Практична робота №1

Створення гри у жанрі 2D Платформер. Робота з Tileset

Мета: набути навичок роботи с Tileset, навчитися створювати сцени для 2D платформеру.

Література

Painting on Tilemaps. https://docs.unity3d.com/Manual/Tilemap-Painting.html

Tilemap. https://docs.unity3d.com/Manual/class-Tilemap.html

Зміст роботи

Для створення рівнів можна скористатися власноруч створеними ігровими активами, а також можна використовувати безкоштовні ассети, що надають різні платформи.

Завдання 1. Ознайомитися з середовищем розробки Unity 6000.0.58f1:

– основними вкладками редактора;

– основними компонентами.

Завдання 2. Розробка концепції. Перш ніж розпочинати розробку, важливо чітко визначити концепцію гри. Які рівні будуть у грі? Які перешкоди та вороги на них зустрічатимуться? Узгодити тему з викладачем.

Завдання 3. Проектування рівнів гри:

– для створення рівнів використати Tileset;

– встановити фонове зображення;

Завдання 4. Програмування.

– додати на сцени додаткові об’єкти (статичні і динамічні);

– додати об’єкт, який керується за допомогою клавіш управління;

Завдання 5. Тестування та налагодження.

– Після того, як буде створено основні елементи гри, потрібно ретельно протестувати її. Перевірте, як працює керування, чи немає помилок та багів.

– Поросіть друзів або знайомих пограти у вашу гру та поділитися своєю думкою. Це допоможе виявити слабкі місця та покращити ігровий процес.

Методичні рекомендації

Покроковий опис створення елементарного Платформеру:

1) У Unity Hub потрібно войти до свого акаунту Unity або створити новий, якщо у вас його ще немає.

2) Натисніть кнопку "New" та оберіть 2D проект. У вікні заповнити поля назви проекту и розташування. Після чого відкриється вікно редактора (рис. 1.).

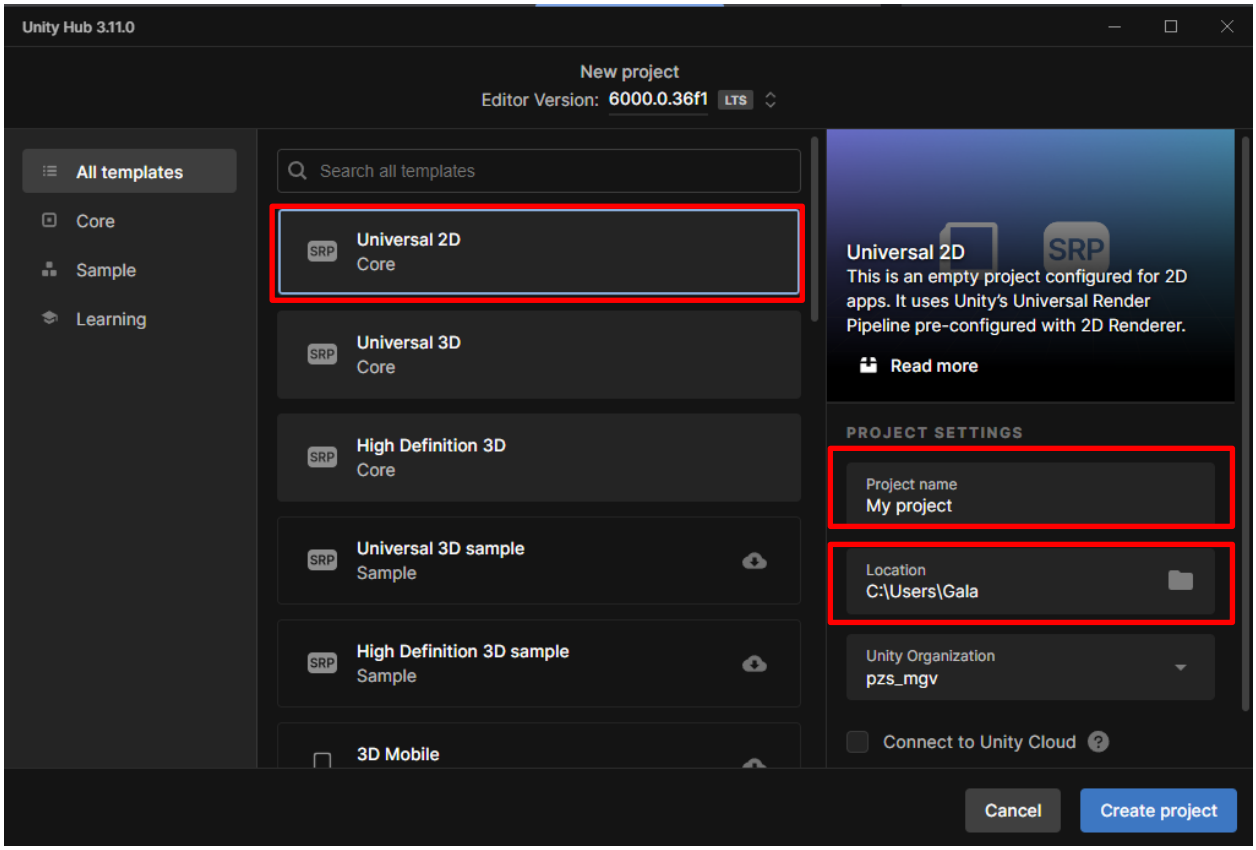


Рисунок 1. - Створення нового проекту

Основними вкладками редактора є:

– Ієрархія об’єктів на сцені;

– Сцена або головне вікно;

– Інспектор об’єктів;

– Інспектор префабів і ресурсів.

Ієрархія об’єктів на сцені - це список усіх об’єктів на поточному рівні.

Інспектор об’єктів показує компоненти та властивості об’єкта, який виділений на даний момент (моделі, текстури, префаби).

3) Вікно редактору (рис.2).

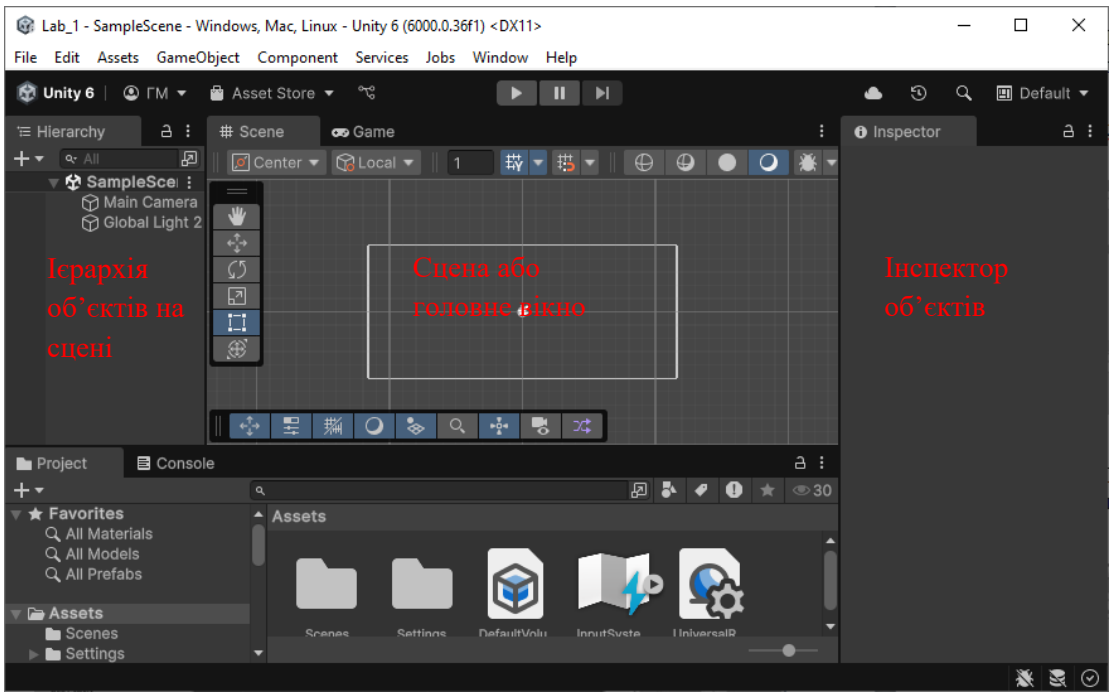


Рисунок 2. - Вікно редактору

4) Перед початком створення гри необхідно підготувати свій проект і сцену. У вкладці Project (Проект) створіть папки: Scenes, Sprites, Scriptes. Ці папки будуть створені в папці Assets вашого проекту.

Папка Assets - це місце, де зберігається все, що додається у вкладку Project. Вона може бути невидимою, залежно від вибраної розмітки вкладки (одна або дві колонки), але ви зможете її побачити, відкривши додаток для експорту файлів.

Ассети проекту. Ассети – це компоненти, які являють собою вже готове рішення, створене вами чи іншими людьми. Це може бути графіка, моделі, звуки, скрипти або плагіни.

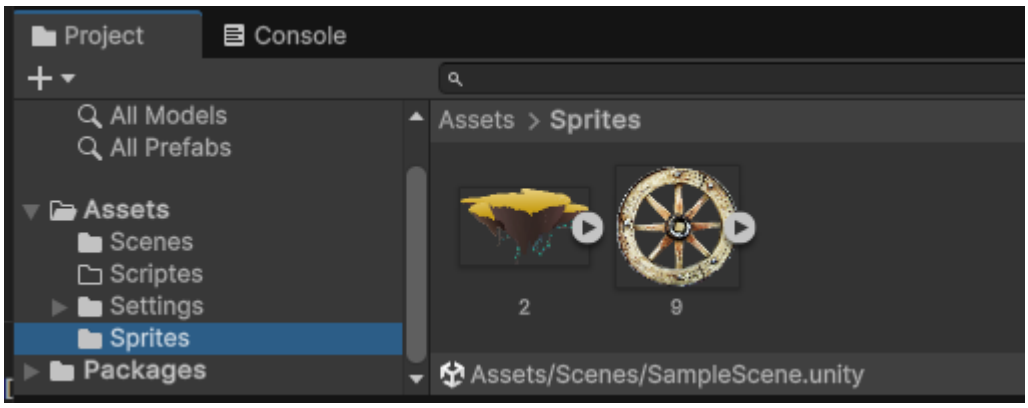
Сцена містить ігровий рівень або меню. Сцени повинні бути збережені вручну.

Sprites - 2D графічні об'єкти - це, по суті, лише стандартні текстури, але існують спеціальні методи для об'єднання та керування текстурами спрайту для ефективності та зручності під час розробки.

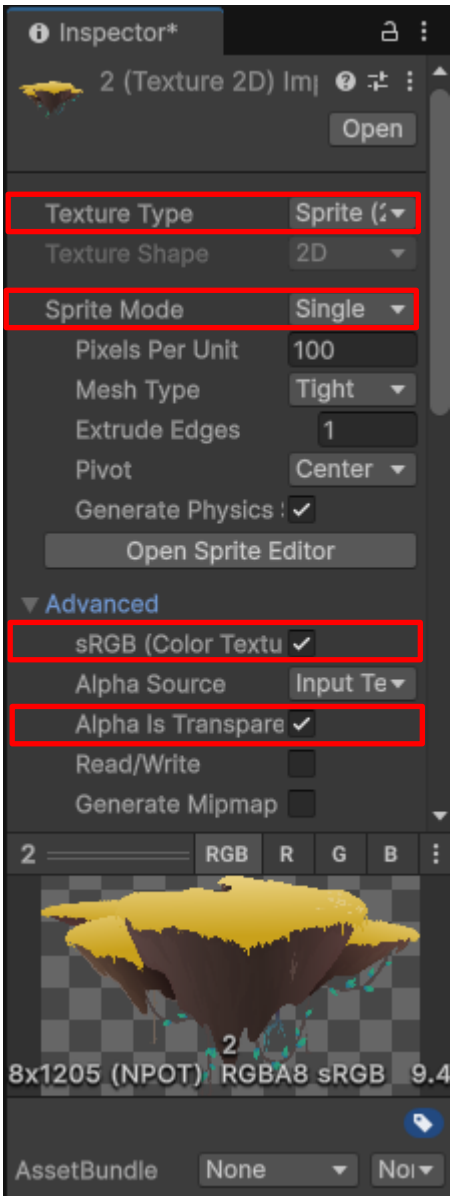
Весь код знаходиться у папці Scriptes – це еквівалент кореневої папки в C# проекті.

Для створення папки виконайте наступні дії: Assets -> Create -> Folder ->ім'я папки.

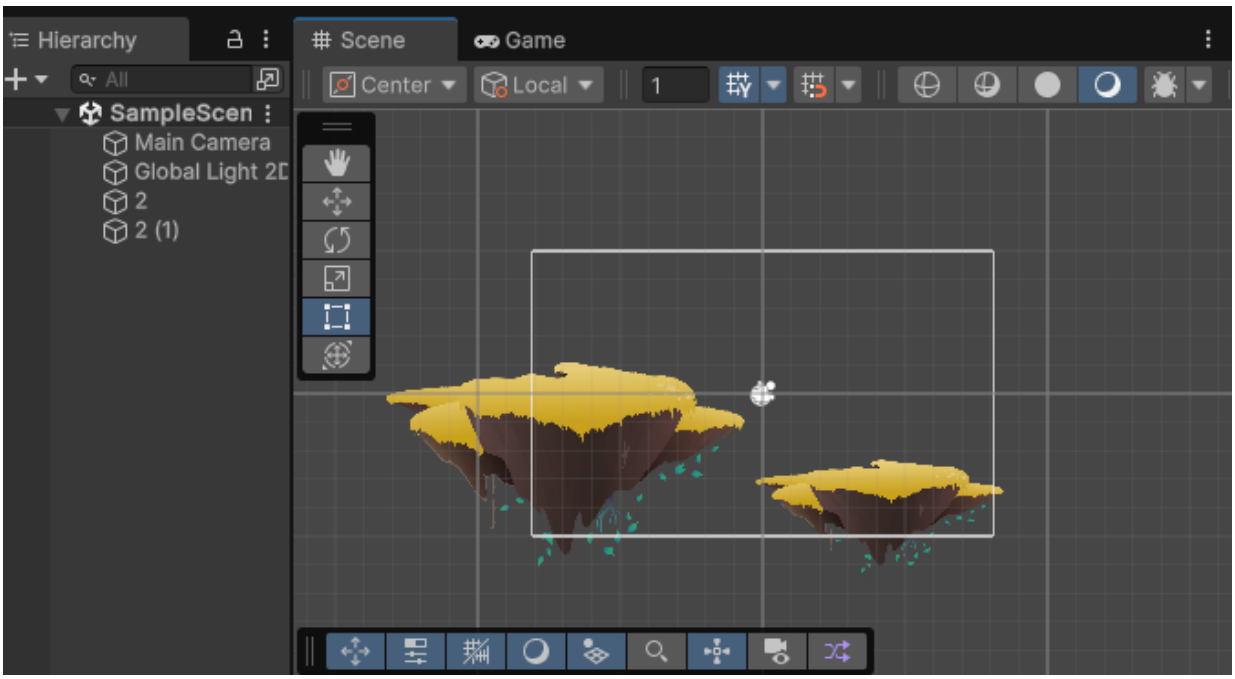
5) До папки Sprites імпортуйте потрібні зображення, краще у форматі png:



Перевірте їх налаштування в Inspector:

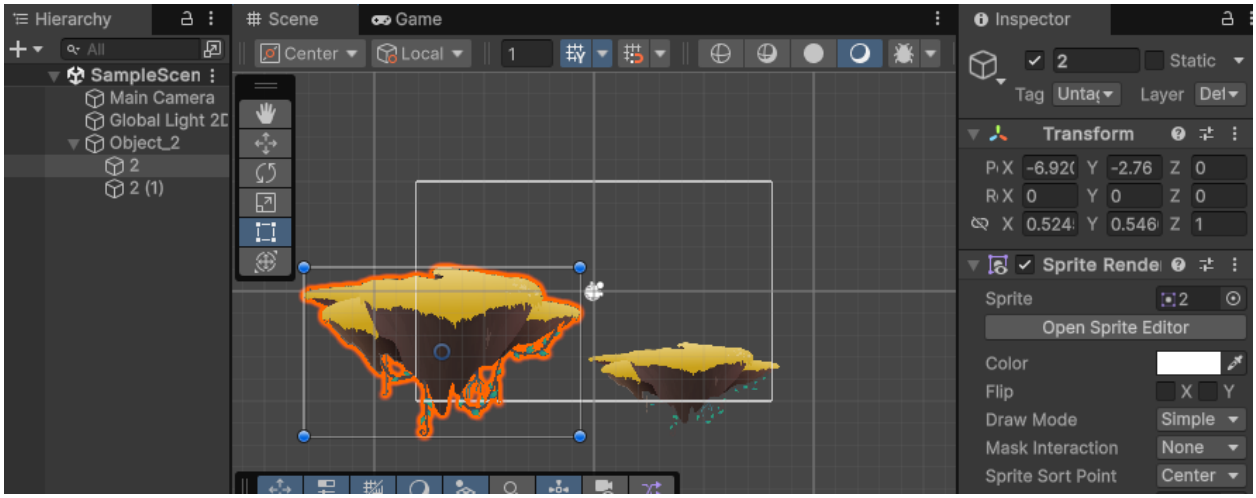


6) Для заповнення сцени необхідно створити порожні об’єкти. В Unity порожній об'єкт можна використовувати в якості папки для інших ігрових об'єктів, куди в подальшому необхідно буде імпортувати зображення з ім'ям потрібні зображення, яке можуть бути основою.



Якщо потрібно зробити копії зображення, скористайтеся кнопками Ctrl+D. Для зміни розміру, розташування зображення, зміни положення використовуємо кнопки:

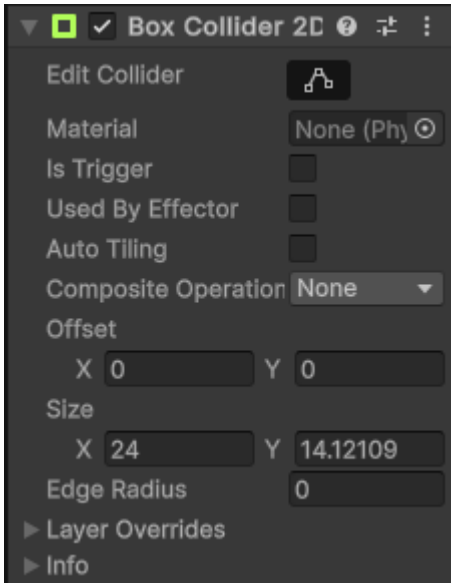
або Inspector -> Transform:



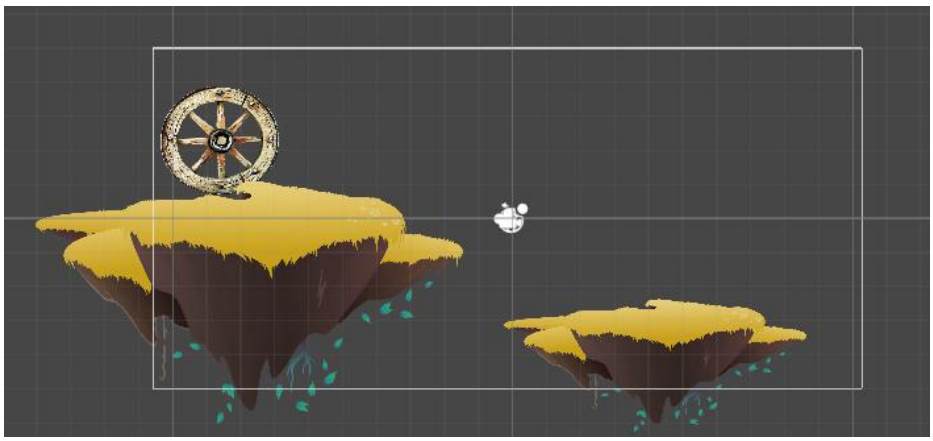
7) Наступним етапом буде створення Colliders. Компоненти коллайдера визначають форму об'єкта для фізичних процесів. Коллайдер невидимий і не обов'язково повинен бути такої ж форми, як і сітка об'єкта.

Для створення коллайдера виконайте наступні дії: Add Component -> Physica 2D -> Box Collider 2D.

Розмір та розташування можна змінити у вкладці Inspector:



Зовнішній вид сцени після розташування всіх ігрових об’єктів:



8) Наступним етапом буде створення скрипта, який надає обертання ігровому об’єкту. Для цього у вкладці Inspector ігрового об’єкта виконайте наступні дії: Add Component -> NewScript -> Name -> Create end Add, або у вкладці Project ->Scriptes -> [KM] -> Create -> c# Script.

Шаблон для базового компонента сценарію:

```
using UnityEngine;

public class AAA : MonoBehaviour
{
    // Start is called once before the first execution of Update after the
MonoBehaviour is created
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

Скрипт взаємодіє з внутрішніми механізмами Unity за рахунок створення класу, успадкованих від вбудованого класу MonoBehaviour.

Кожен раз, коли приєднується скриптовий компонент до ігрового об'єкту, створюється новий екземпляр об'єкта. Ім'я класу і ім'я файлу повинні бути однаковими, для того, щоб скриптовий компонент міг бути приєднаний до ігрового об'єкту.

Функція Start() викликається перед першим запуском будь-яких Update() функцій. Функція Update() - це місце для розміщення коду, викликається один раз за кадр. Це основна подія для промальовування кадру.

За обертання об'єкта відповідає функція transform.Rotate(). Vector3 Створює новий вектор із заданими xyz-компонентами.

Скрипт для обертання колеса:

```
public class AAA : MonoBehaviour
{
    [SerializeField]
    private float speed = 10;
    void Start() { }

    void Update()
    {
        transform.Rotate(new Vector3(0f, 0f, speed));
        //transform.Rotate((new Vector3(0f, 0f, -speed)) *
Time.deltaTime);
    }

      }
```

Time.deltaTime – функція використовується, щоб не залежить від частоти кадра.

Скрипт для руху колеса за допомогою кнопок управління курсором:

```
private Rigidbody2D _rigitbody2D;

void Start () {
    _rigitbody2D = GetComponent<Rigidbody2D>();
}

void Update () {
    float right = Input.GetAxis("Horizontal");

    if (right != 0)
    {
        var newPosition = transform.position;
        newPosition += Vector3.right * right * Time.deltaTime;
        _rigitbody2D.MovePosition(newPosition);
    }
```

Rigidbody2D – Компонент, що поміщає об'єкт під контроль фізичного двигуна.

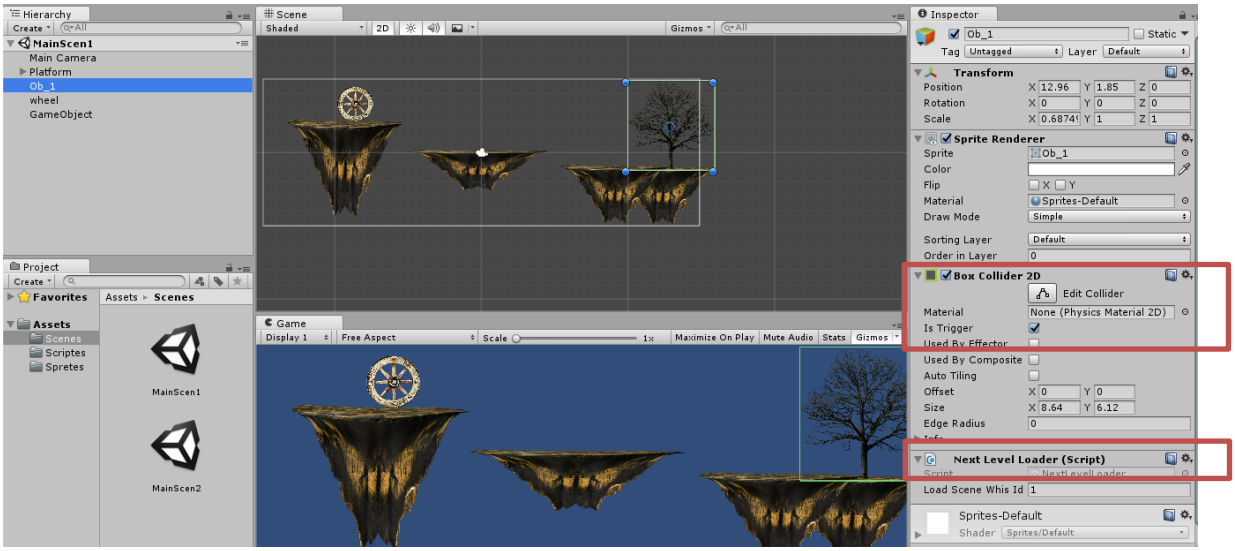
transform.position - Властивість, об'єкта, що містить в собі дані про стан об'єкта в ігровому світі.

Для збереження сцени використовуємо команду File ->Save Scene.

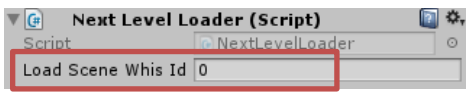
Для переходу з першої сцени на другу створить новий скрипт додавши його на об’єкт, де знаходиться коллайдер тригер:

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

public class NextLevelLoader : MonoBehaviour
{

    public int LoadSceneWhisId = 1;

    private void OnTriggerEnter2D(Collider2D otherCollider)
    {
        if (otherCollider.GetComponent<Wheel>() != null)
        {
            SceneManager.LoadScene(LoadSceneWhisId);
        }
    }

}
```

Приклад: Коллайдер-тригер для зупинки об’єкта і переходу на новий рівень

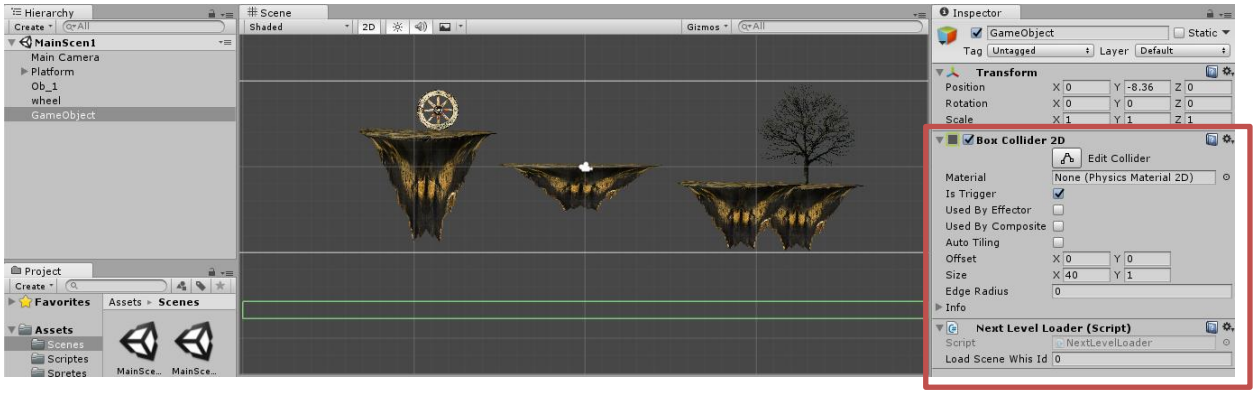


Для переходу на наступний рівень встановить 1 у поле Load Scene Whis Id і 0 для повернення на перший рівень.

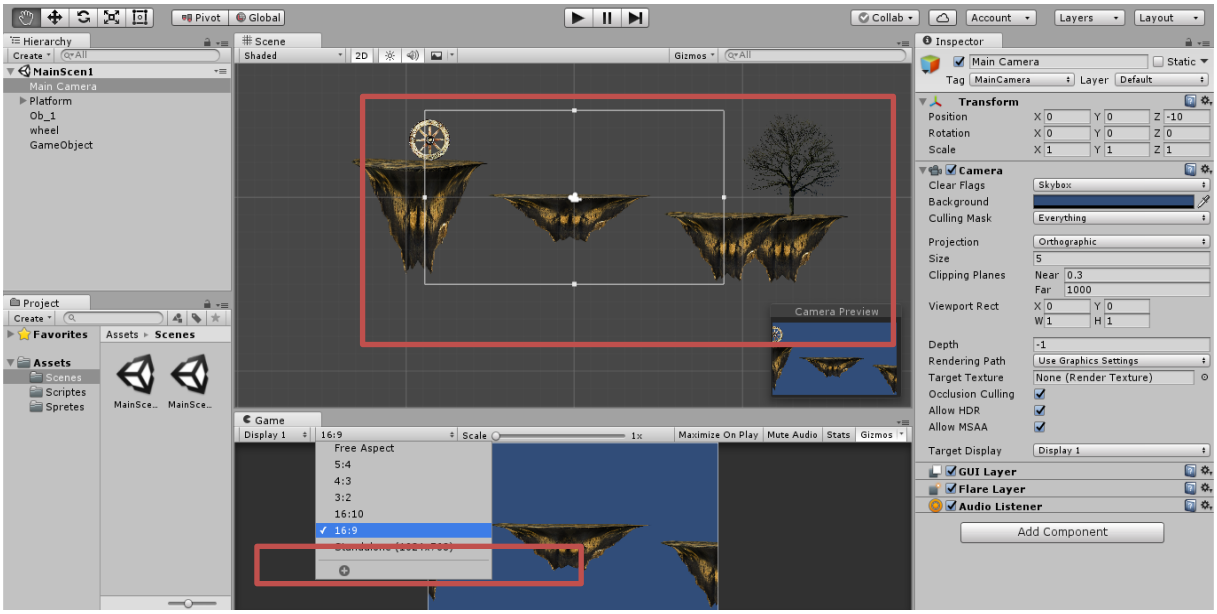


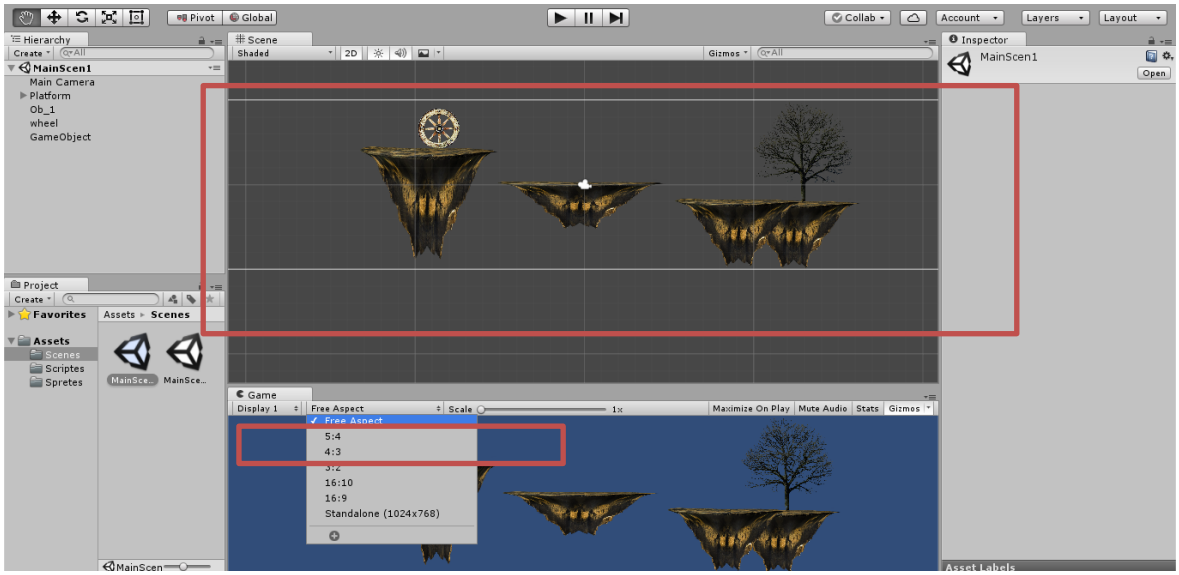
Для того, щоб об’єкт зупиняв вільне падіння, створить на кожному рівні пустий об’єкт (ПКМ-> Create Empty), встановіть на ньому коллайдер тригер та скрипт, що повертає об’єкт на перший рівень.

Коллайдер-тригер для зупинки об’єкта і переходу на перший рівень:



При створенні програмного продукту необхідно звернути увагу також на розташування сцен на екрані:





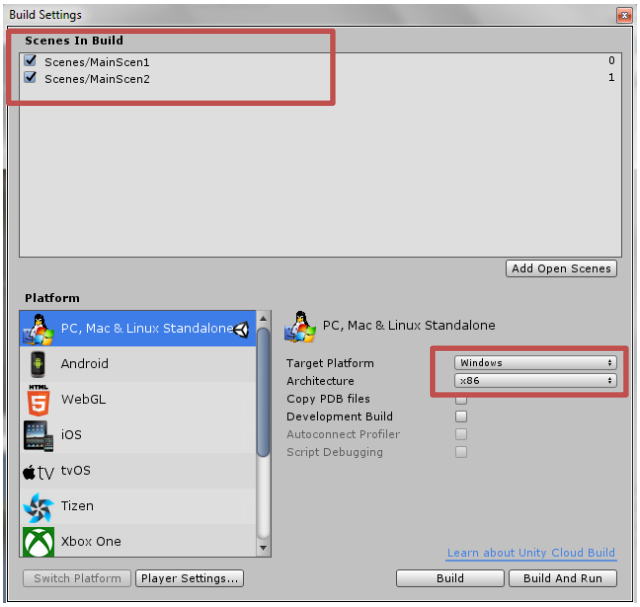
Камери є пристроями, які захоплюють і відображають світ гри. Шляхом настройки і маніпулювання камерами, можна зробити гру цікавою і унікальною. Можна мати необмежену кількість камер в сцені. Можна налаштувати рендеринг камерами в будь-якому порядку, на будь-якому місці екрану, або в деяких частинах екрану.

Для того, щоб слідувати за об’єктом налаштуємо камеру наступним чином:

```
Camera.main.transform.position = transform.position - Vector3.forward * 10;
```

Створені сцени зберігаємо за допомогою команди File>Build Settings. У вікно, що відкриється перетягніть створені сцени.

У вікні Build Settings з’явяться номери сцен, які необхідно запам’ятати:



Використання Tileset

Для оформлення проекту знадобиться стартовий набір елементів, з якого буде складено ігрову сцену - Tileset. По суті, це колекція намальованих елементів, з яких вибудовується все оточення.

Завантажимо Tileset та перевіримо його налаштування (рис.1)

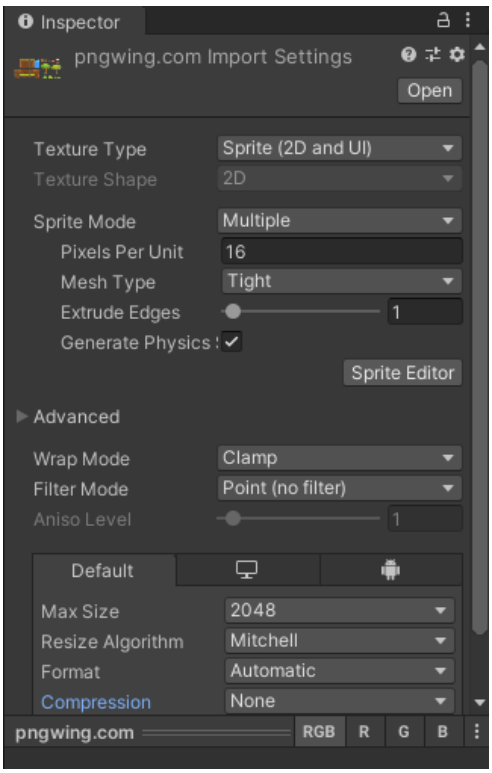


Рис.1. Завантажений Tileset

– *Sprite Mode* (Режим кадрів) => *Multiple* (Множинний), це дозволить багаторазово використовувати кожен кадр (спрайт) тайлсету у процесі створення рівня;

– Pixel per Unit (Піксель на одиницю) краще знизити до 16, у цьому випадку саме цей показник дозволить досягти оптимальної якості картинки.

– *Filter Mode* => *Point* (Точковий), оскільки вибраний сет створено у стилі Pixel Art;

– *Compression* (Стиск) краще вимкнути, вибравши режим None.

Після цього відкриємо **Sprite Editor**, натиснувши відповідну кнопку на панелі Inspector (рис. 2), де буде показано приклад розбивки тайлсета на окремі спрайти.

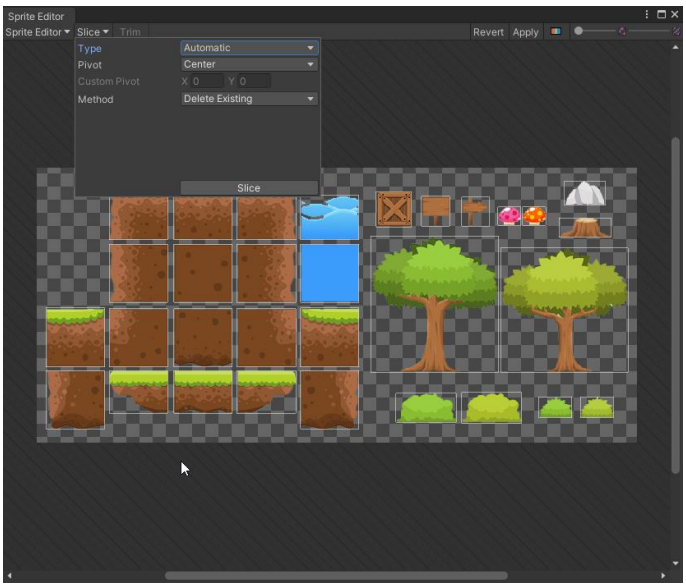


Рис. 2. Тайловій сет відкрито у Sprite Editor

За замовчанням сітки може і не бути. Якщо потрібно задати (або змінити) розміри блоків, відкриваємо панель *Slice* (Нарізати) (рис.3) та вибирати тип сітки, після чого натиснути *Slice*, щоб застосувати.

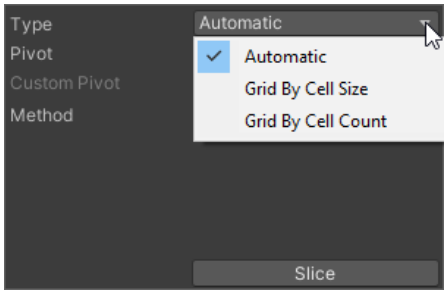


Рис.3. Вибір типу сітки

Перед тим, як вийти з редактора, натискаємо кнопку *Apply*. Результат показано на рисунку 4.

Рис.4. Результат виконання дій у Sprite Editor

Тепер необхідно створити Tilemap - карту шарів, на якій і будуть розміщуватись всі об'єкти в сцені. Для цього натиснемо ПКМ у вікні **Hierarchy** та оберемо 2D *Object* => *Tilemap* (рис. 5).

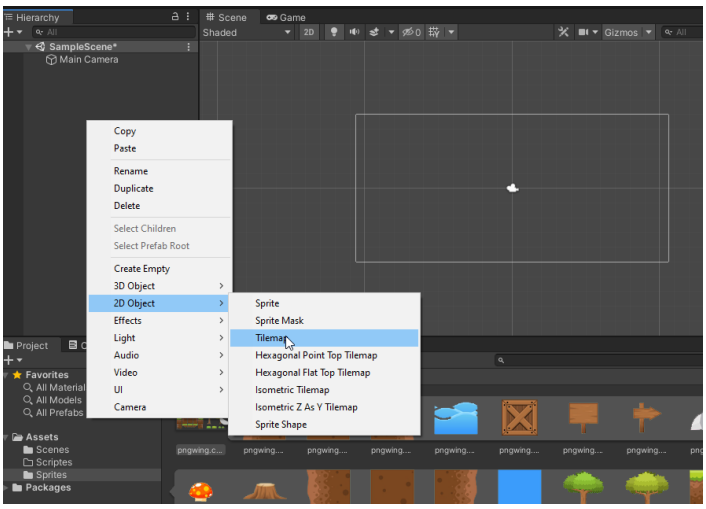


Рис. 5. Створення Tilemap

В результаті буде створена сітка рівня (***Grid***), в яку вкладено карту тайлів. Grid може містити необмежену кількість карт, Tilemap є шаром.

Щоб перетворити обраний Tileset на палітру зразків, виконаємо наступну дію: меню ***Window => 2D => Tile Palette***.

Відкриється додаткове вікно, куди слід перенести всі необхідні для роботи спрайти. Щоб створити нову палітру, клацніть по кнопці ***Create New Palette***, введемо назву, наприклад - land і натиснемо *Create*. При цьому програма запропонує вибрати папку на комп'ютері, у яку зберігається панель. Давайте створимо нову папку Palletes усередині самого аскета (рис. 6).

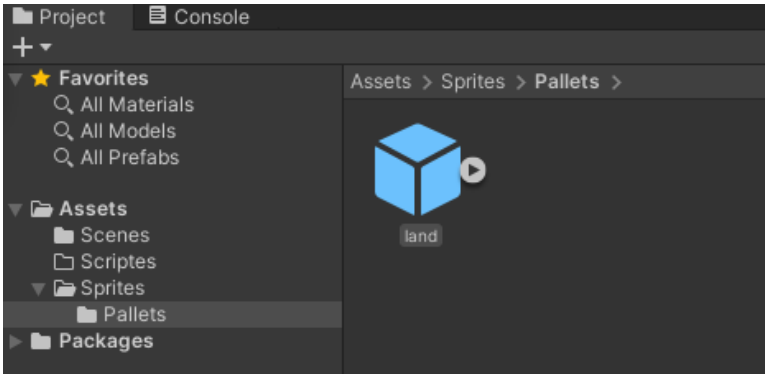


Рис.6. Нова папка Palletes

Тепер необхідно завантажити потрібний Tileset у вікно Tile Palette. Для цього достатньо перетягнути його з папки проекту до робочої області палітри (рис. 7). При цьому тайлсет буде розбитий на окремі кадри.

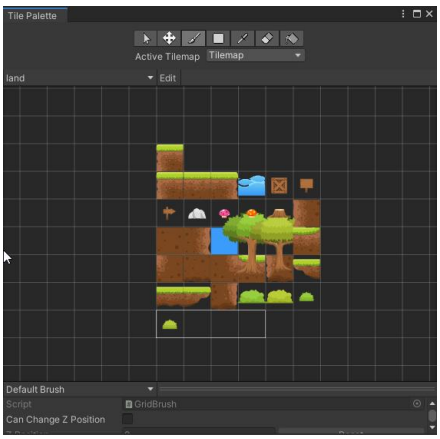


Рис.7. Завантажений Tileset у вікні Tile Palette

В ієрархії проекту з'явиться створена раніше папка, до якої будуть додані всі кадри палітри (Рис. 8)

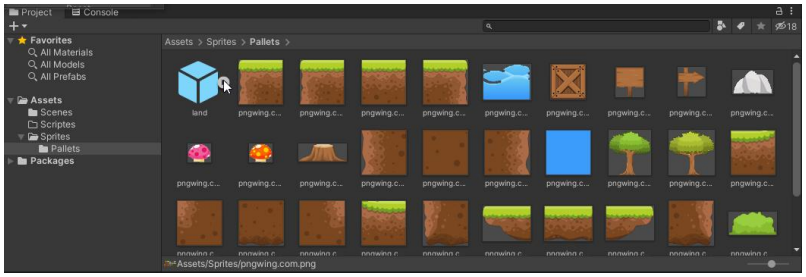


Рис.8. Папка Palettes

Щоб промалювати рівень, необхідно вибрати інструмент ***Brush*** і клацнути по вибраному кадру на панелі. Малювати можна не лише покадрово, а й кількома спрайтами одночасно. Для цього достатньо виділити потрібну область та продовжити складання рівня.

Щоб розмістити нові об'єкти поверх доданих до сцені, або додати фон, потрібно створити ще один шар Tilemap всередині сітки. Перейменуємо його згідно вмісту, наприклад, Tilemap_Background. Щоб зберегти правильну послідовність шарів, необхідно присвоїти кожному порядковий номер — *Order in Layer* (рис. 9)

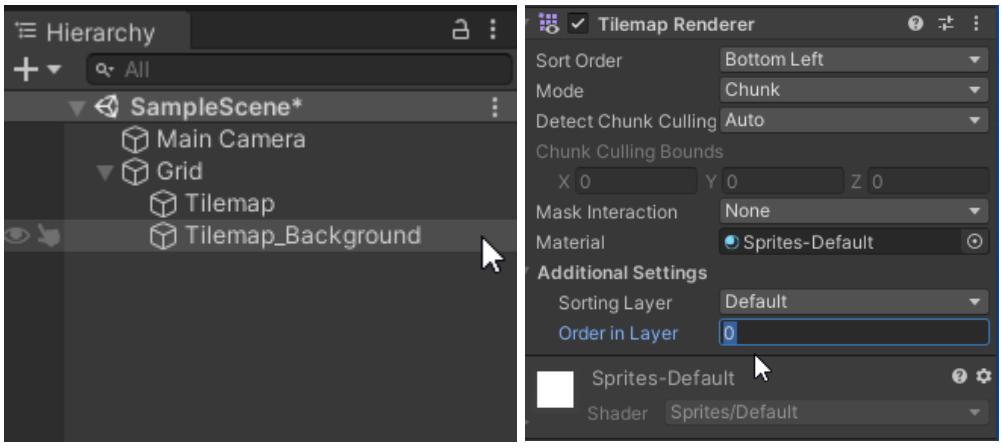


Рис.9. Присвоєння порядкового номеру шару

Контрольні питання:

1. Що таке Unity?
2. Що таке ассет проекту?
3. Що таке спрайт?
4. Що таке скрипт?
5. Що таке сцена?
6. Що таке коллайдер і як його створити?
7. Що таке Tileset?
8. Що таке тайл?
9. Як працюють тайлові карти.
10. Як включати тайлові карти в Unity та налаштовувати сітку.
11. Як додавати в проект спрайти, перетворювати їх на тайли, а потім додавати їх на палітру тайлів.
12. Як використовувати інструменти редактора тайлів для створення рівнів.
13. Як розміщувати тайли на різних шарах.