умови. Для цього треба було б створити окремий менеджер рівнів, але для нас буде достатньо простого коду в скрипті Control.cs:

```
if (clovers == 10)
{
    SceneManager.LoadScene("2");
}
```

Але треба не забути дописати вверху:

```
using UnityEngine.SceneManagement;
```

та додати нову сцену до білду (File -> Build Settings), бо інакше отримаємо помилку при переході.

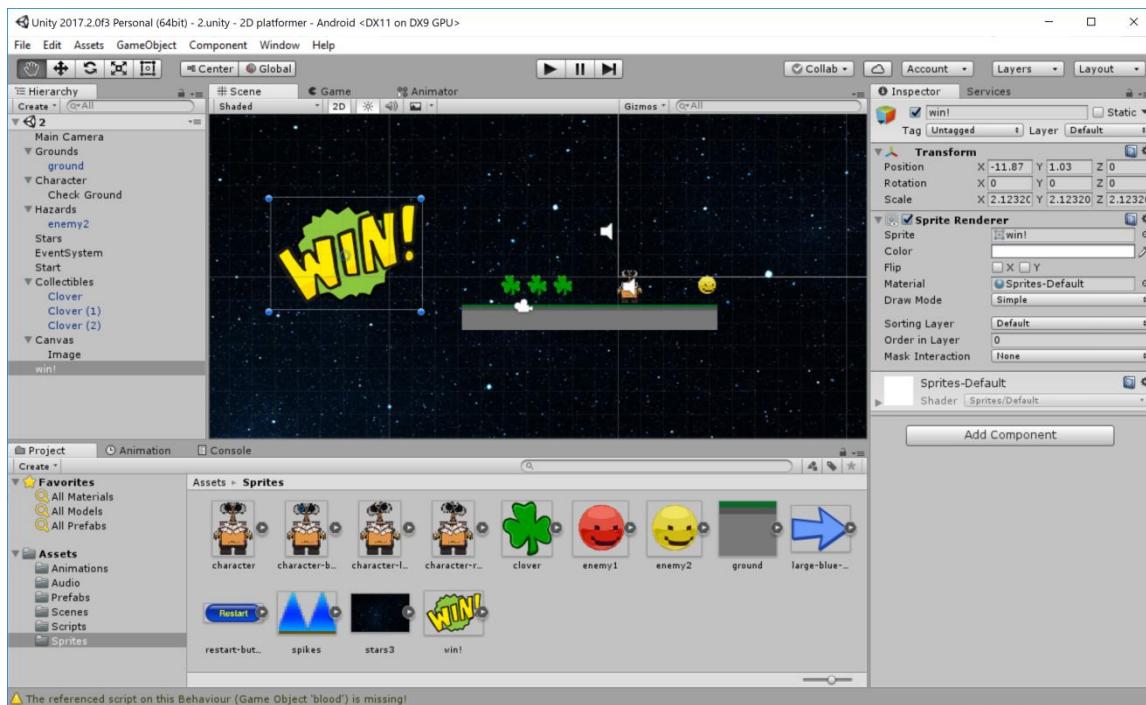


Рисунок 7.10 – Завершальна сцена

Аналогічно можна встановити завантаження нового рівня на тригер onTriggerEnter і тим самим реалізувати двері з одного рівня на інший чи прив'язати перехід до будь-якої події. За описаним алгоритмом створюється послідовність рівнів, заставка, вікно налаштувань тощо.

Очікувані практичні результати роботи:

- третя, підсумкова, версія мобільного ігрового додатку із анімацією головного персонажа, підтримкою та підрахунком предметів, що збираються, та переходом між рівнями при виконання певних умов.

## 8 Заключний практичний семінар

Заключний практичний семінар орієнтовано на обговорення створеного студентом при виконанні комплексу лабораторних робіт 1-3 закінченого мобільного ігрового додатку у середовищі Unity для платформи Android.

Мета практичного семінару:

- оцінити правильність визначення концепції ігрового додатку, відповідності обраному жанру та коректності відображення основних графічних елементів;

- оцінити правильність визначення поведінки та взаємодії ігрових об’єктів відповідно до відомих принципів ігрової механіки;
- оцінити як саме були реалізовані особливості реалізації ігрових об’єктів та механіки відповідно до того, що проект є саме мобільним ігровим додатком;
- оцінити привабливість проекту та надати рекомендації щодо способів подальшого розвитку розробленого мобільного ігрового додатку.

На заключному семінарі кожен студент виступає з презентацією своєї розробки в якій повинен:

1. Надати відомості щодо головної мети та завдань розробки мобільного ігрового додатку.
2. Обґрунтувати основні рішення, прийняті при виконанні лабораторних робіт.
3. Продемонструвати розроблений ігровий додаток на мобільному пристрої із ОС Android.
4. Визначити основні вади і недоліки розробленого додатку.
5. Запропонувати щонайменше п’ять основних покращень, що можуть бути внесені до проекту.

Крім того доповідач повинен надати можливість тестування розробленого додатку. Викладач довільно призначає студента, який тестує розроблений ігровий додаток і виступає як опонент доповідача, висвітлюючи вади та недоліки розробки.

В ході обговорення студенти групи оцінюють правильність визначення основних графічних елементів ігрового додатку, правильність визначення основних подій та дій об’єктів (ігрової механіки) та визначають рекомендації щодо подальшого розвитку розробленого мобільного ігрового додатку. Результати виступу доповідача та «опонента» оцінюються за критеріями, що наведені в додатку В.

За результатами обговорення на практичному семінарі студенти навчаються оцінювати ігрові мобільні додатки, виявляти недоліки та помилки, що були допущені на етапах розробки графічного інтерфейсу, ігрової механіки та тестування ігрового додатку, визначати подальші шаги розвитку додатку з урахуванням мобільної платформи реалізації та загального результатів обговорення.

Успішне засвоєння матеріалів практичного семінару сприяє формуванню у студента наступних соціальних навичок та вмінь: критично

ставитися до результатів особистої роботи, вміти проводити самооцінку виконаних робіт, обґрунтувати той чи інший спосіб діяльності при виконанні практичних завдань, презентувати та аргументувати свої рішення, обговорювати в групі результати роботи. Форми прояву: приймати участь у груповому обговоренні, оцінюванні результатів виконаної роботи, вміння самостійно формулювати та доносити до співрозмовника свої думки, вміння ясно і конкретно висловлювати думки, слухати та розуміти співрозмовника, використовувати прийоми ділового спілкування (публічного мовлення, презентаційних виступів, доповідей) та раніше отримані знання як основу для засвоєння нових знань, дотримуватись етичних норм поведінки, ділового та партнерського спілкування.

## 9 Заключний звіт

### 9.1 Структура звіту

У заключний звіт з виконання комплексу лабораторних роботі повинні бути включені наступні пункти:

1. Титульна сторінка.
2. Мета роботи.
3. Наскізне завдання до комплексу лабораторних робіт.
4. Короткі теоретичні відомості.
5. Попередні ескізи загального вигляду рівня із платформами, ескізи спрайтів головного героя, перешкод, предметів, та ворогів.
6. Опис використаного технічного і програмного забезпечення.
7. Опис проекту для користувача із поясненнями основних моментів ігрової механіки.
8. Опис закінченого проекту з зазначенням всіх об'єктів, подій і дій.
9. Пропозиції щодо подальшого розвитку ігрового додатку.
10. Висновки.

### 9.2 Вимоги до змісту окремих частин заключного звіту

Перш за все, треба за заначити, що звіт повинен представляти собою цільний самостійний документ, який не потребує жодних уточнень та не визиває у читача багато уточнюючих питань.

**Мета роботи** повинна відображати тему комплексу лабораторних робіт а також конкретні завдання, поставлені студенту на період виконання роботи.

За обсягом мета роботи становить від кількох рядків до 0,5 сторінки, але всі заявлені задачі повинні бути повністю виконані.

**Наскрізне завдання** формується у відповідності до поставлених задач, а також із урахуванням особливостей жанру ігрового додатку, який було обрано для виконання комплексу лабораторних робіт.

**Короткі теоретичні відомості** представляють собою короткий теоретичний опис жанру 2D-платформеру з висвітленням основних особливостей і обов'язкових елементів. Матеріал розділу не повинен копіювати зміст методичного посібника або підручника з даної теми, а обмежується викладом основних понять, що необхідні для розуміння подальшого змісту звіту. Обсяг цього підрозділу не повинен перевищувати 1/3 частини всього звіту.

Попередні ескізи ігрових елементів – загального вигляду рівня із платформами, спрайтів головного героя, перешкод, предметів, та ворогів – наводяться для того, що можна було оцінити відповідність результату та початкового бачення гри її розробником. Треба звернути увагу, що мають бути наведені саме початкові ескізи, без урахування досвіду та змін, що було зроблено в початкових планах вже в період розробки ігрового додатку.

**Опис використаного технічного і програмного забезпечення** містить дуже стислий огляд усіх інструментальних засобів, що були задіяні і процесі створення ігрового додатку.

**Опис проекту для користувача** із поясненнями як саме необхідно грati, в чому є основні принципи ігрової механіки, яку мету переслідує та які нагороди отримує користувач. Сам тут буде правильним розмістити скриншоти ключових моментів гри із необхідними поясненнями.

**Опис закінченого проекту для розробника** повинен містити усі ігрові об'єкти, що присутні у грі. Для кожного об'єкту наводиться його назва, призначення та спосіб використання (або поведінка) у грі. Також необхідно навести усі скрипти із необхідними коментарями.

**Пропозиції щодо подальшого розвитку** створеного ігрового додатку повинні містити щонайменше п'ять конкретних пропозицій щодо вдосконалення та розширення ігрового додатку.

**Висновки.** У висновках коротко викладаються результати роботи: отримані результати та навички, здобуті під час виконання комплексу лабораторних робіт.

Заключний звіт оформлюється відповідно до ДСТУ [6]. Зразок оформлення титульного аркуша звіту наведено в додатку Д.

## 10 Література

1. Довідник модуля «Розробка ігрових додатків для OS Android»  
дисципліни «Інструментальна підтримка розробки комп’ютерних ігрових додатків»
2. Siddharth Shekar, Wajahat Karim. Mastering Android Game Development with Unity. – Packt Publishing Ltd., 2017. – 346 p.
3. Valera Cogut. Unity 5 for Android Essentials. Packt Publishing Ltd., 2015. – 200 p.
4. Thomas Finnegan. Learning Unity Android Game Development. – Packt Publishing Ltd., 2017. – 346 p.
5. Mark A Garzone. Software Testing: A Guide to Testing Mobile Apps, Websites, and Games. – CreateSpace Independent Publishing Platform, 2014. – 156 p.
6. ДСТУ 3008-95 «Документи. Звіти у сфері науки і техніки. Структура і правила оформлення».

## Додаток А. Критерій оцінювання лабораторної роботи

<b>Хід виконання</b>	Самостійність виконання	<ul style="list-style-type: none"> <li>• Робота виконується повністю самостійно (без допомоги викладача) з елементами інновації, креативності</li> <li>• Робота виконується повністю самостійно</li> <li>• Робота викується з незначною допомогою викладача</li> <li>• Робота не може бути виконана без допомоги викладача</li> </ul>	3 2 1 0
	Виконання лабораторної роботи в передбачені терміни	<ul style="list-style-type: none"> <li>• Робота виконана з випередженням терміну</li> <li>• Робота виконана у відведений термін</li> <li>• Робота виконана із незначним (менше ніж тиждень) порушенням терміну</li> <li>• Робота виконана із значним (більш ніж тиждень) порушенням терміну</li> </ul>	3 2 1 0
<b>Результат виконання</b>	Відповідність результату теоретичним посилання	<ul style="list-style-type: none"> <li>• Знає та використовує методи, способи, технології інтелектуального аналізу та обробки даних, здійснює опрацювання, інтерпретацію та узагальнення отриманих результатів</li> <li>• Знає але не в повній мірі використовує методи, способи, технології інтелектуального аналізу та обробки даних, здійснює опрацювання, інтерпретацію та узагальнення отриманих результатів</li> <li>• Має часткові знання щодо методів, способів, технологій інтелектуального аналізу та обробки даних. Не в повній мірі здійснює опрацювання, інтерпретацію та узагальнення отриманих результатів</li> <li>• Не здійснює опрацювання, інтерпретацію та узагальнення отриманих результатів</li> </ul>	3 2 1 0
	Збіг отриманих результатів з тестовими розрахунками	<ul style="list-style-type: none"> <li>• Отримані результати повністю співпадають з тестовими розрахунками, виконано аналіз трудомісткості отримання результату</li> <li>• Отримані результати повністю співпадають з тестовими розрахунками.</li> <li>• Отримані результати частково співпадають з тестовими розрахунками.</li> <li>• Отримані результати не співпадають з тестовими розрахунками.</li> </ul>	3 2 1 0

Підсумкова оцінка виконання лабораторної роботи

12 – 10	Перевищує вимоги	4
9 – 8	Відповідає вимогам	3
7 – 5	Переважно відповідає вимогам	2
4 – 3	Майже відповідає вимогам	1
2 – 0	Не відповідає вимогам	0

## Додаток Б. Критерії оцінювання презентації результатів на заключному практичному семінарі

<b>Студент пояснює процес та висновки проекту та демонструє отримані знання.</b>	Визначення теми	<ul style="list-style-type: none"> <li>Чітко визначає тему чи основні питання, їх значення.</li> <li>Чітко визначає тему.</li> <li>Визначає тему.</li> <li>Не визначає тему.</li> </ul>	3 2 1 0
	Доказовість	<ul style="list-style-type: none"> <li>Підтверджує достовірність основних результатів шляхом аналізу відповідних та точних доказів</li> <li>Підтверджує достовірність основних результатів шляхом використання необхідних доказів</li> <li>Підтверджує достовірність результатів шляхом використання мінімально необхідних доказів</li> <li>Не підтверджує результати доказами</li> </ul>	3 2 1 0
	Джерела	<ul style="list-style-type: none"> <li>Використовує технології та інструментарій пошукових систем. Надає докази обширного та достовірного дослідження з використанням численних та різноманітних джерел.</li> <li>Не використовує технології та інструментарій пошукових систем. Надає докази достовірності дослідження за різноманітними джерелами.</li> <li>Надає докази достовірності дослідження за рядом джерел.</li> <li>Надає незначну кількість доказів дослідження (або не надає взагалі) за сторонніми джерелами.</li> </ul>	3 2 1 0
	Вирішення проблеми	<ul style="list-style-type: none"> <li>Генерує нові ідеї вирішення складної проблеми та виходить з поясненнями за межі навчання.</li> <li>Надає різноманітні свідчення вирішення складної проблеми та виходить з поясненнями за межі навчання.</li> <li>Надає деякі свідчення вирішення проблеми.</li> <li>Надає мало свідчень вирішення проблеми та виходу за межі навчання.</li> </ul>	3 2 1 0

	Ідейна база	<ul style="list-style-type: none"> <li>• Поєднує і оцінює існуючі ідеї для формування нових поглядів.</li> <li>• Поєднує існуючі ідеї для формування нових поглядів.</li> <li>• Поєднує існуючі ідеї.</li> <li>• Незначною мірою поєднує існуючі ідеї.</li> </ul>	3 2 1 0
<b>Організація матеріалу презентації Студент демонструє логічність та організованість</b>	Представлення теми	<ul style="list-style-type: none"> <li>• Представляє тему чітко та творчо</li> <li>• Представляє тему чітко</li> <li>• Представляє тему</li> <li>• Не представляє тему</li> </ul>	3 2 1 0
	Утримання фокусу на темі	<ul style="list-style-type: none"> <li>• Підтримує чіткий фокус на темі.</li> <li>• Підтримує фокус на темі.</li> <li>• Деякою мірою підтримує фокус на темі.</li> <li>• Не підтримує фокус на темі.</li> </ul>	3 2 1 0
	Використання переходів	<ul style="list-style-type: none"> <li>• Ефективно використовує поступові переходи для поєднання ключових моментів.</li> <li>• Використовує поступові переходи для поєднання ключових моментів.</li> <li>• Використовує деякі переходи для поєднання ключових моментів.</li> <li>• Використовує деякі переходи, що рідко дозволяють поєднати ключові моменти</li> </ul>	3 2 1 0
	Висновки	<ul style="list-style-type: none"> <li>• Презентація закінчується логічним, ефективним і відповідним висновком.</li> <li>• Презентація закінчується відповідним очевидним висновком</li> <li>• Презентація закінчується очевидним висновком.</li> <li>• Презентація закінчується без висновку</li> </ul>	3 2 1 0
<b>Питання й відповіді</b>	Представлення теми	<ul style="list-style-type: none"> <li>• Має систематичні знання в досліджуваній тематиці, демонструє вміння осмислювати тему і робити обґрутовані висновки. Точно та належним чином на всі запитання аудиторії</li> <li>• Показує широкі знання теми, відповідаючи впевнено, точно та належним чином на всі запитання та зауваження аудиторії.</li> <li>• Демонструє знання теми, відповідаючи точно та відповідним чином на питання та відгуки.</li> </ul>	3 2 1

		<ul style="list-style-type: none"> <li>• Демонструє неповне знання теми, відповідаючи неточно і невідповідно на питання та відгуки.</li> </ul>	0
<b>Використання мови та стиль вираження себе в усній формі</b> <b>Студент ефективно передає свої ідеї</b>	Зоровий контакт	<ul style="list-style-type: none"> <li>• Ефективно використовує зоровий контакт.</li> <li>• Підтримує зоровий контакт.</li> <li>• Є деякий зоровий контакт, але він не підтримується.</li> <li>• Неefективний зоровий контакт.</li> </ul>	3 2 1 0
	Мова	<ul style="list-style-type: none"> <li>• Говорить чітко, по суті і впевнено, використовуючи правильні обсяг і темп</li> <li>• Говорить чітко, використовуючи правильні обсяг і темп</li> <li>• Мова частково чітка, частково нечітка</li> <li>• Мова нечітка, не підходящий темп</li> </ul>	3 2 1 0
	Спілкування	<ul style="list-style-type: none"> <li>• Демонструє знання граматичних, стилістичних особливостей лексики, термінології, лексичних структур. Використовує типові для тематичної комунікації лексико-синтаксичні моделі.</li> <li>• Демонструє знання граматичних, стилістичних особливостей лексики, термінології, лексичних структур. Не використовує типові для тематичної комунікації лексико-синтаксичні моделі.</li> <li>• Частково відсутні знання граматичних, стилістичних особливостей лексики, термінології.</li> <li>• Відсутні знання граматичних, стилістичних особливостей лексики, термінології.</li> </ul>	3 2 1 0
	Залучення аудиторії	<ul style="list-style-type: none"> <li>• Повністю залучає аудиторію.</li> <li>• Старається залучити аудиторію.</li> <li>• Випадкове залучення аудиторії.</li> <li>• Аудиторія не залучається.</li> </ul>	3 2 1 0
	Одяг	<ul style="list-style-type: none"> <li>• Одягнений належним чином</li> <li>• Одягнений неналежним чином</li> </ul>	1 0

Підсумкова оцінка презентації та виступу на заключному практичному семінарі

Сумарний бал		Зарахований бал
44 – 35	Перевищує вимоги	4
34 – 27	Відповідає вимогам	3
26 – 20	Частково відповідає вимогам	2
19 – 10	Майже відповідає вимогам	1
9 – 0	Не відповідає вимогам	0

## Додаток В. Критерій оцінювання заключного звіту

<b>Завдання роботи</b>	Формулювання завдання роботи	<ul style="list-style-type: none"> <li>Завдання на роботу сформульовано докладно, з можливістю перевірки зрозуміння виконавцем</li> <li>Завдання на роботу сформульовано, можливість перевірки розуміння утруднена</li> <li>Завдання на роботу сформульовано частково, перевірка розуміння не можлива</li> <li>Завдання на роботу не сформульовані, перевірка розуміння не можлива</li> </ul>	3 2 1 0
	Очікувані результати	<ul style="list-style-type: none"> <li>Очікувані результати сформульовані повністю, дозволяють легке порівняння з отриманими результатами, наведені повні тестові розрахунки</li> <li>Очікувані результати сформульовані повністю, наведені частково тестові розрахунки, що не дозволяє легкого порівняння з отриманими результатами</li> <li>Очікувані результати сформульовані частково, наведені частково тестові розрахунки, що не дозволяє порівняння з отриманими результатами</li> <li>Очікувані результати не сформульовані, тестові розрахунки не наведені</li> </ul>	3 2 1 0
<b>Опис процедури (процесу) виконання роботи</b>	Опис устаткування	<ul style="list-style-type: none"> <li>Наведено докладний перелік та опис використаного устаткування (комп'ютери, засоби виводу графічної інформації, комунікаційне обладнання, тощо)</li> <li>Наведено перелік та частковий опис використаного устаткування (комп'ютери, засоби виводу графічної інформації, комунікаційне обладнання, тощо)</li> <li>Наведено частковий перелік використаного устаткування (комп'ютери, засоби виводу графічної інформації, комунікаційне обладнання, тощо)</li> <li>Перелік та опис використаного устаткування (комп'ютери, засоби виводу графічної інформації, комунікаційне обладнання, тощо) відсутній</li> </ul>	3 2 1 0
	Опис використаного програмного продукту	<ul style="list-style-type: none"> <li>Наведено докладний перелік та опис використаного стандартного програмного продукту (операційна система, середа розробки, прикладні пакети, тощо)</li> <li>Наведено перелік та частковий опис використаного стандартного програмного продукту (операційна система, середа розробки, прикладні пакети, тощо)</li> <li>Наведено частковий перелік використаного стандартного програмного продукту (операційна система, середа розробки, прикладні пакети, тощо)</li> </ul>	3 2 1 0

		<ul style="list-style-type: none"> <li>Перелік використаного стандартного програмного продукту (операційна система, середа розробки, прикладні пакети, тощо) не наведено</li> </ul>	
	Процедура виконання	<ul style="list-style-type: none"> <li>Усі кроки докладно перераховані та легко можуть бути повторені</li> <li>Усі кроки докладно перераховані, але не можуть бути легко повторені</li> <li>Не всі кроки перераховані і не можуть бути легко повторені</li> <li>Процедури виконання не приведена</li> </ul>	3 2 1 0
	Наведено опис розроблених за завданням програм (алгоритмів)	<ul style="list-style-type: none"> <li>Наведено блок-схеми та докладний текстовий опис розроблених програм (алгоритмів)</li> <li>Наведено блок-схеми та частковий текстовий опис розроблених програм (алгоритмів)</li> <li>Наведено блок-схеми розроблених програм (алгоритмів). Текстовий опис розроблених алгоритмів не наведено.</li> <li>Опис розроблених за завданням програм та алгоритмів не наведено</li> </ul>	3 2 1 0
	Наведено опис отриманих за завданням результаті	<ul style="list-style-type: none"> <li>Наведені отримані результати та їх порівняння очікуваними результатами та з тестовими розрахунками</li> <li>Наведені отримані результати. Їх порівняння з очікуваними результатами та з тестовими розрахунками не наведено.</li> <li>Відсутній опис отриманих результатів</li> </ul>	3 2 0
<b>Заключна частина</b>	Висновки	<ul style="list-style-type: none"> <li>Обґрунтовано підтверджуються теоретичні положення, підтверджена правильність роботи програми, алгоритму</li> <li>Теоретичні положення підтверджуються частково, правильність роботи алгоритму (програми) підтверджена частково</li> <li>Теоретичні положення, правильність роботи алгоритму (програми) не підтвердженні</li> <li>Відсутні висновки за роботою</li> </ul>	3 2 1 0
	Подальший розвиток за напрямом роботи	<ul style="list-style-type: none"> <li>Надано розширеній опис шляхів удосконалення запропонованих при виконанні роботи рішень (процедур, алгоритмів, програм), представлені напрямки подальшого розвитку підходів щодо рішень поставленої проблеми.</li> <li>Надано опис шляхів удосконалення запропонованих при виконанні роботи рішень (процедур, алгоритмів, програм), представлені напрямки подальшого розвитку підходів щодо рішень</li> </ul>	3 2

		<ul style="list-style-type: none"> <li>поставленої проблеми.</li> <li>Надано опис шляхів удосконалення запропонованих при виконанні роботи рішень (процедур, алгоритмів, програм), напрямки подальшого розвитку підходів щодо рішень поставленої проблеми не представлені.</li> <li>Шляхи удосконалення запропонованих при виконанні роботи рішень не запропоновані, напрямки подальшого розвитку підходів щодо рішень поставленої проблеми не представлені.</li> </ul>	1 0
<b>Звіт як технічний текст</b>	Містить: назву, прізвище студента, прізвище викладача, дату	<ul style="list-style-type: none"> <li>Немає відсутніх компонентів</li> <li>Відсутня 1 компонента</li> <li>Відсутні 2 -4 компоненти</li> <li>Відсутні більш ніж 4 компоненти</li> </ul>	3 2 1 0
	Акуратно оформлена	<ul style="list-style-type: none"> <li>Звіт оформленний акуратно</li> <li>Звіт оформленено</li> <li>Звіт оформленено не акуратно</li> </ul>	2 1 0
	Помилки	<ul style="list-style-type: none"> <li>Помилки відсутні</li> <li>Невелика кількість граматичних та синтаксичних помилок (на сторінці не більш ніж 1 помилка)</li> <li>Велика кількість граматичних та синтаксичних помилок (кількість помилок більше ніж кількість сторінок звіту)</li> </ul>	2 1 0
	Технічні аспекти	<ul style="list-style-type: none"> <li>Відсутні помилки в пунктуації, використанні заголовних букв і орфографії.</li> <li>Майже немає помилок в пунктуації, використанні заголовних букв і орфографії.</li> <li>Багато помилок в пунктуації, використанні заголовних букв і орфографії.</li> <li>Численні помилки в пунктуації, використанні заголовних букв і орфографії, що не дають можливості зрозуміти думку автора</li> </ul>	3 2 1 0
	Використання слів	<ul style="list-style-type: none"> <li>Речення побудовані без помилок, всі слова використовуються вірно.</li> <li>Майже немає помилок в структурі речень і використанні слів.</li> </ul>	3 2 1

		<ul style="list-style-type: none"><li>• Багато помилок в побудові структури речень структури і використанні слів.</li><li>• Численні і відволікаючі помилки в структурі речень і використанні слова</li></ul>	0
--	--	---	---

### Підсумкова оцінка заключного звіту

Сумарний бал		Зарахований бал
40 – 36	Перевищує вимоги	4
35 – 30	Відповідає вимогам	3
29 – 25	Відповідає вимогам із зауваженнями	2
24 – 15	Частково відповідає вимогам	1
14 – 0	Не відповідає вимогам	0

**Додаток Д. Приклад оформлення титульної сторінки  
заключного звіту.**

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**

Кафедра прикладної математики та інформатики

**ЗАКЛЮЧНИЙ ЗВІТ**  
з виконання комплексу лабораторних робіт  
з дисципліни:  
«Інструментальна підтримка розробки комп’ютерних ігрових  
додатків.  
Розробка ігрових додатків для ОС Android»

Виконав студент гр. ПЗ 14  
\_\_\_\_\_ Іванов І.І.  
«\_\_\_\_» 2017 р.

Прийняв \_\_\_\_\_  
Викладач кафедри ПМІ  
\_\_\_\_\_ Петров В.О.  
«\_\_\_\_» 2017 р.

Покровськ 2018

## Навчальне видання

Навчальний модуль «Розробка ігрових додатків для ОС Android» з дисципліни «Інструментальна підтримка розробки комп’ютерних ігрових додатків» підготовки студентів освітнього рівня «магістр» спеціальності 121 «Інженерія програмного забезпечення»

Укладачі: Цололо Сергій Олексійович  
Дікова Юлія Леонідівна