Tower Defense

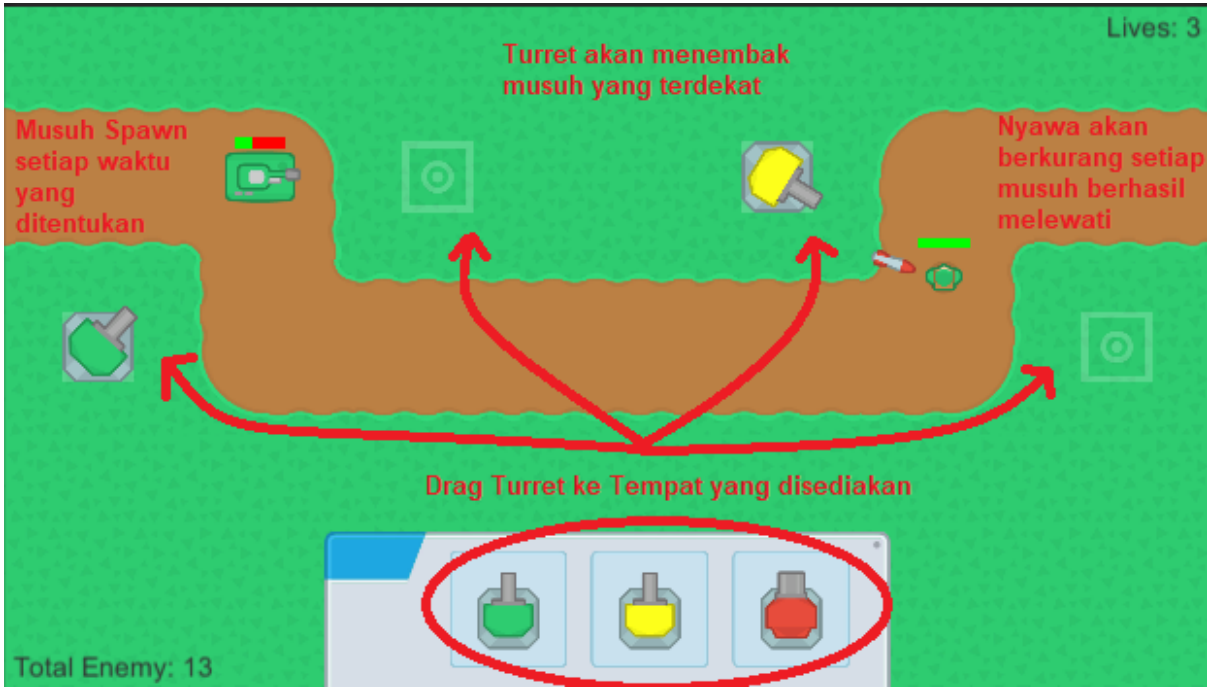
Site: DILo Game Academy
Course: Game Programming Studi Independen
Book: Tower Defense
Printed by: 408 Dewa Sinar Surya
Date: Monday, 4 October 2021, 7:55 PM

Table of contents

1. Pengantar

Pada materi kali ini, kita akan membuat game Tower Defense, yaitu sebuah game dimana player bertugas mencegah setiap musuh untuk mencapai area yang harus dilindungi player. Di sini kita akan menggunakan tema militer, sehingga tower yang digunakan akan tampak seperti turret yang menembakkan peluru kepada musuh dan musuhnya adalah berupa pasukan tentara dan tank.

Berikut adalah penjelasan singkat tentang game dalam bentuk gambar:



Selama menyelesaikan materi ini kita juga akan mempelajari tentang:

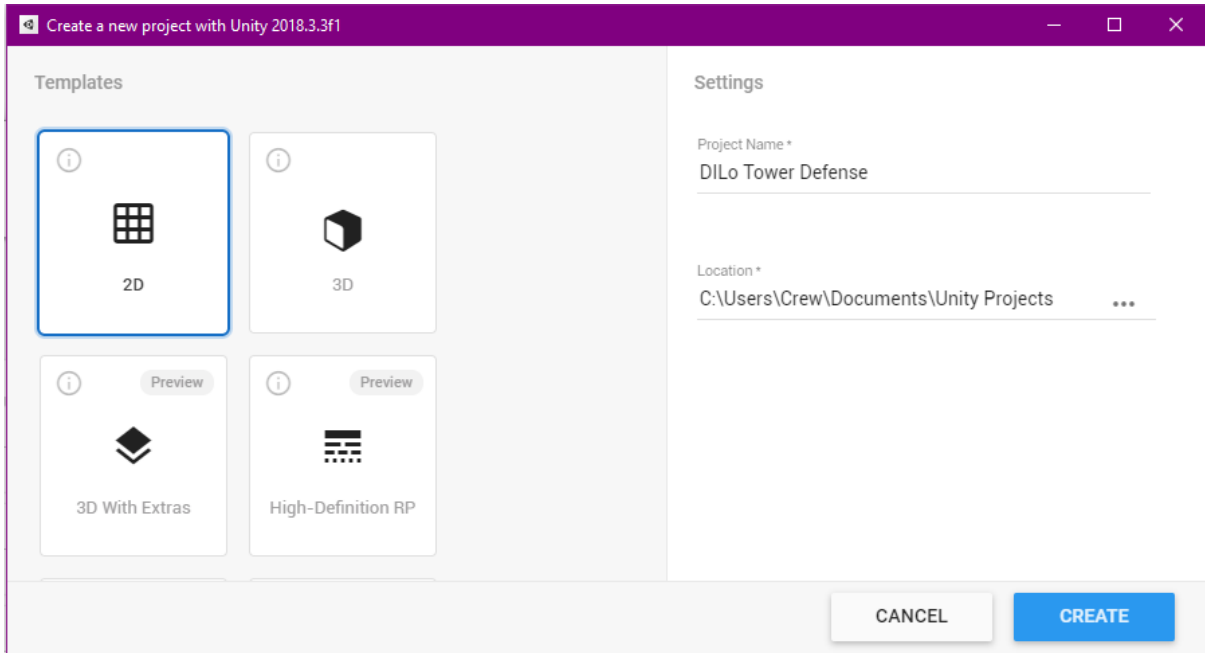
- Bagaimana menggunakan Canvas UI beserta Component-nya.
- Mengenal tentang Base Prefab dan Prefab Variant.
- Membuat sebuah Singleton pada script.
- Memahami konsep serta implementasi Drag and Drop.
- Memahami cara penggunaan event Trigger pada Collision.
- Membuat Object bergerak mengikuti Path, dan

- Menambahkan audio pada Game.

Sebelum masuk ke tahap pembuatan game Tower Defense, download terlebih dahulu asset pada link berikut:

https://drive.google.com/file/d/14FL-NIHYvneAO358Qgg2WfNbhIdqfR2h/view?usp=sharing

Setelah itu buatlah project unity baru menggunakan template 2D.



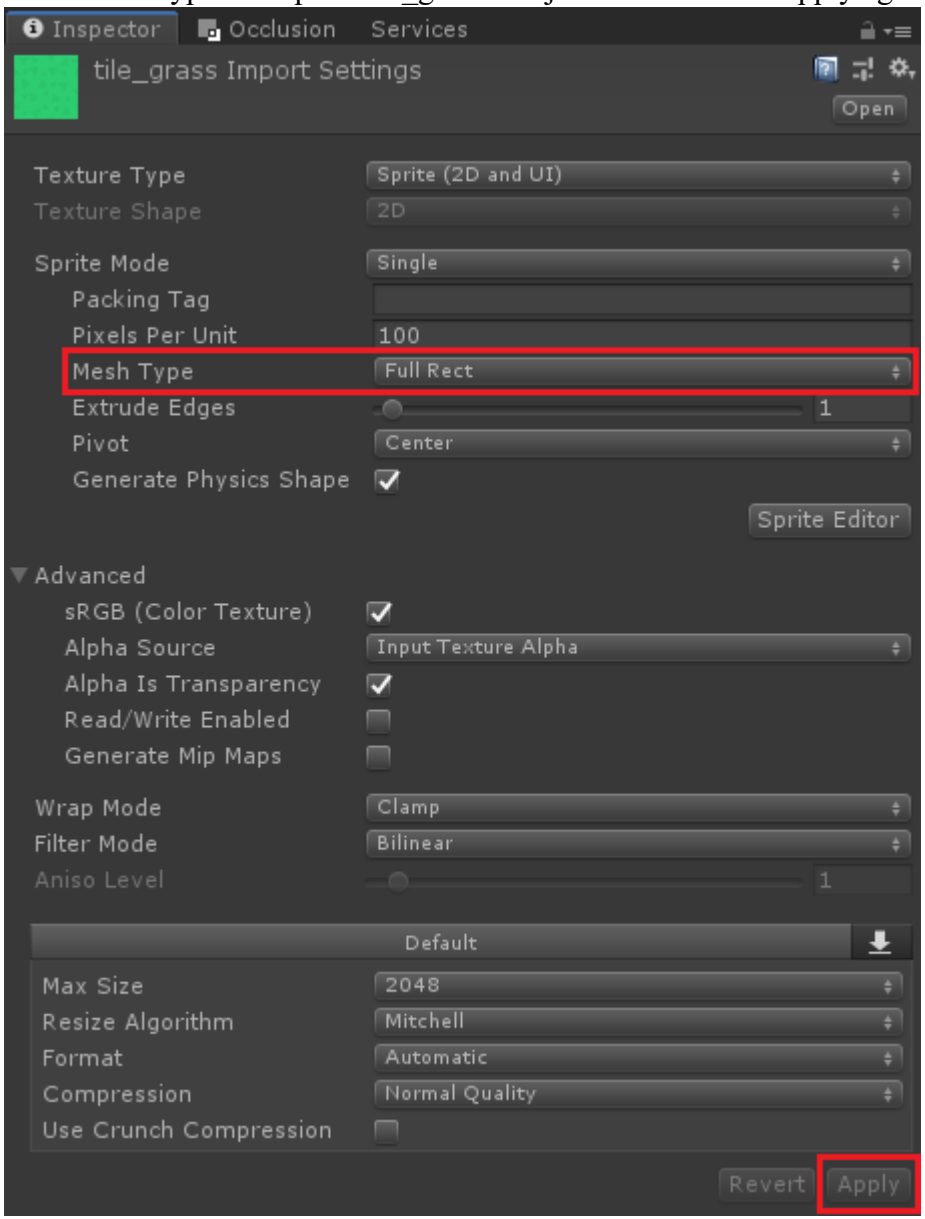
Dan setelah memasuki scene awal, sebaiknya langsung ubah saja nama scene-nya menjadi Gameplay agar lebih menjelaskan scene yang akan kita gunakan.

2. Constructing Level

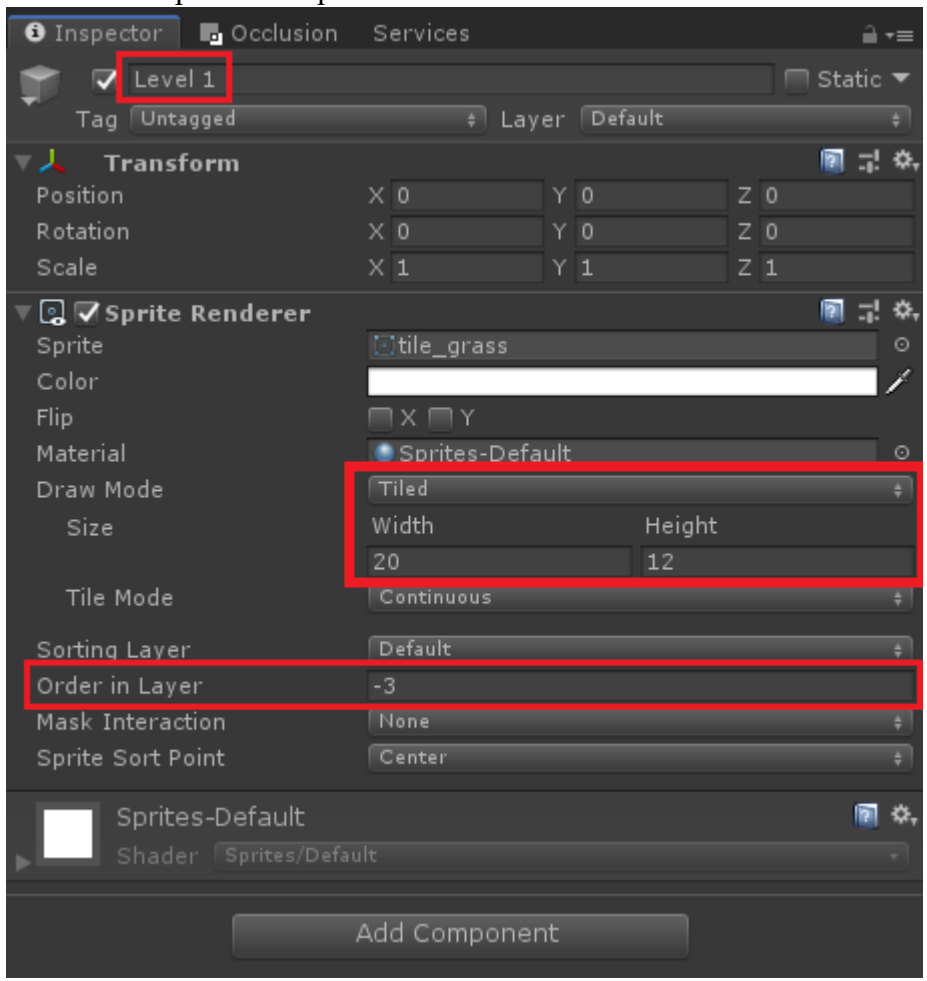
Constructing Level

1. Menjadikan sprite sebagai game object

- Ubah MeshType dari sprite tile_grass menjadi FullRect dan Apply agar sprite ini bisa diulang-ulang seperti tile.

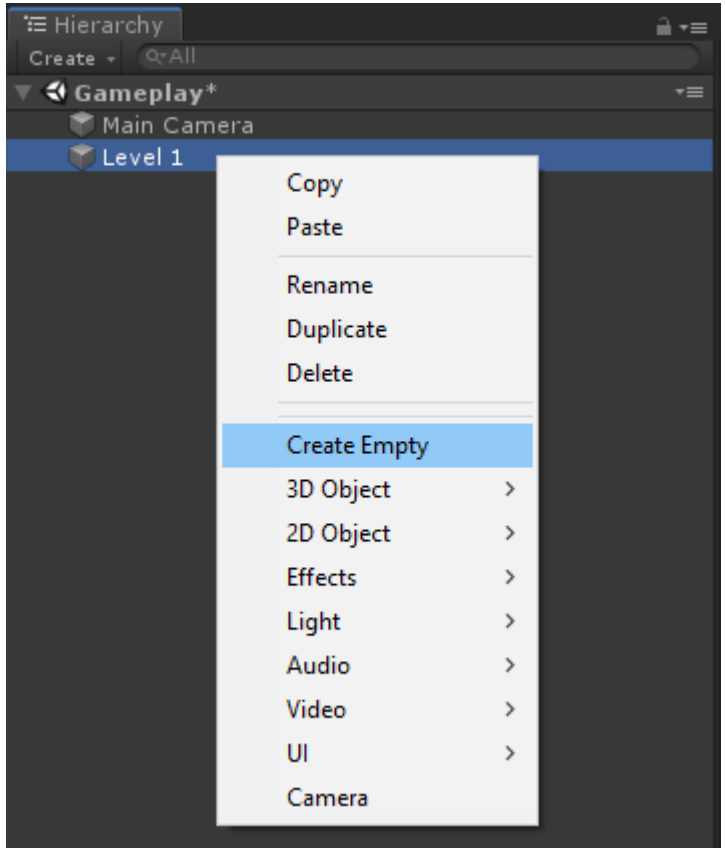


- Drag sprite tile_grass ke dalam Hierarchy dan akan terbentuk game object baru dengan komponen Sprite Renderer.
- Lalu ubah namanya menjadi Level 1 (penamaan game object ini penting untuk memudahkan kita mengenali sebuah game object).
- Ubah Order in Layer pada Sprite Renderer-nya menjadi -3 agar sprite ini selalu berada di paling belakang pada layar.
- Ubah Draw Mode menjadi Tiled dan atur size nya sesuai pada gambar di bawah. Mode Tiled berfungsi untuk membuat sprite menjadi tile based ketika size diperbesar/diperkecil.



2. Membuat game object kosong untuk mengelompokkan

- Klik kanan pada Level 1 dan klik Create Empty.



- Ubah namanya menjadi Road.
- Buatlah game object baru dari sprite road_1 sebagai child dari Road.
- Lalu duplicate hingga menjadi 7 dan ubah masing-masing posisinya menjadi:

Game Object	Posisi (x, y)
road_1 (1)	(-10, 2.5)
road_1 (2)	(-7.5, 2.5)
road_1 (3)	(-2.5, 0)
road_1 (4)	(0, 0)
road_1 (5)	(2.5, 0)
road_1 (6)	(7.5, 2.5)

road_1 (7)	(10, 2.5)

- Kemudian buat game object baru dari sprite road_2.
- Duplicate hingga menjadi 4 dan ubah masing-masing posisi serta rotasinya menjadi:

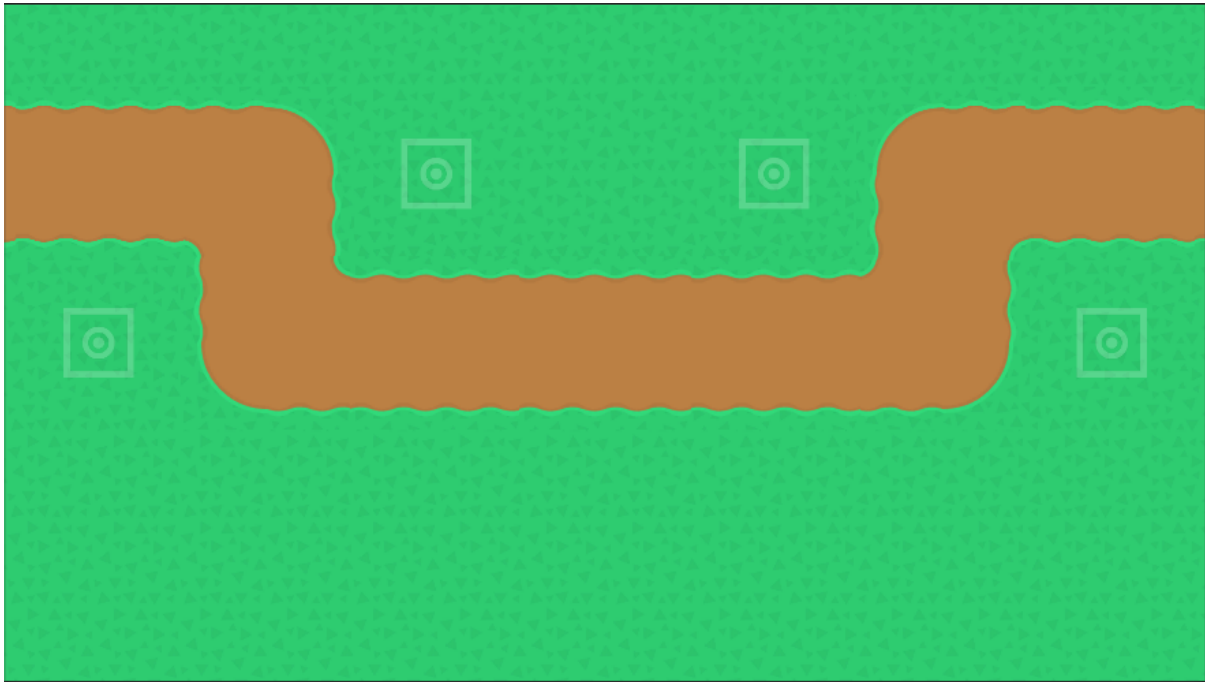
Game Object	Posisi (x, y)	Rotasi (z)
road_2 (1)	(-5, 2.5)	0
road_2 (2)	(-5, 0)	180
road_2 (3)	(5, 0)	-90
road_2 (4)	(5, 2.5)	90

- Lalu blok semua game object road_1 dan road_2 yang telah dibuat dan ubah Order in Layer-nya menjadi -2.
- Buat game object kosong lagi di bawah Level 1 bernama Tower Placement.
- Lalu buatlah game object dari sprite tower_placement sebagai child dari Tower Placement.
- Duplicate hingga menjadi 4 dan ubah masing-masing posisinya menjadi:

Game Object	Posisi (x, y)
tower_placement (1)	(-7.5, 0)
tower_placement (2)	(-2.5, 2.5)
tower_placement (3)	(2.5, 2.5)
tower_placement (4)	(7.5, 0)

- Lalu blok semua game object tower_placement dan ubah Order in Layer-nya menjadi -1.

3. Tampilan setelah menyelesaikan semua tahap 1

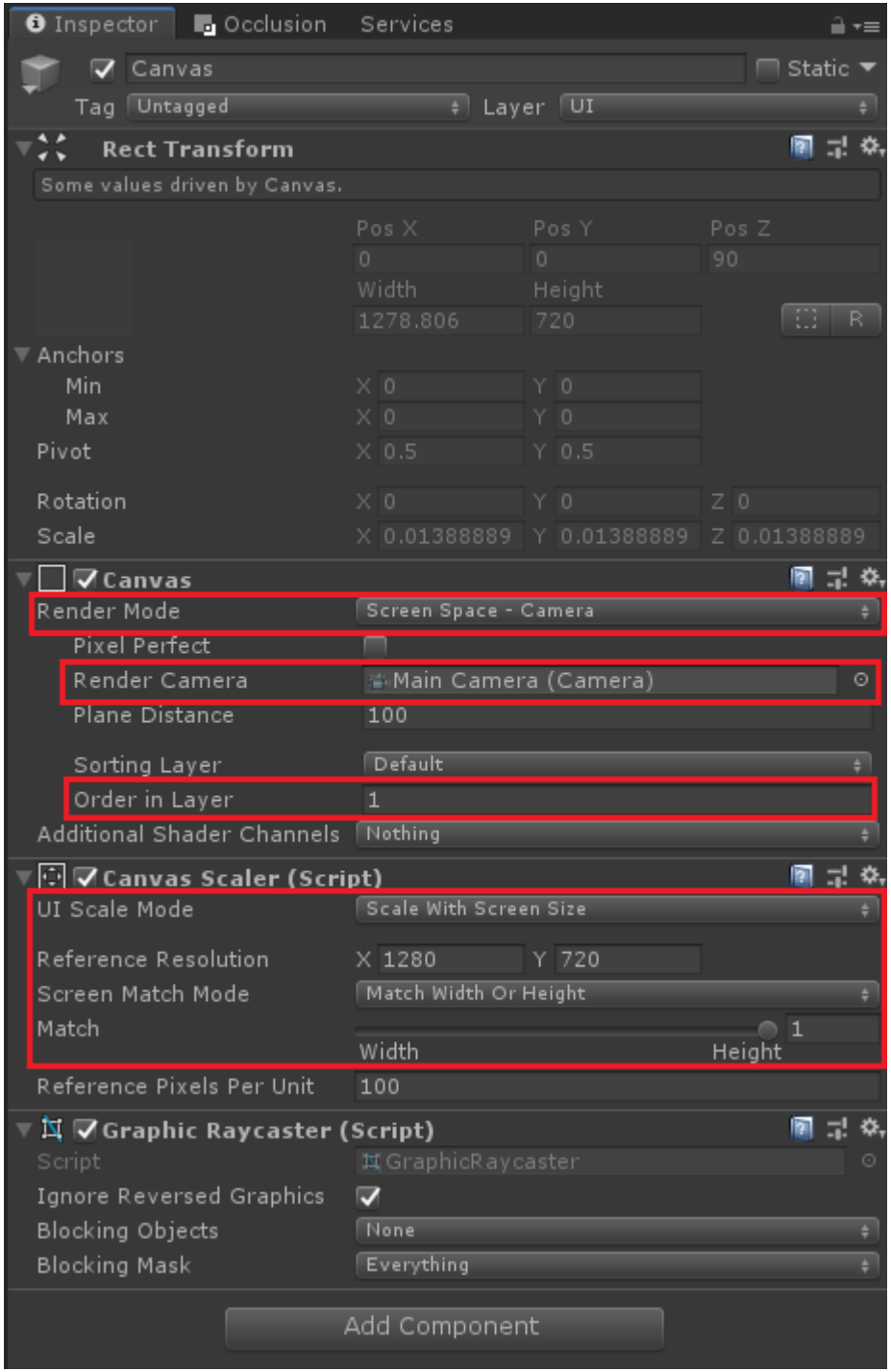
4. 

3. Create Tower Selection UI

Create Tower Selection UI

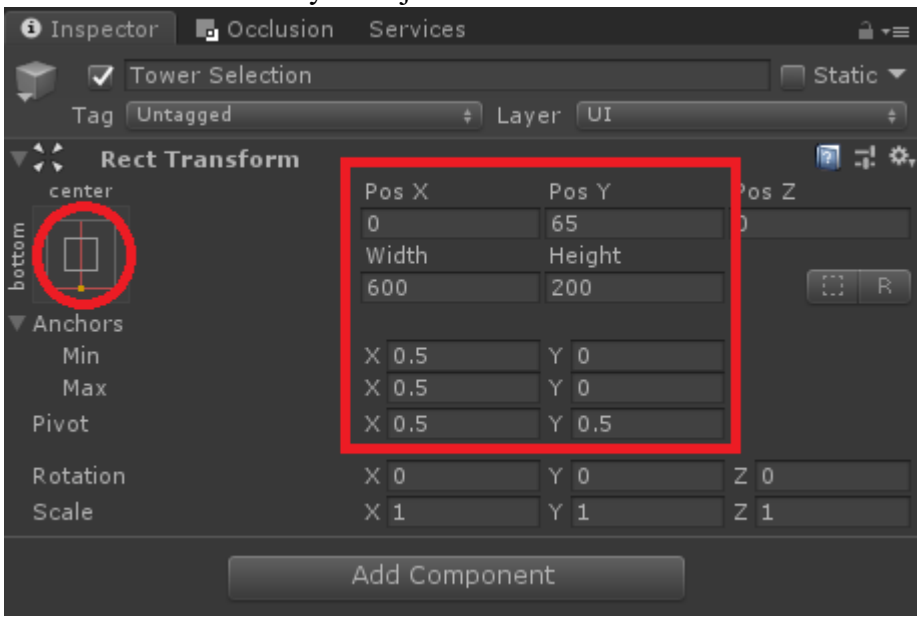
1. Membuat Canvas untuk menampung game object UI

- Pada window Hierarchy, klik Create UI Canvas.
- Pada komponen Canvas Scaler, ubah UI Scale Mode menjadi Scale With Screen Size agar lebih mudah untuk disesuaikan di ukuran pixel layar yang berbeda.
- Lalu ubah nilai-nilai Inspector pada Canvas hingga menjadi:

- 

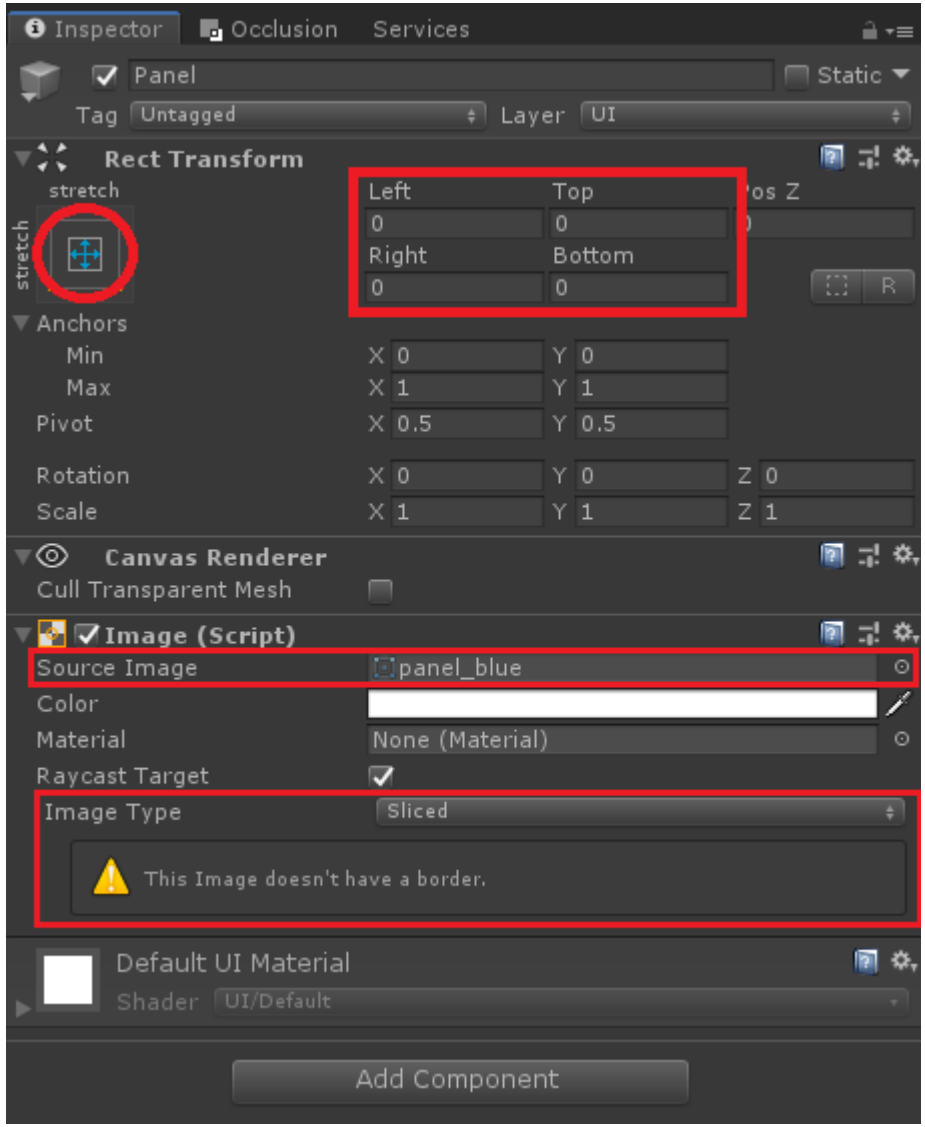
2. Membuat Image UI

- Buat game object kosong bernama Tower Selection pada Canvas.
- Ubah Rect Transform-nya menjadi:

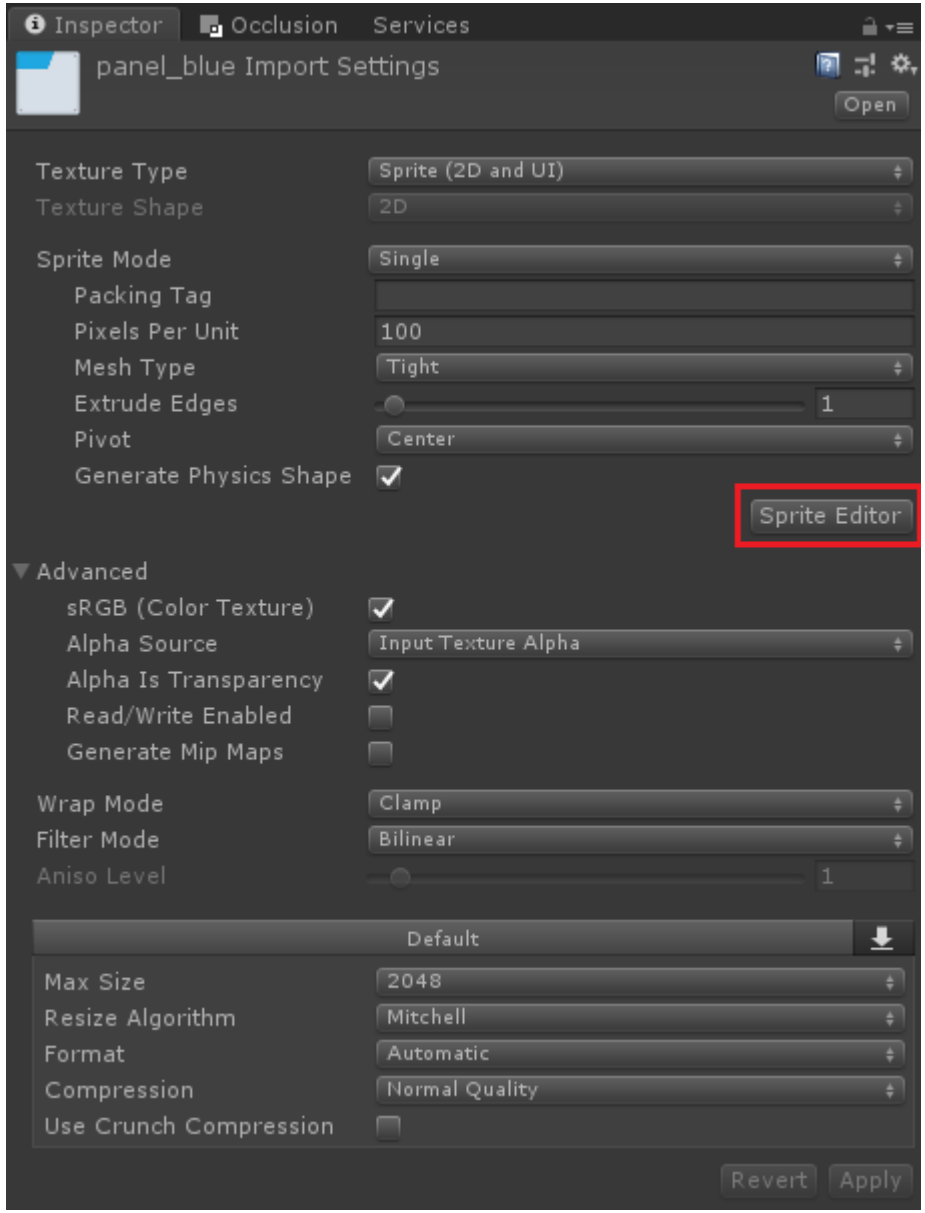


Untuk mengatur RectTransform, biasakan pilih template anchor terlebih dahulu (bagian yang dilingkari), lalu bagian yg diberi kotak pada RectTransform, namanya akan otomatis berubah sesuai kebutuhan anchor tersebut. Jadi, jangan bingung ketika pada awal membuat elemen UI, tetapi nama-nama variable-nya tidak sama.

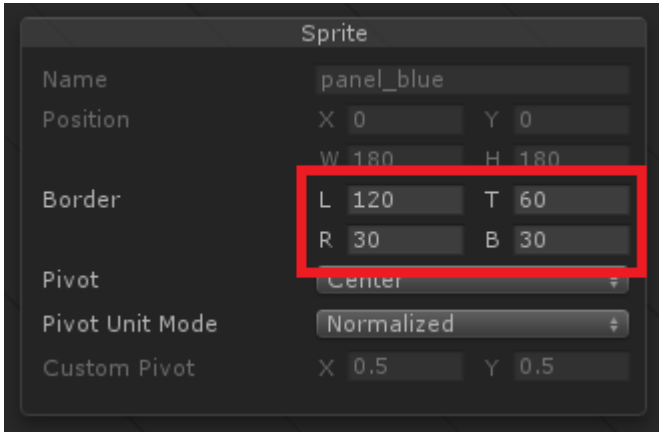
- Klik kanan pada Tower Selection, pilih Create UI Image.
- Ubah namanya menjadi Panel.
- Lalu ubah Source Image pada komponen Image tersebut menjadi sprite panel_blue dan ubah Image Type-nya menjadi Sliced.
- Lalu ubah Rect Transform-nya menjadi:



- Pada gambar di atas terdapat warning This Image doesn’t have a border. Hal yang perlu dilakukan ialah pilih file panel_blue dari window Project. Lalu masuk ke Sprite Editor lewat Inspector-nya.

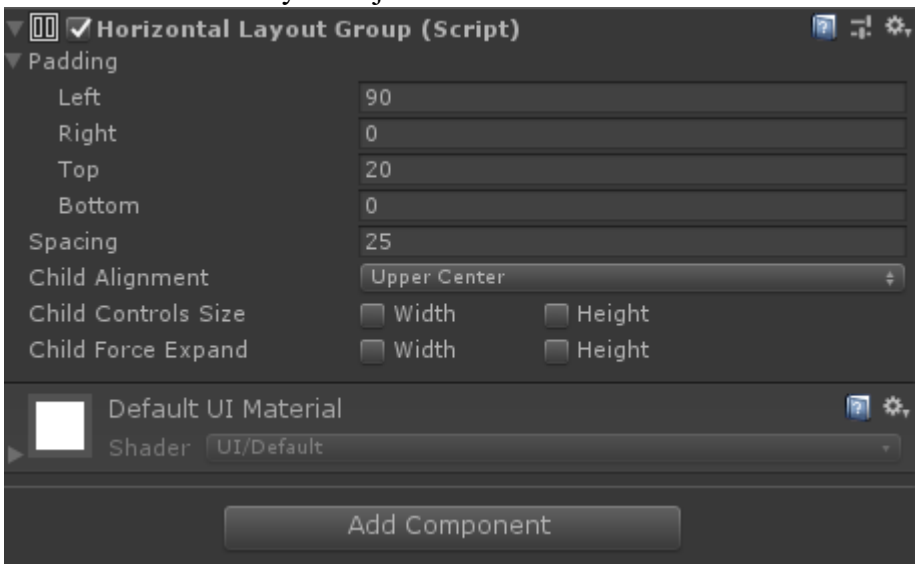


- Ubah pengaturan sprite menjadi seperti ini:



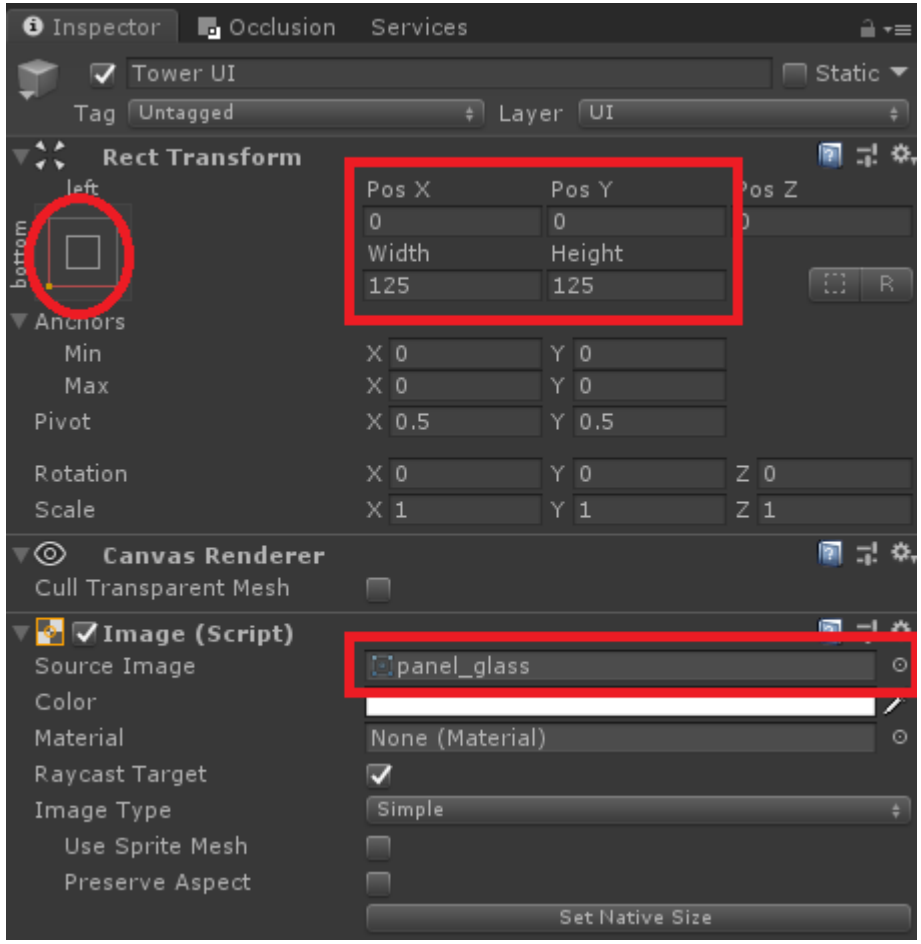
3. Menambahkan komponen Layout

- Pada game object Panel, tambahkan Horizontal Layout Group lewat inspectornya. Fungsinya ialah merapikan setiap child dari game object ini secara otomatis menjadi berderetan secara horizontal.
- Lalu ubah nilai-nilainya menjadi:

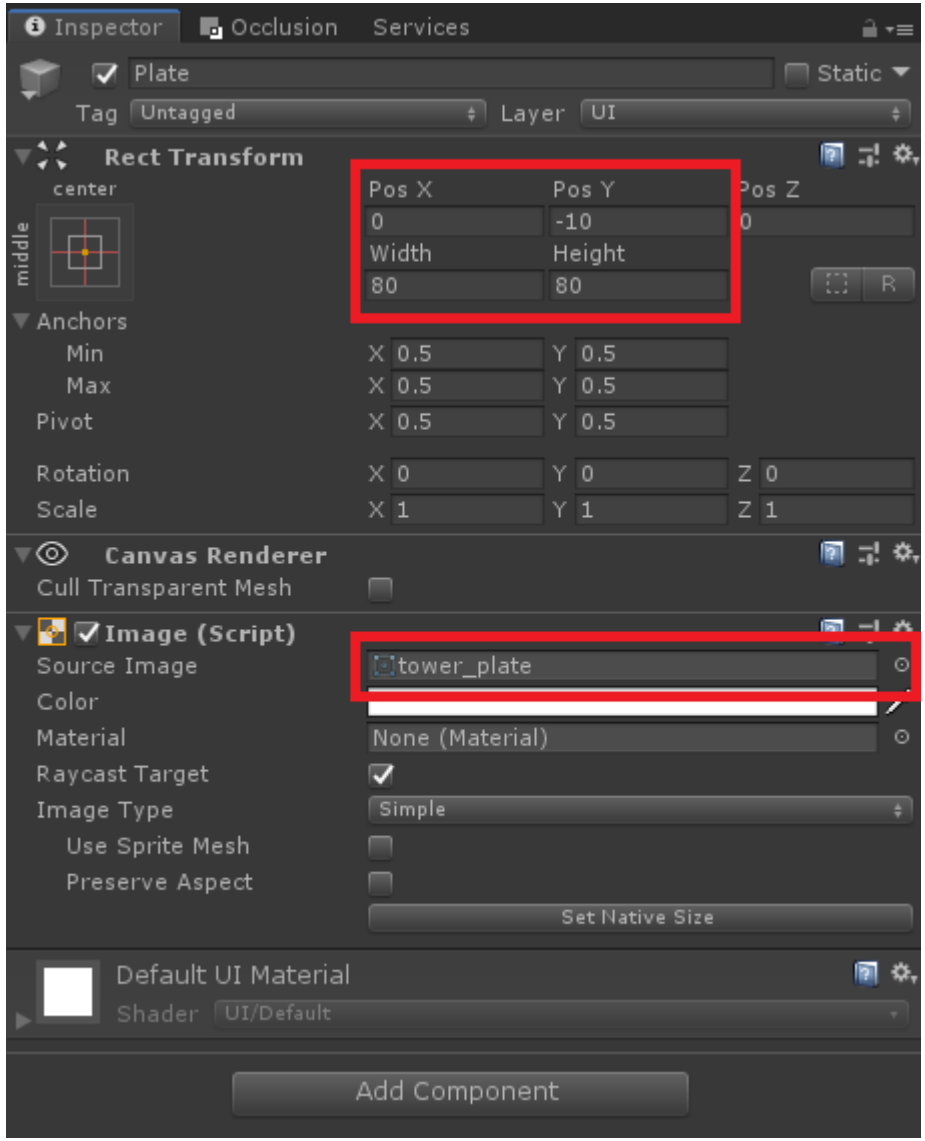


4. Membuat Prefab UI

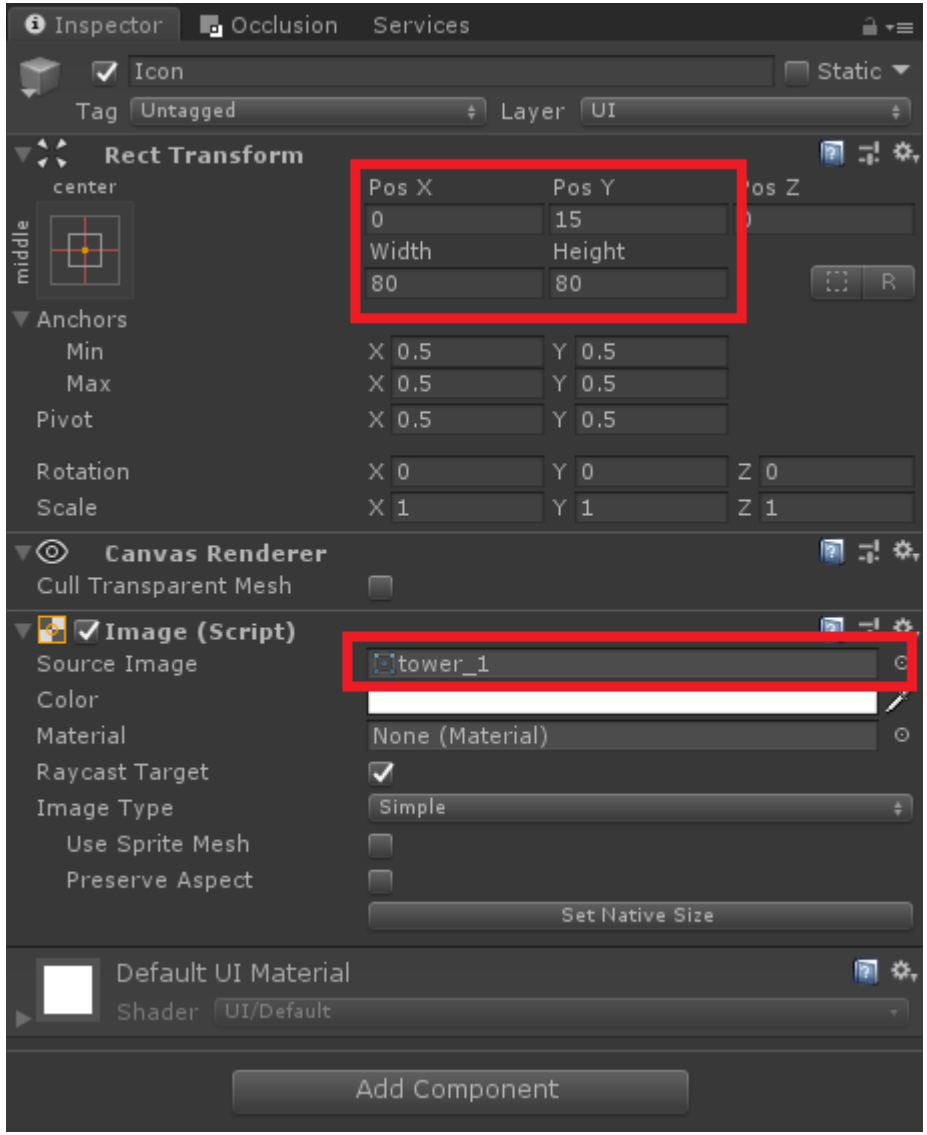
- Pada game object Panel, buat Image UI baru sebagai child-nya.
- Ubah namanya menjadi Tower UI dan masukkan sprite panel_glass ke dalam Image tersebut.



- Buatlah Image UI baru lagi sebagai child Tower UI, ubah namanya menjadi Plate dan ganti sprite-nya menjadi tower_plate dan ubah Rect Transform-nya menjadi:



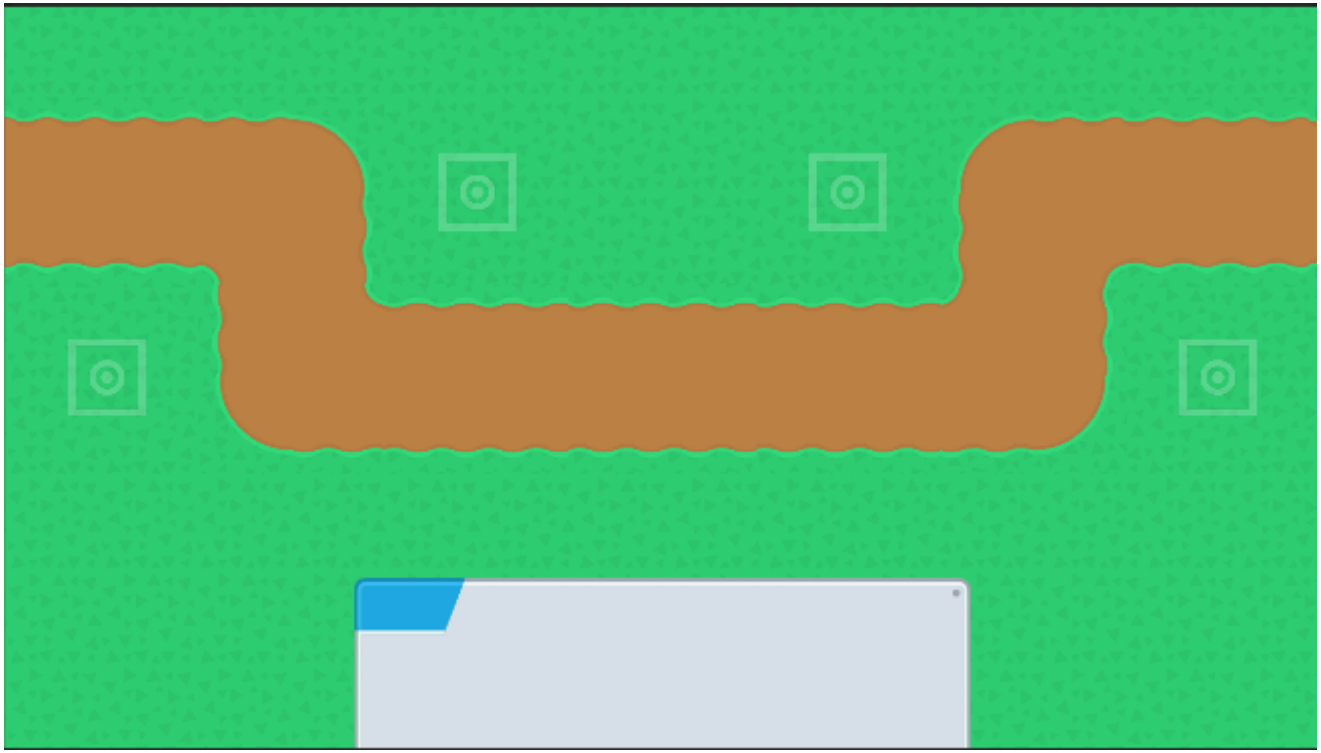
- Lalu, buat Image UI satu lagi sebagai child dari Plate, dan beri nama Icon, lalu atur Rect Transform-nya seperti ini:



Setelah selesai, pindah ke window Project, buatlah folder Prefabs, lalu drag game object Tower UI ke sana.

- Kamu sudah berhasil membuat sebuah prefab, lalu hapus saja Tower UI yang sekarang ada di dalam Hierarchy (child dari Panel).

5. Tampilan setelah menyelesaikan semua tahap 2



4. Create Tower and Level Manager

Create Tower and Level Manager

1. Membuat script sebagai komponen

- Buatlah folder Scripts di dalam Assets untuk mengelompokkan setiap script.
- Lalu klik kanan pada folder tersebut, pilih Create à Script.
- Ubah namanya menjadi Tower dan masukkan kode berikut.

```
using UnityEngine;


public class Tower : MonoBehaviour
{
    // Tower Component
    [SerializeField] private SpriteRenderer _towerPlace;
    [SerializeField] private SpriteRenderer _towerHead;


    // Tower Properties
    [SerializeField] private int _shootPower = 1;
    [SerializeField] private float _shootDistance = 1f;
    [SerializeField] private float _shootDelay = 5f;
    [SerializeField] private float _bulletSpeed = 1f;
    [SerializeField] private float _bulletSplashRadius = 0f;


    // Fungsi yang digunakan untuk mengambil sprite pada Tower Head
    public Sprite GetTowerHeadIcon ()
    {
        return _towerHead.sprite;
    }
}
```

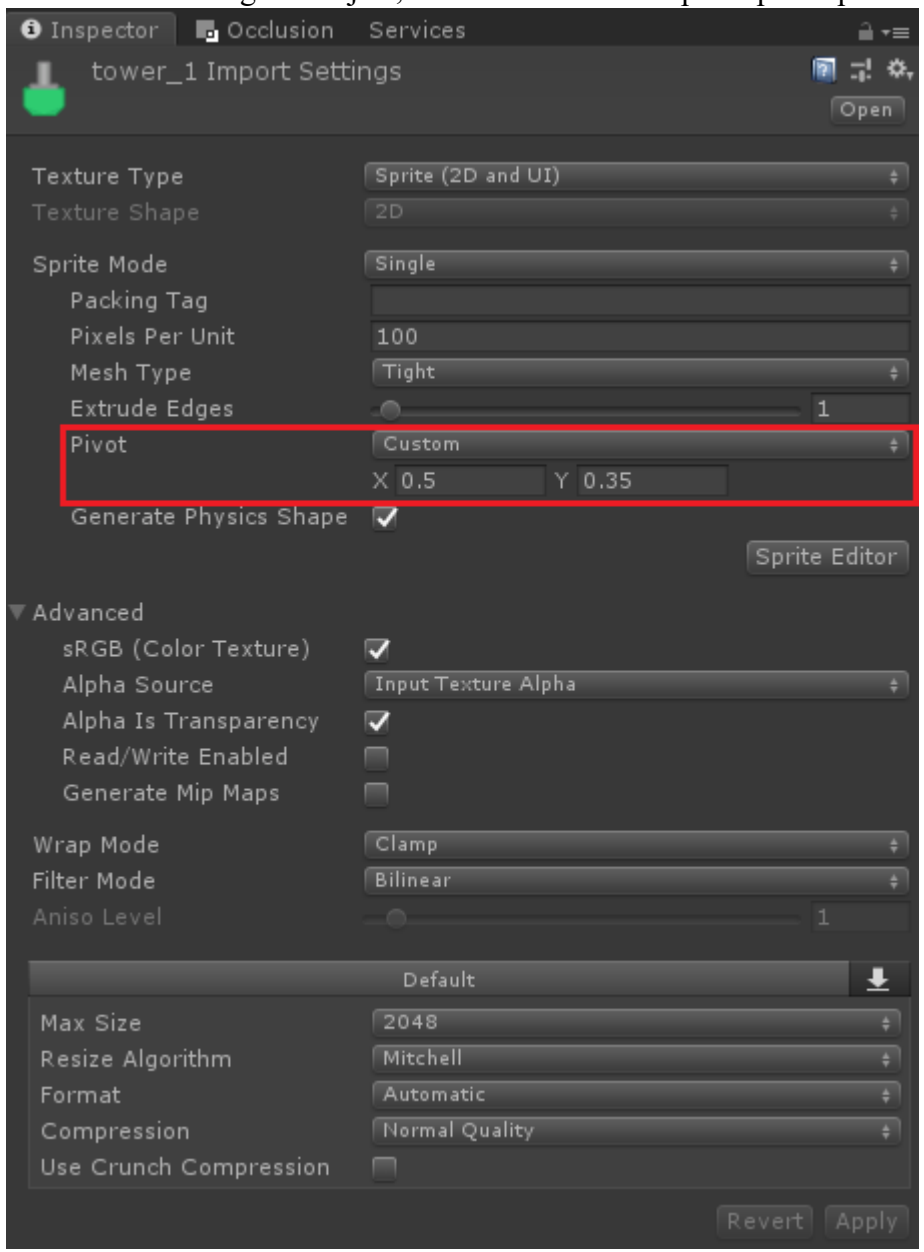
Fungsi SerializeField pada kode di atas adalah menampilkan variable selain public ke Inspector tanpa bisa diakses oleh class lain.

2. Membuat Base Prefab

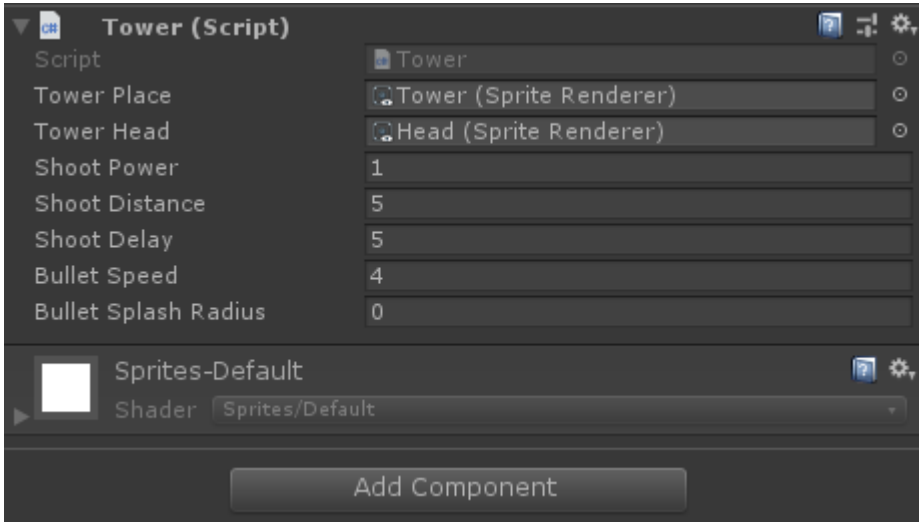
- Buat game object dari sprite tower_plate dan beri nama Tower.
- Lalu buat game object dari sprite tower_1 sebagai child dari Tower dan beri nama Head dan ubah posisi z-nya menjadi -1 agar lebih di atas dari

Tower.

- Karena posisi Head terlihat aneh, sekarang ubah Pivot dari setiap sprite tower_1, tower_2, tower_3 menjadi seperti ini (fungsi pivot adalah pusat rotasi dari sebuah game object, maka dari itu karena pusat pada sprite ini tidak ditengah, kita harus mengubahnya secara manual):



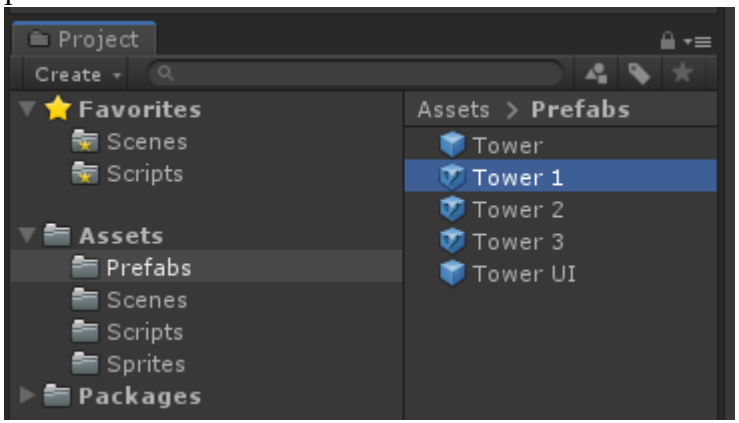
- Kemudian, drag script Tower ke dalam komponen game object Tower. Masukkan game object Head ke dalam Tower Head dan game object Tower sendiri ke dalam Tower Place.



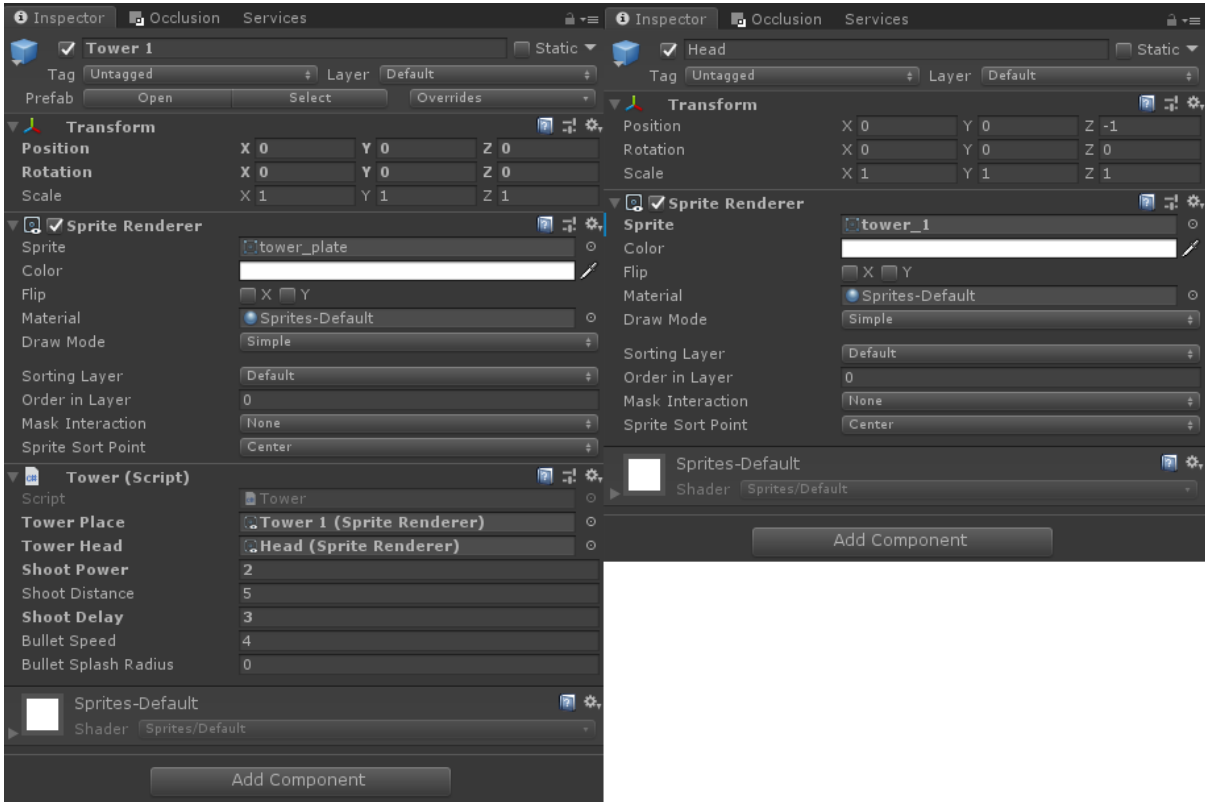
- Setelah selesai, drag game object Tower ke dalam Assets/Prefabs agar menjadi sebuah prefab dan hapus game object Tower yang ada di Hierarchy.

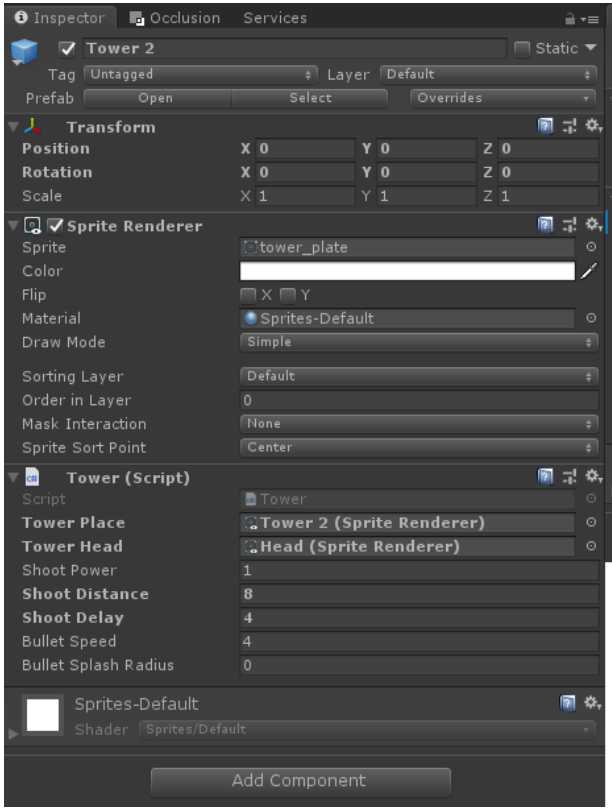
3. Membuat Prefab Variant

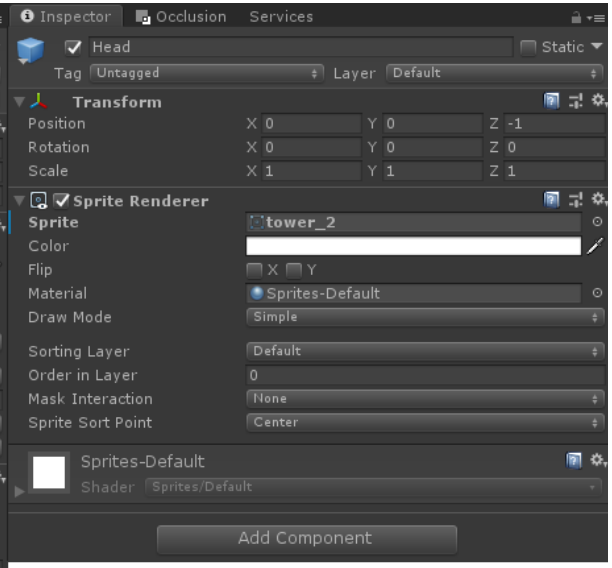
- Pilihlah prefab Tower yang telah dibuat, lalu klik kanan dan Pilih Create à Prefab Variant (fungsi ini ialah membuat sebuah turunan prefab). Buatlah hingga menjadi 3 dan namai sesuai urutan saja agar mudah dikenali. Prefab Variant ini juga memudahkan untuk mengubah value sebuah prefab secara keseluruhan.

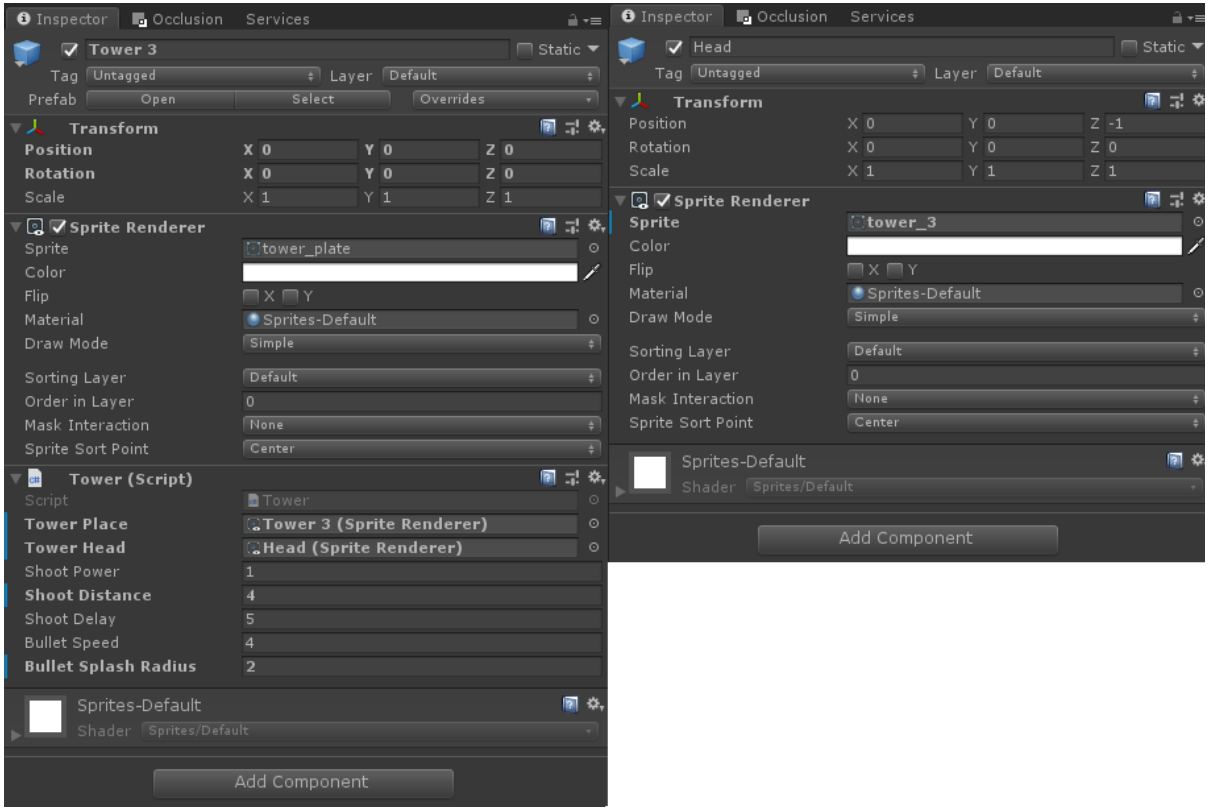


- Buatlah setiap prefab tersebut menjadi seperti ini:


Inspector
Occlusion
Services
Tower 1
Static
Tag Untagged
Layer Default
Prefab Open Select Overrides
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Sprite Renderer
Sprite tower_plate
Color
Flip X Y
Material Sprites-Default
Draw Mode Simple
Sorting Layer Default
Order in Layer 0
Mask Interaction None
Sprite Sort Point Center
Tower (Script)
Script Tower
Tower Place Tower 1 (Sprite Renderer)
Tower Head Head (Sprite Renderer)
Shoot Power 2
Shoot Distance 5
Shoot Delay 3
Bullet Speed 4
Bullet Splash Radius 0
Sprites-Default
Shader Sprites/Default
Add Component
Inspector
Occlusion
Services
Head
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z -1
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Sprite Renderer
Sprite tower_1
Color
Flip X Y
Material Sprites-Default
Draw Mode Simple
Sorting Layer Default
Order in Layer 0
Mask Interaction None
Sprite Sort Point Center
Sprites-Default
Shader Sprites/Default
Add Component


Inspector
Occlusion
Services
Tower 2
Static
Tag
Untagged
Layer
Default
Prefab
Open
Select
Overrides
Transform
Position
X 0
Y 0
Z 0
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Sprite Renderer
Sprite
tower_plate
Color
Flip
X
Y
Material
Sprites-Default
Draw Mode
Simple
Sorting Layer
Default
Order in Layer
0
Mask Interaction
None
Sprite Sort Point
Center
Tower (Script)
Script
Tower
Tower Place
Tower 2 (Sprite Renderer)
Tower Head
Head (Sprite Renderer)
Shoot Power
1
Shoot Distance
8
Shoot Delay
4
Bullet Speed
4
Bullet Splash Radius
0
Sprites-Default
Shader
Sprites/Default
Add Component


Inspector
Occlusion
Services
Head
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 0
Z -1
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Sprite Renderer
Sprite
tower_2
Color
Flip
X
Y
Material
Sprites-Default
Draw Mode
Simple
Sorting Layer
Default
Order in Layer
0
Mask Interaction
None
Sprite Sort Point
Center
Sprites-Default
Shader
Sprites/Default
Add Component



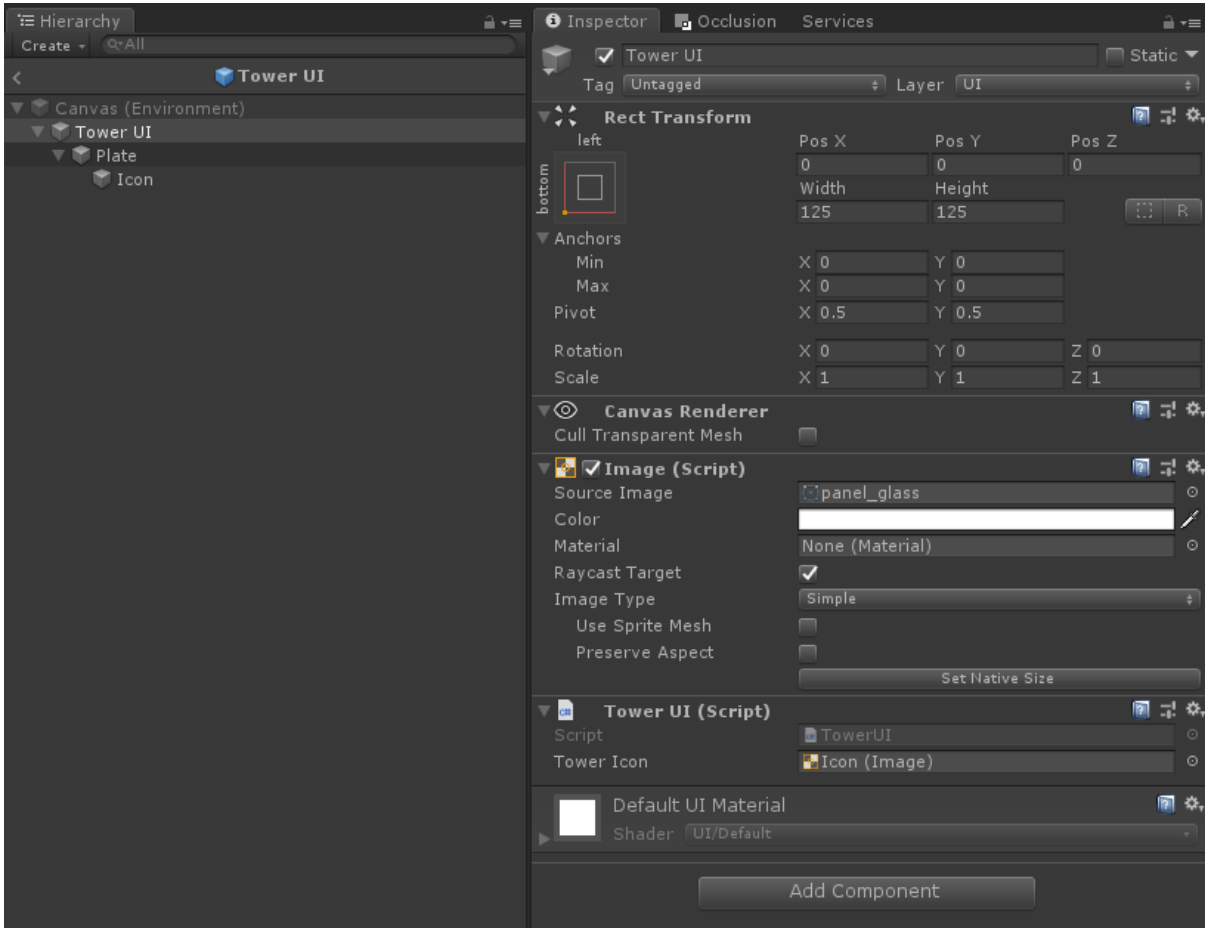
Untuk melihat value khusus pada sebuah prefab variant ditandai dengan penulisan yang di bold, seperti pada variable Tower Place, Tower Head, dsb. Value tersebut tidak akan tertimpa ketika kamu mengubah base prefab-nya.

- Buatlah script TowerUI untuk mengatur prefab Tower UI dan tambahkan ke dalam komponen prefab Tower UI. Script ini akan memiliki fungsi untuk mengubah tower data pada prefab Tower UI.

```
using UnityEngine;
using UnityEngine.UI;


public class TowerUI : MonoBehaviour
{
    [SerializeField] private Image _towerIcon;


    private Tower _towerPrefab;


    public void SetTowerPrefab (Tower tower)
    {
        _towerPrefab = tower;
        _towerIcon.sprite = tower.GetTowerHeadIcon ();
    }
}
```

Jika tidak menambahkan using UnityEngine.UI, kamu tidak akan bisa mengakses library dari komponen UI.

- Masukkan game object Icon ke dalam variable Tower Icon pada Inspector Tower UI.



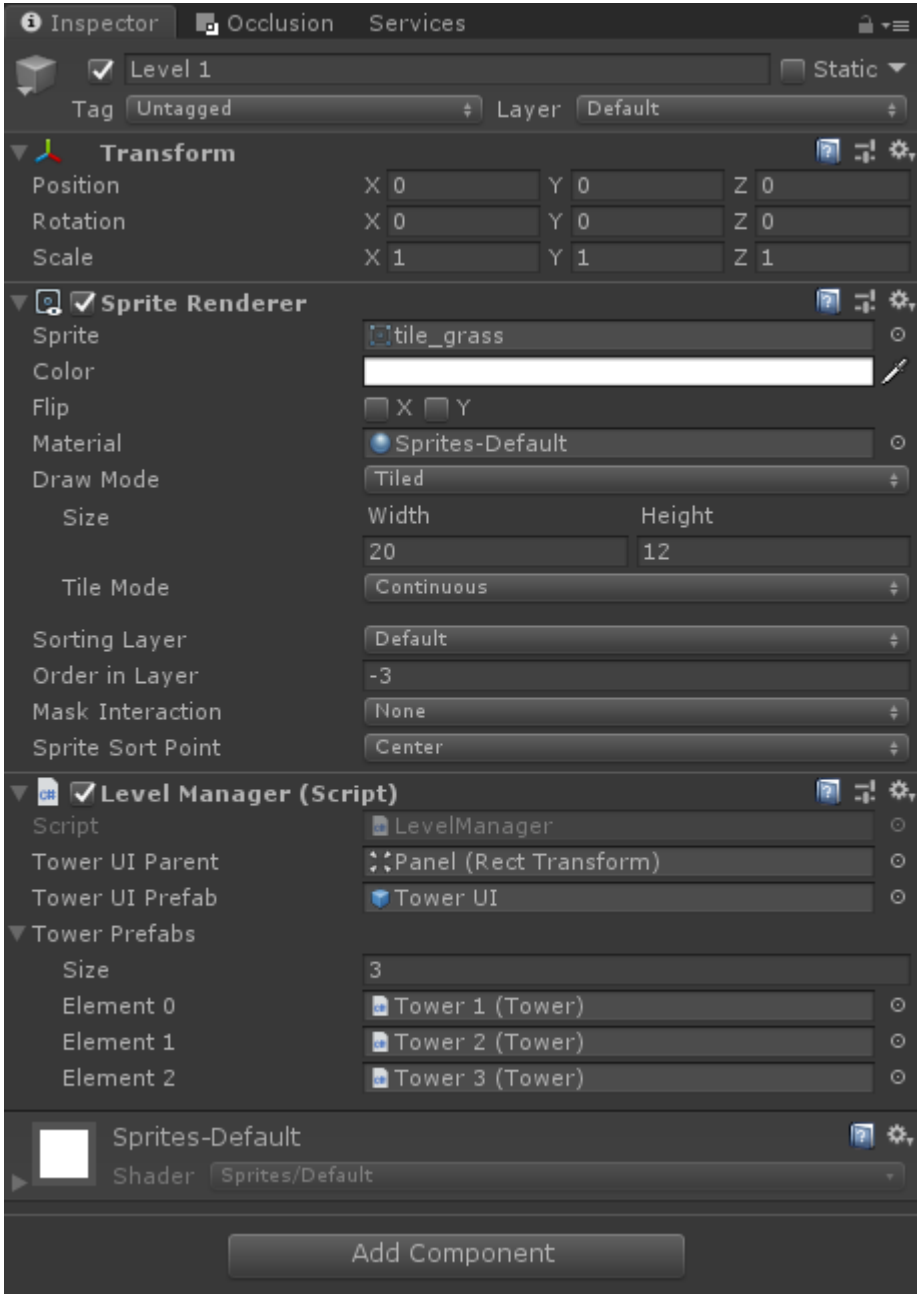
4. Membuat Singleton

- Buatlah script LevelManager untuk memunculkan setiap Tower UI ke dalam Panel Tower Selection berdasarkan prefab yang didaftarkan. Script ini juga merupakan Singleton, yaitu object script ini hanya akan ada satu selama game dimainkan dan dapat diakses melalui variable Instance-nya.

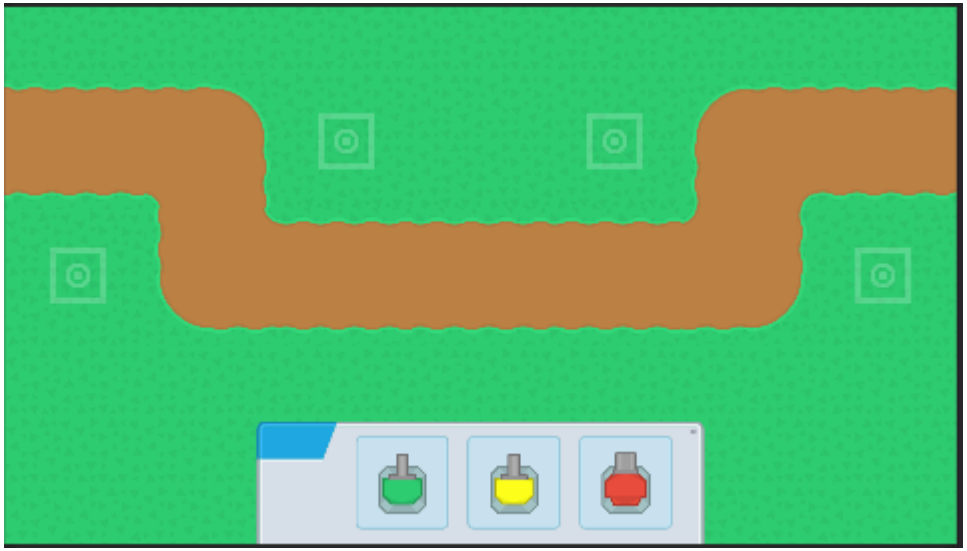
```
using UnityEngine;


public class LevelManager : MonoBehaviour
{
   // Fungsi Singleton
   private static LevelManager _instance = null;
   public static LevelManager Instance
   {
      get
      {
         if (_instance == null)
         {
            _instance = FindObjectOfType<LevelManager> ();
         }


         return _instance;
      }
   }


   [SerializeField] private Transform _towerUIParent;
   [SerializeField] private GameObject _towerUIPrefab;


   [SerializeField] private Tower[] _towerPrefabs;


   private void Start ()
   {
```

```
        InstantiateAllTowerUI ();
    }


    // Menampilkan seluruh Tower yang tersedia pada UI Tower Selection
    private void InstantiateAllTowerUI ()
    {
        foreach (Tower tower in _towerPrefabs)
        {
            GameObject newTowerUIObj = Instantiate
(_towerUIPrefab.gameObject, _towerUIParent);
            TowerUI newTowerUI =
newTowerUIObj.GetComponent<TowerUI> ();


            newTowerUI.SetTowerPrefab (tower);
            newTowerUI.transform.name = tower.name;
        }
    }
}
```

- Masukkan script ini ke dalam Level 1 dan masukkan Canvas/Tower Selection/Panel ke dalam variable Tower UI Parent.
- Masukkan prefab Tower UI ke dalam Tower UI Prefab.
- Terakhir, masukkan setiap tower prefab ke dalam array Tower Prefabs.



5. Tampilan setelah menyelesaikan semua tahap 3 ketika dimainkan



5. Drag and Drop Tower

Drag and Drop Tower

1. Collider with Trigger Event

- Pada setiap game object tower_placement yang telah dibuat, tambahkan komponen Box Collider 2D dan centang bagian Is Trigger-nya.
- Buatlah script baru bernama TowerPlacement dengan kode berikut.

```
using UnityEngine;


public class TowerPlacement : MonoBehaviour
{
   private Tower _placedTower;


   // Fungsi yang terpanggil sekali ketika ada object Rigidbody yang menyentuh area collider
   private void OnTriggerEnter2D (Collider2D collision)
   {
      if (_placedTower != null)
      {
         return;
      }


      Tower tower = collision.GetComponent<Tower> ();
      if (tower != null)
      {
         tower.SetPlacePosition (transform.position);
         _placedTower = tower;
      }
   }


   // Kebalikan dari OnTriggerEnter2D, fungsi ini terpanggil sekali ketika object tersebut meninggalkan area collider
   private void OnTriggerExit2D (Collider2D collision)
```

```
    {
        if (_placedTower == null)
        {
            return;
        }

        _placedTower.SetPlacePosition (null);
        _placedTower = null;
    }
}
```

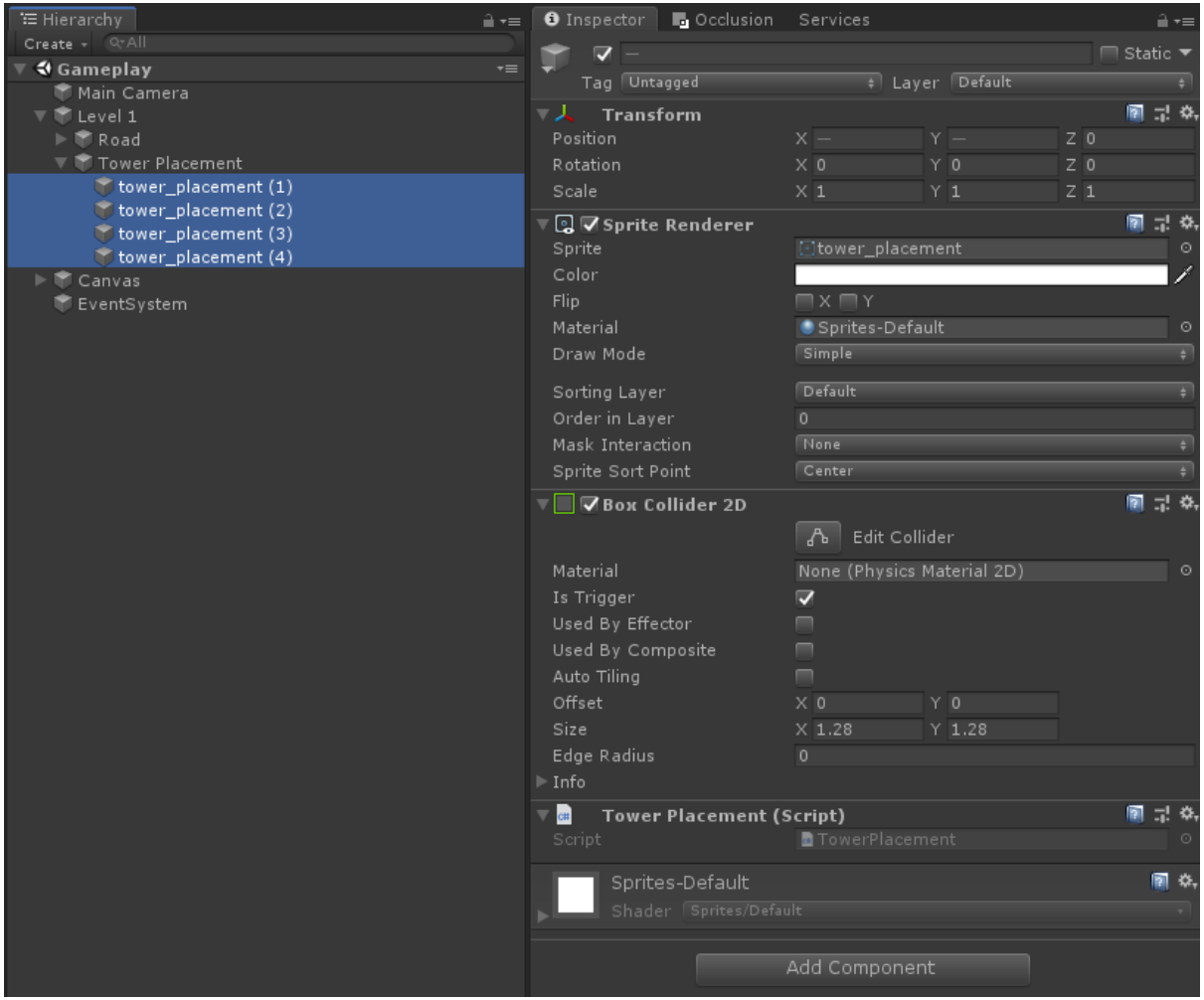
Lalu, sebelum masuk ke tahap selanjutnya, masukkan code berikut pada class utama script Tower.cs:

```
public Vector2? PlacePosition { get; private set; }
```

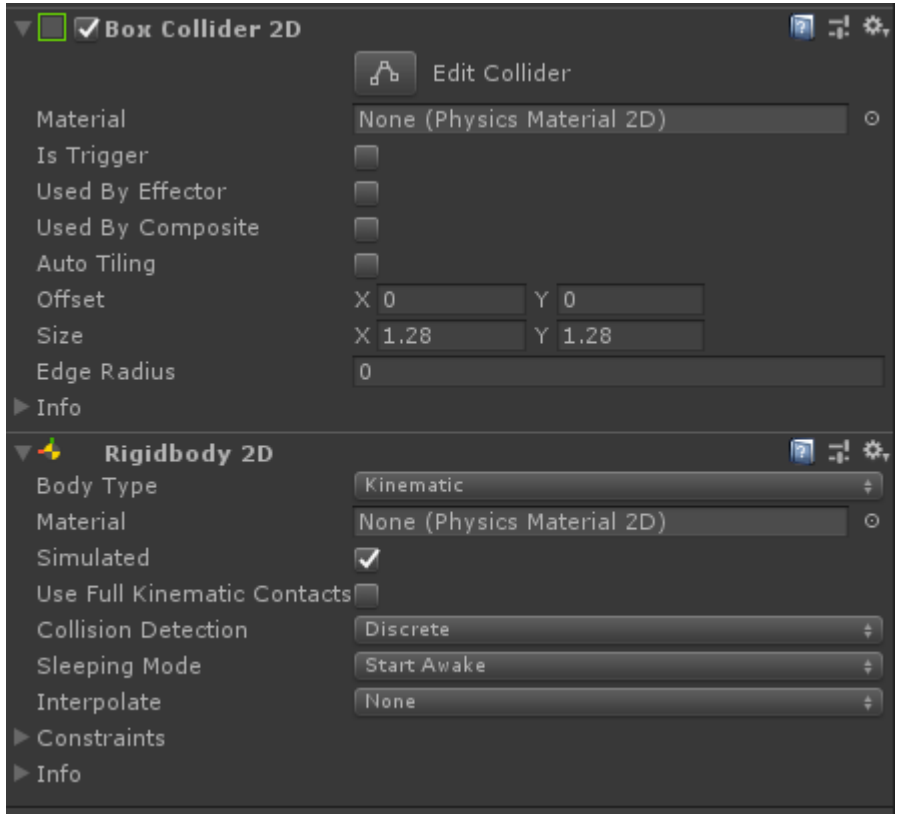
Lalu, buat fungsi berikut pada script Tower.cs:

```
public void SetPlacePosition(Vector2? newPosition)
{
    PlacePosition = newPosition;
}
```

- Tambahkan script TowerPlacement ke setiap game object tower_placement.



- Pada base prefab Tower juga tambahkan komponen Box Collider 2D dan Rigidbody 2D dan ubah Body Type-nya menjadi Kinematic. Fungsi Kinematic ialah maksudnya, game object tersebut hanya mendapat kalkulasi physic berdasarkan script yang diinputkan. Jika tidak diatur dari script, maka benda tidak akan otomatis terkalkulasi oleh physic dari Unity-nya, sebagai contoh: benda tidak akan dipengaruhi oleh gravitasi.



2. Drag Event Interface

- Tambahkan kode tambahan (yang berhuruf tebal) ke dalam script LevelManager.

```
// ...
    [SerializeField] private Tower[] _towerPrefabs;

    private List<Tower> _spawnedTowers = new List<Tower> ();

// ...

    // Menampilkan seluruh Tower yang tersedia pada UI Tower Selection
    private void InstantiateAllTowerUI ()
    {
        foreach (Tower tower in _towerPrefabs)
        {
            GameObject newTowerUIObj = Instantiate
(_towerUIPrefab.gameObject, _towerUIParent);
            TowerUI newTowerUI =
newTowerUIObj.GetComponent<TowerUI> ();

            newTowerUI.SetTowerPrefab (tower);
            newTowerUI.transform.name = tower.name;
        }
    }

    // Mendaftarkan Tower yang di-spawn agar bisa dikontrol oleh
LevelManager
    public void RegisterSpawnedTower (Tower tower)
    {
        _spawnedTowers.Add (tower);
```

```
    }
}
```

- Tambahkan kode tambahan ke dalam script Tower.

```
// ...
    // Tower Properties
    [SerializeField] private int _shootPower = 1;
    [SerializeField] private float _shootDistance = 1f;
    [SerializeField] private float _shootDelay = 5f;
    [SerializeField] private float _bulletSpeed = 1f;
    [SerializeField] private float _bulletSplashRadius = 0f;

    // Digunakan untuk menyimpan posisi yang akan ditempati selama tower di drag
    public Vector2? PlacePosition { get; private set; }

    public void SetPlacePosition (Vector2? newPosition)
    {
        PlacePosition = newPosition;
    }

    public void LockPlacement ()
    {
        transform.position = (Vector2) PlacePosition;
    }

    // Mengubah order in layer pada tower yang sedang di drag
    public void ToggleOrderInLayer (bool toFront)
    {
        int orderInLayer = toFront ? 2 : 0;
```

```
        _towerPlace.sortingOrder = orderInLayer;
        _towerHead.sortingOrder = orderInLayer;
    }

    // Fungsi yang digunakan untuk mengambil sprite pada Tower Head
    public Sprite GetTowerHeadIcon ()
    {
        return _towerHead.sprite;
    }
// ...
```

- Lalu, tambahkan kode berikut ke dalam script TowerUI, kode ini berisi interface yang dapat mentrigger event dari interaksi Drag.

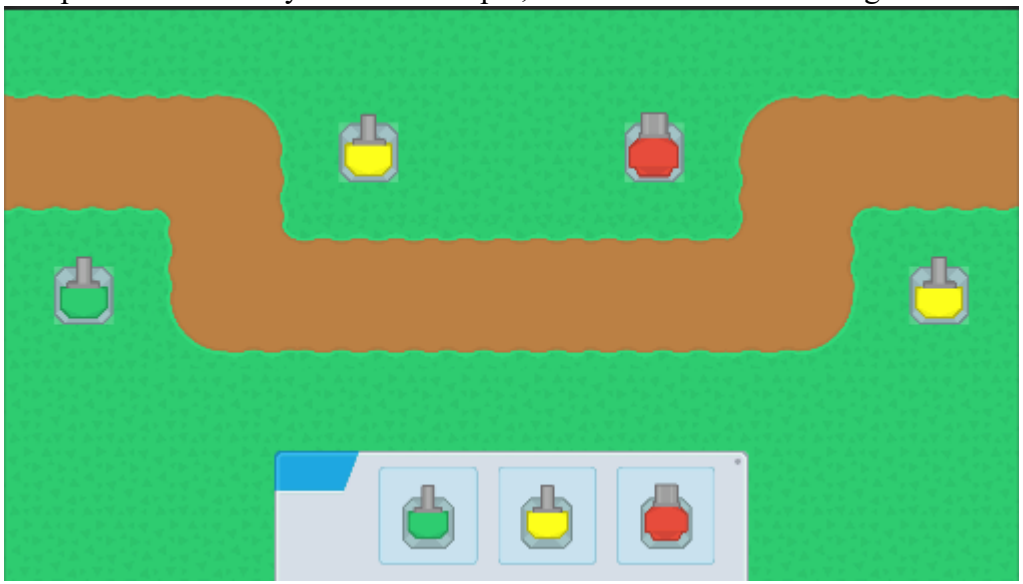
```
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;


public class TowerUI : MonoBehaviour, IBeginDragHandler,
IDragHandler, IEndDragHandler
{
   [SerializeField] private Image _towerIcon;


   private Tower _towerPrefab;
   private Tower _currentSpawnedTower;


   public void SetTowerPrefab (Tower tower)
   {
      _towerPrefab = tower;
      _towerIcon.sprite = tower.GetTowerHeadIcon ();
   }


   // Implementasi dari Interface IBeginDragHandler
   // Fungsi ini terpanggil sekali ketika pertama men-drag UI
   public void OnBeginDrag (PointerEventData eventData)
   {
      GameObject newTowerObj = Instantiate
(_towerPrefab.gameObject);
      _currentSpawnedTower = newTowerObj.GetComponent<Tower>
();
      _currentSpawnedTower.ToggleOrderInLayer (true);
```

```
    }

    // Implementasi dari Interface IDragHandler
    // Fungsi ini terpanggil selama men-drag UI
    public void OnDrag (PointerEventData eventData)
    {
        Camera mainCamera = Camera.main;
        Vector3 mousePosition = Input.mousePosition;
        mousePosition.z = -mainCamera.transform.position.z;
        Vector3 targetPosition = Camera.main.ScreenToWorldPoint
(mousePosition);

        _currentSpawnedTower.transform.position = targetPosition;
    }

    // Implementasi dari Interface IEndDragHandler
    // Fungsi ini terpanggil sekali ketika men-drop UI ini
    public void OnEndDrag (PointerEventData eventData)
    {
        if (_currentSpawnedTower.PlacePosition == null)
        {
            Destroy (_currentSpawnedTower.gameObject);
        }
        else
        {
            _currentSpawnedTower.LockPlacement ();
```

```
            _currentSpawnedTower.ToggleOrderInLayer (false);

            LevelManager.Instance.RegisterSpawnedTower
(_currentSpawnedTower);

            _currentSpawnedTower = null;

        }

    }

}
```

- Sebelum mencoba play, tambahkan terlebih dahulu prefab Tower ke dalam variable Tower Prefab pada Inspector TowerUI yang baru saja ditambahkan. Inspector TowerUI ada pada game object Level 1.

3. Tampilan setelah menyelesaikan tahap 4, kamu sudah bisa men-drag Tower baru dari Tower Selection



6. Create Enemy Variant

Create Enemy Variant

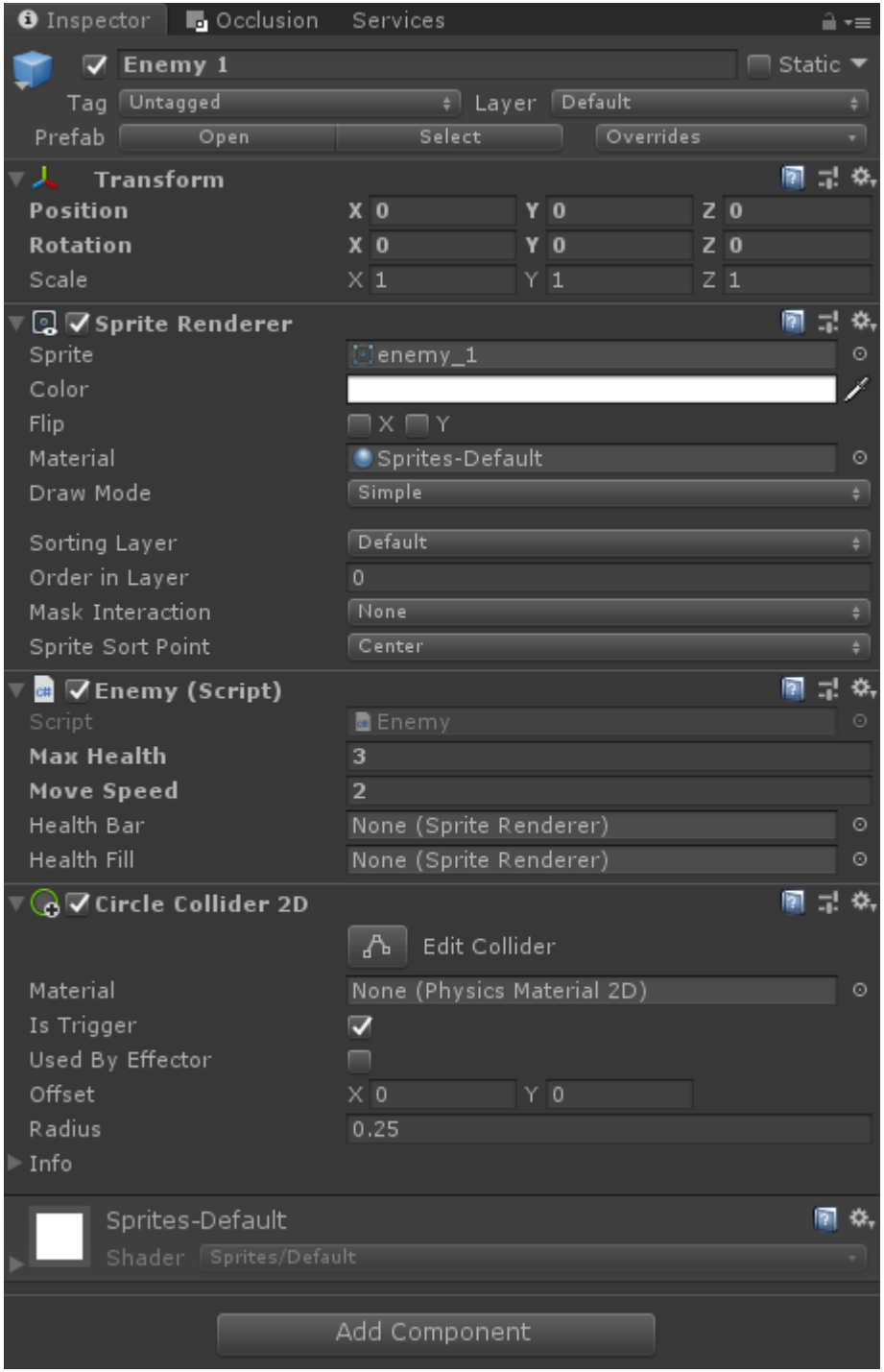
1. Membuat Enemy

- Buat script Enemy dengan kode berikut.

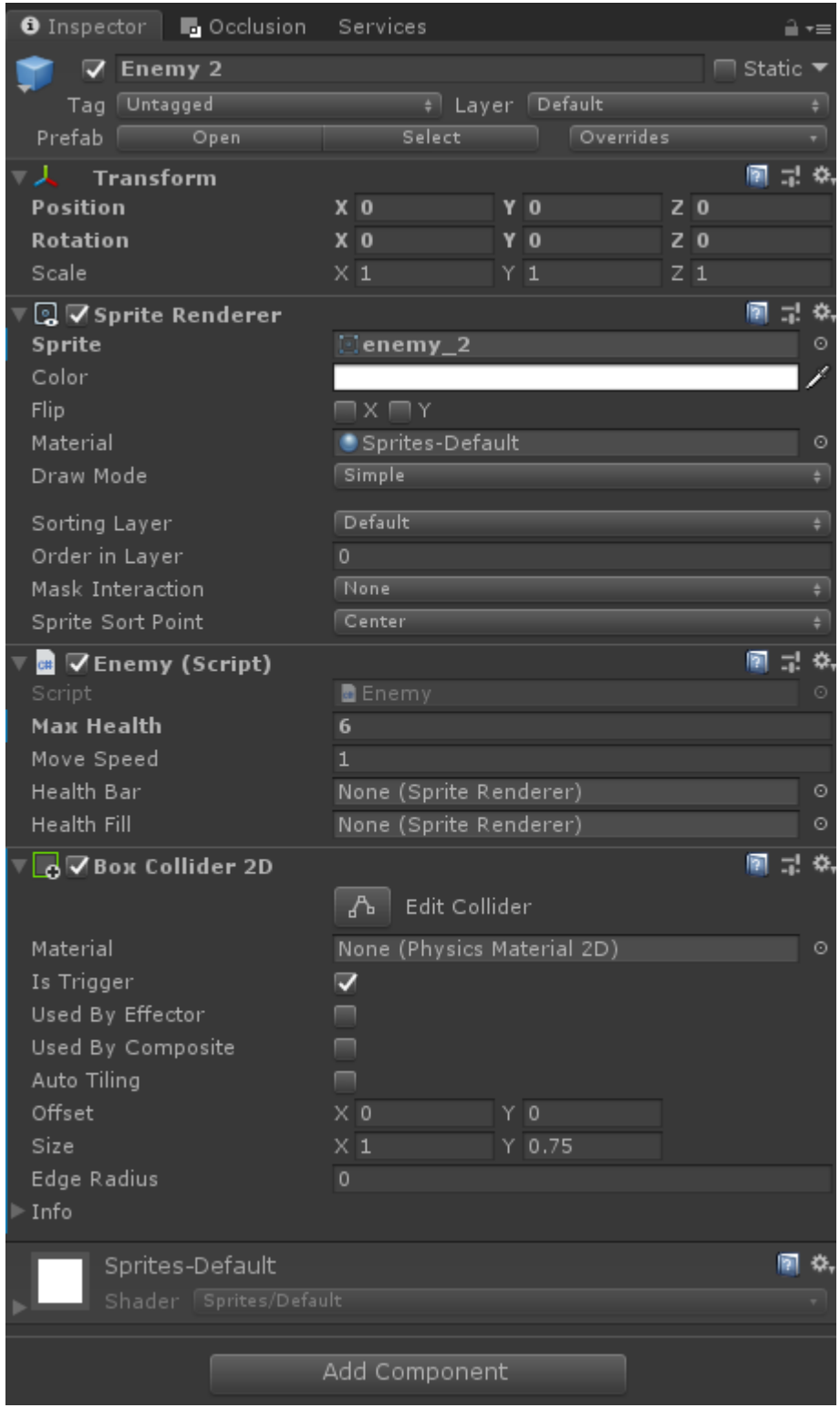
```
using UnityEngine;


public class Enemy : MonoBehaviour
{
    [SerializeField] private int _maxHealth = 1;
    [SerializeField] private float _moveSpeed = 1f;
    [SerializeField] private SpriteRenderer _healthBar;
    [SerializeField] private SpriteRenderer _healthFill;


    private int _currentHealth;


    // Fungsi ini terpanggil sekali setiap kali menghidupkan game object yang memiliki script ini
    private void OnEnable ()
    {
        _currentHealth = _maxHealth;
        _healthFill.size = _healthBar.size;
    }
}
```

- Buat game object baru dari sprite enemy_1 dan ubah namanya menjadi Enemy.
- Lalu masukkan script Enemy ke dalam game object tersebut.

- Drag game object Enemy ke dalam Assets/Prefabs untuk membuat base prefab Enemy dan hapus yang ada di Hierarchy.
- Buatlah 2 prefab variant dari prefab Enemy tersebut.
- Untuk variant pertama, tambahkan Circle Collider 2D dan ubah radiusnya menjadi 0.25 agar lebih sesuai dengan ukuran sprite-nya. Jangan lupa juga mencentang Is Trigger.

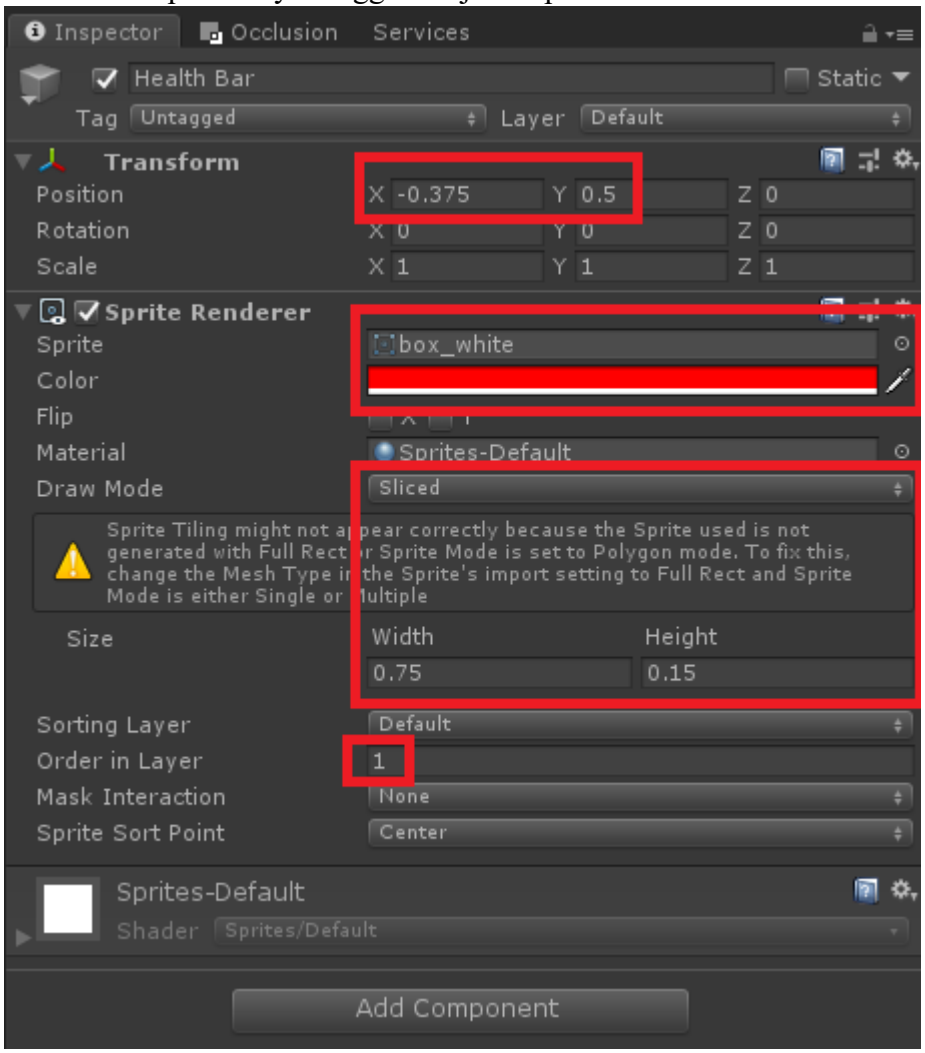


- Sedangkan pada variant kedua, ubah sprite-nya menjadi enemy_2.
- Tambahkan Box Collider 2D dan ubah size-nya menjadi (1, 0.75), jangan lupa mencentang Is Trigger-nya.

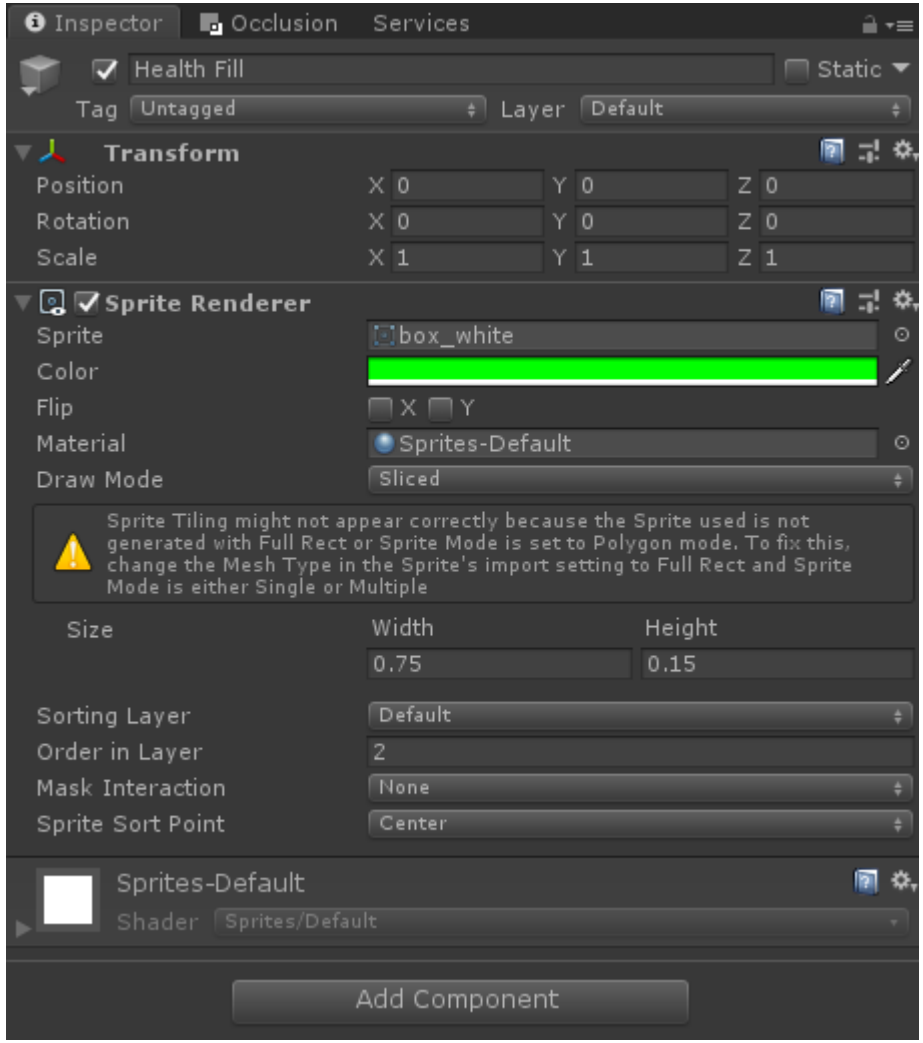

Inspector
Occlusion
Services
Enemy 2
Static
Tag
Untagged
Layer
Default
Prefab
Open
Select
Overrides
Transform
Position
Rotation
Scale
Sprite Renderer
Sprite
enemy_2
Color
Flip
Material
Sprites-Default
Draw Mode
Simple
Sorting Layer
Default
Order in Layer
Mask Interaction
None
Sprite Sort Point
Center
Enemy (Script)
Script
Enemy
Max Health
Move Speed
Health Bar
None (Sprite Renderer)
Health Fill
Box Collider 2D
Edit Collider
Material
None (Physics Material 2D)
Is Trigger
Used By Effector
Used By Composite
Auto Tiling
Offset
Size
Edge Radius
Info
Sprites-Default
Shader
Sprites/Default
Add Component

2. Menambahkan Health Bar

- Pilih sprite box_white di window Project, lalu di window Inspector, ubah pivot-nya menjadi Left.
- Buka Prefab Enemy, drag sprite panel_white ke arah game object Enemy untuk membuat game object baru sebagai child Enemy, ubah namanya menjadi Health Bar.
- Lalu atur Inspector-nya hingga menjadi seperti ini:



- Duplikat game object Health Bar, lalu ubah namanya menjadi Health Fill dan masukkan ke dalam child Health Bar.
- Masukkan Health Bar dan Health Fill pada component Enemy di game object Enemy.
- Lalu ubah Order in Layer-nya menjadi 2, ubah warna-nya menjadi hijau sebagai penanda health terisi, dan atur posisinya pada titik 0.


Inspector
Occlusion
Services
Health Fill
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 0
Z 0
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Sprite Renderer
Sprite
box_white
Color
Flip
X
Y
Material
Sprites-Default
Draw Mode
Sliced
Sprite Tiling might not appear correctly because the Sprite used is not generated with Full Rect or Sprite Mode is set to Polygon mode. To fix this, change the Mesh Type in the Sprite's import setting to Full Rect and Sprite Mode is either Single or Multiple
Size
Width
Height
0.75
0.15
Sorting Layer
Default
Order in Layer
2
Mask Interaction
None
Sprite Sort Point
Center
Sprites-Default
Shader
Sprites/Default
Add Component

7. Enemy Following Path

Enemy Following Path

1. Menambahkan Fungsi Enemy untuk menuju sebuah titik.

- Tambahkan kode berikut pada script Enemy.

```
// ...
    private int _currentHealth;


    public Vector3 TargetPosition { get; private set; }
    public int CurrentPathIndex { get; private set; }


    // Fungsi ini terpanggil sekali setiap kali menghidupkan game object yang memiliki script ini
    private void OnEnable ()
    {
        _currentHealth = _maxHealth;
        _healthFill.size = _healthBar.size;
    }


    public void MoveToTarget ()
    {
        transform.position = Vector3.MoveTowards (transform.position, TargetPosition, _moveSpeed * Time.deltaTime);
    }


    public void SetTargetPosition (Vector3 targetPosition)
    {
        TargetPosition = targetPosition;
        _healthBar.transform.parent = null;


        // Mengubah rotasi dari enemy
```

```
    Vector3 distance = TargetPosition - transform.position;
    if (Mathf.Abs (distance.y) > Mathf.Abs (distance.x))
    {
        // Menghadap atas
        if (distance.y > 0)
        {
            transform.rotation = Quaternion.Euler (new Vector3 (0f, 0f,
90f));
        }
        // Menghadap bawah
        else
        {
            transform.rotation = Quaternion.Euler (new Vector3 (0f, 0f,
-90f));
        }
    }
    else
    {
        // Menghadap kanan (default)
        if (distance.x > 0)
        {
            transform.rotation = Quaternion.Euler (new Vector3 (0f, 0f,
0f));
        }
        // Menghadap kiri
        else
        {
```

```
            transform.rotation = Quaternion.Euler (new Vector3 (0f, 0f,
180f));
        }
      }

      _healthBar.transform.parent = transform;
   }

   // Menandai indeks terakhir pada path
   public void SetCurrentPathIndex (int currentIndex)
   {
      CurrentPathIndex = currentIndex;
   }
}
```

2. Membuat Spawner

- Tambahkan kode berikut ke dalam script LevelManager untuk memunculkan musuh setiap jeda waktu yang ditentukan.

```
// ...
    [SerializeField] private Tower[] _towerPrefabs;
    [SerializeField] private Enemy[] _enemyPrefabs;

    [SerializeField] private Transform[] _enemyPaths;
    [SerializeField] private float _spawnDelay = 5f;

    private List<Tower> _spawnedTowers = new List<Tower> ();
    private List<Enemy> _spawnedEnemies = new List<Enemy> ();

    private float _runningSpawnDelay;

    private void Start ()
    {
        InstantiateAllTowerUI ();
    }

    private void Update ()
    {
        // Counter untuk spawn enemy dalam jeda waktu yang ditentukan
        // Time.unscaledDeltaTime adalah deltaTime yang independent, tidak terpengaruh oleh apapun kecuali game object itu sendiri,
        // jadi bisa digunakan sebagai penghitung waktu
        _runningSpawnDelay -= Time.unscaledDeltaTime;
        if (_runningSpawnDelay <= 0f)
```

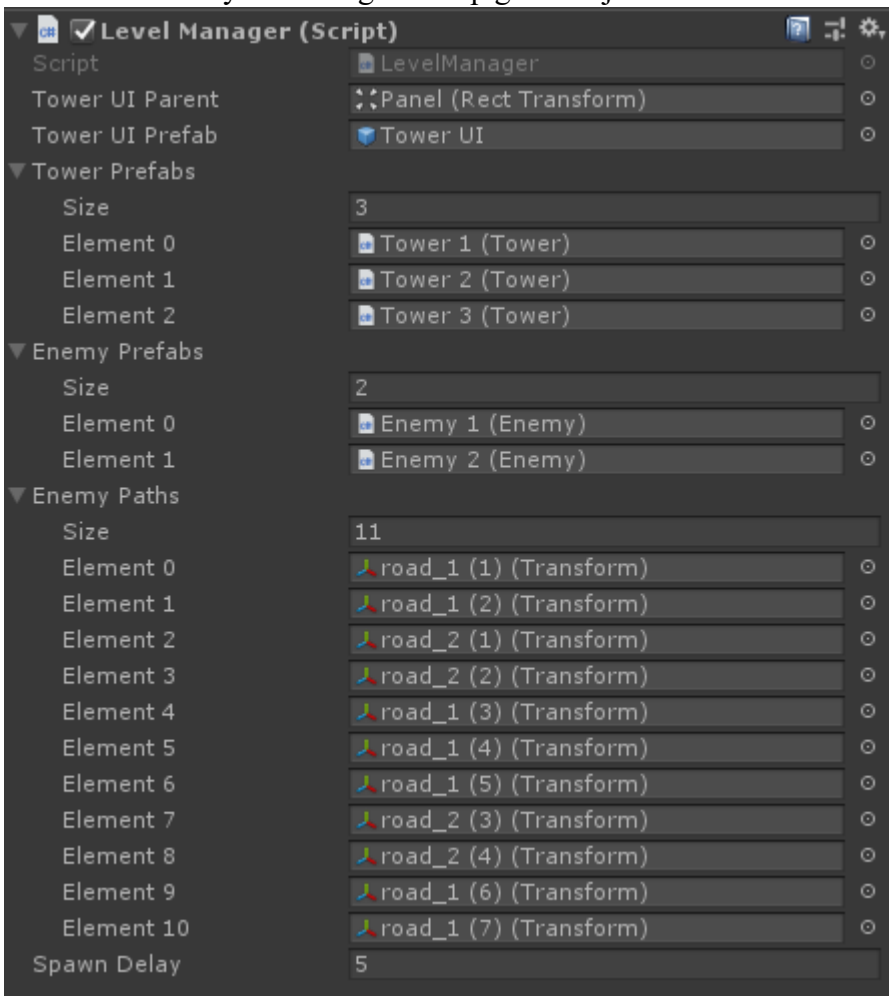
```
        {
            SpawnEnemy ();
            _runningSpawnDelay = _spawnDelay;
        }


        foreach (Enemy enemy in _spawnedEnemies)
        {
            if (!enemy.gameObject.activeSelf)
            {
                continue;
            }


            // Kenapa nilainya 0.1? Karena untuk lebih mentoleransi perbedaan posisi,
            // akan terlalu sulit jika perbedaan posisinya harus 0 atau sama persis
            if (Vector2.Distance (enemy.transform.position, enemy.TargetPosition) < 0.1f)
            {
                enemy.SetCurrentPathIndex (enemy.CurrentPathIndex + 1);
                if (enemy.CurrentPathIndex < _enemyPaths.Length)
                {
                    enemy.SetTargetPosition (_enemyPaths[enemy.CurrentPathIndex].position);
                }
                else
                {
```

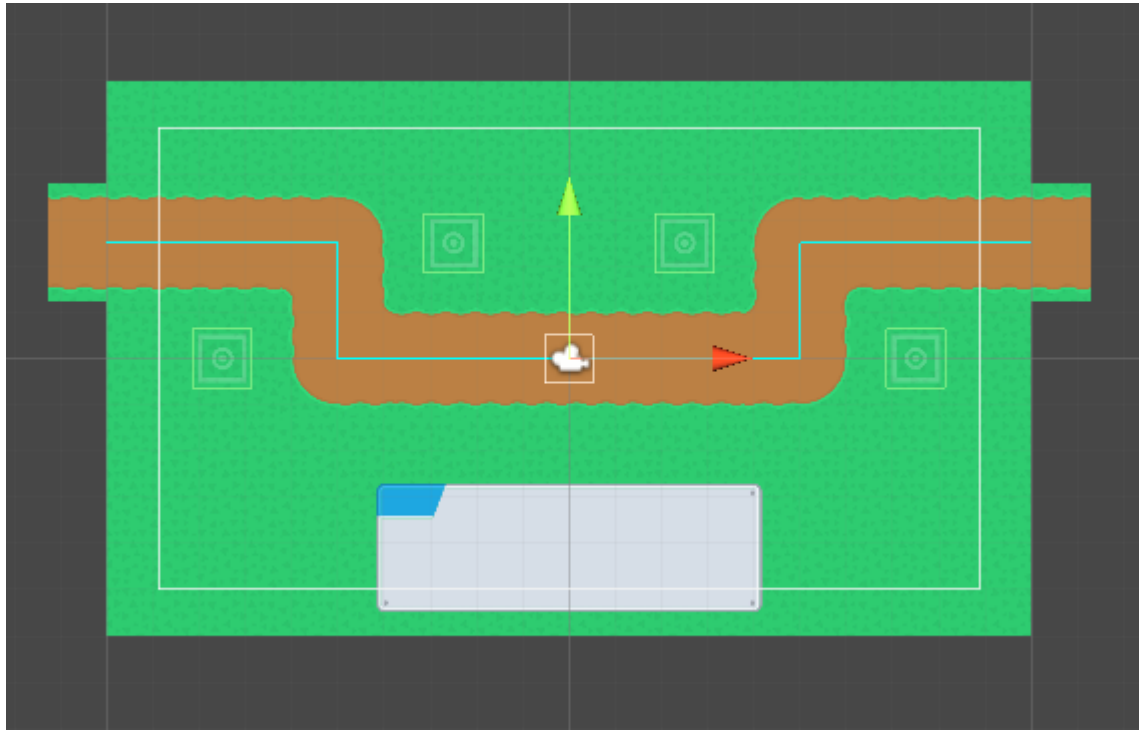
```
                enemy.gameObject.SetActive (false);
            }
        }
        else
        {
            enemy.MoveToTarget ();
        }
    }
}

// ...

public void RegisterSpawnedTower (Tower tower)
{
    _spawnedTowers.Add (tower);
}

private void SpawnEnemy ()
{
    int randomIndex = Random.Range (0, _enemyPrefabs.Length);
    string enemyIndexString = (randomIndex + 1).ToString ();

    GameObject newEnemyObj = _spawnedEnemies.Find (
        e => !e.gameObject.activeSelf && e.name.Contains
(enemyIndexString)
    )?.gameObject;
```

```
    if (newEnemyObj == null)
    {
        newEnemyObj = Instantiate
(_enemyPrefabs[randomIndex].gameObject);
    }

    Enemy newEnemy = newEnemyObj.GetComponent<Enemy> ();
    if (!_spawnedEnemies.Contains (newEnemy))
    {
        _spawnedEnemies.Add (newEnemy);
    }

    newEnemy.transform.position = _enemyPaths[0].position;
    newEnemy.SetTargetPosition (_enemyPaths[1].position);
    newEnemy.SetCurrentPathIndex (1);
    newEnemy.gameObject.SetActive (true);
  }

  // Untuk menampilkan garis penghubung dalam window Scene
  // tanpa harus di-Play terlebih dahulu
  private void OnDrawGizmos ()
  {
    for (int i = 0; i < _enemyPaths.Length - 1; i++)
    {
        Gizmos.color = Color.cyan;
```

```
        Gizmos.DrawLine (_enemyPaths[i].position, _enemyPaths[i + 1].position);
    }
  }
}
```

- Jangan lupa untuk mengisi variable Enemy Prefabs pada game object Level 1 dengan prefab variant yang telah dibuat.
- Lalu isilah Enemy Path dengan setiap game object road secara berurutan.





- Setelah di play seharusnya musuh bisa dimunculkan dalam setiap jeda 5 detik.

8. Tower Attacking Enemy

Tower Attacking Enemy

- Sebelum menyuruh tower menyerang musuh, kita akan menambahkan fungsi berikut agar Enemy bisa hancur ketika health-nya habis. Masukkan code berikut ke dalam script Enemy:

```
// ...
        // Menghadap kiri
        else
        {
            transform.rotation = Quaternion.Euler (new Vector3 (0f, 0f,
180f));
        }
    }
  }

  public void ReduceEnemyHealth (int damage)
  {
    _currentHealth -= damage;
    if (_currentHealth <= 0)
    {
      gameObject.SetActive (false);
    }
  }
//...
```

- Buatlah script Bullet dengan kode berikut.

```
using UnityEngine;


public class Bullet : MonoBehaviour
{
    private int _bulletPower;
    private float _bulletSpeed;
    private float _bulletSplashRadius;


    private Enemy _targetEnemy;


    // FixedUpdate adalah update yang lebih konsisten jeda pemanggilannya
    // cocok digunakan jika karakter memiliki Physic (Rigidbody, dll)
    private void FixedUpdate ()
    {
        if (_targetEnemy != null)
        {
            if (!_targetEnemy.gameObject.activeSelf)
            {
                gameObject.SetActive (false);
                _targetEnemy = null;
                return;
            }


            transform.position = Vector3.MoveTowards (transform.position,
_targetEnemy.transform.position, _bulletSpeed * Time.fixedDeltaTime);
```

```
        Vector3 direction = _targetEnemy.transform.position -
transform.position;

        float targetAngle = Mathf.Atan2 (direction.y, direction.x) *
Mathf.Rad2Deg;

        transform.rotation = Quaternion.Euler (new Vector3 (0f, 0f,
targetAngle - 90f));

      }

   }


   private void OnTriggerEnter2D (Collider2D collision)

   {

      if (_targetEnemy == null)

      {

         return;

      }


      if (collision.gameObject.Equals (_targetEnemy.gameObject))

      {

         gameObject.SetActive (false);


         // Bullet yang memiliki efek splash area

         if (_bulletSplashRadius > 0f)

         {

            LevelManager.Instance.ExplodeAt (transform.position,
_bulletSplashRadius, _bulletPower);

         }

         // Bullet yang hanya single-target
```

```
            else
            {
                _targetEnemy.ReduceEnemyHealth (_bulletPower);
            }

            _targetEnemy = null;
        }
    }

    public void SetProperties (int bulletPower, float bulletSpeed, float
bulletSplashRadius)
    {
        _bulletPower = bulletPower;
        _bulletSpeed = bulletSpeed;
        _bulletSplashRadius = bulletSplashRadius;
    }

    public void SetTargetEnemy (Enemy enemy)
    {
        _targetEnemy = enemy;
    }
}
```

- Script Bullet akan error karena tidak adanya fungsi ExplodeAt pada LevelManager, maka dari itu akan kita tambahkan beberapa kode pada

LevelManager juga:

```
// ...
    private List<Tower> _spawnedTowers = new List<Tower> ();
    private List<Enemy> _spawnedEnemies = new List<Enemy> ();
    private List<Bullet> _spawnedBullets = new List<Bullet> ();

// ...
        newEnemy.transform.position = _enemyPaths[0].position;
        newEnemy.SetTargetPosition (_enemyPaths[1].position);
        newEnemy.SetCurrentPathIndex (1);
        newEnemy.gameObject.SetActive (true);
    }

    public Bullet GetBulletFromPool (Bullet prefab)
    {
        GameObject newBulletObj = _spawnedBullets.Find (
            b => !b.gameObject.activeSelf && b.name.Contains
(prefab.name)
        )?.gameObject;

        if (newBulletObj == null)
        {
            newBulletObj = Instantiate (prefab.gameObject);
        }

        Bullet newBullet = newBulletObj.GetComponent<Bullet> ();
        if (!_spawnedBullets.Contains (newBullet))
```

```
        {
            _spawnedBullets.Add (newBullet);
        }

        return newBullet;
    }

    public void ExplodeAt (Vector2 point, float radius, int damage)
    {
        foreach (Enemy enemy in _spawnedEnemies)
        {
            if (enemy.gameObject.activeSelf)
            {
                if (Vector2.Distance (enemy.transform.position, point) <=
radius)
                {
                    enemy.ReduceEnemyHealth (damage);
                }
            }
        }
    }
// ...
```

- Drag sprite bullet_1 ke dalam Hierarchy untuk membuat game object baru dan ubah namanya menjadi Bullet.
- Masukkan Box Collider 2D, ubah size-nya menjadi (0.25, 0.6) dan centang Is Trigger.
- Masukkan Rigidbody2D dan ubah mode-nya menjadi Kinematic.

- Masukkan script Bullet dalam game object Bullet.
- Drag game object Bullet ke Assets/Prefabs untuk membuat base prefab Bullet dan hapus yang sebelumnya di Hierarchy.
- Buat 2 variant dari prefab tersebut.
- Ubahlah sprite pada variant yg kedua menjadi bullet_2 dan atur size Box Collider 2D nya menjadi (0.35, 0.75).
- Tambahkan kode berikut pada script Tower, lalu masukkan prefab Bullet 1 pada Tower 1 dan 2, kemudian prefab Bullet 2 pada Tower 3.

```
using System.Collections.Generic;
// ...
    [SerializeField] private float _bulletSpeed = 1f;
    [SerializeField] private float _bulletSplashRadius = 0f;

    [SerializeField] private Bullet _bulletPrefab;

    private float _runningShootDelay;
    private Enemy _targetEnemy;
    private Quaternion _targetRotation;

    // Mengecek musuh terdekat
    public void CheckNearestEnemy (List<Enemy> enemies)
    {
        if (_targetEnemy != null)
        {
            if (!_targetEnemy.gameObject.activeSelf || Vector3.Distance
(transform.position, _targetEnemy.transform.position) >
_shootDistance)
            {
                _targetEnemy = null;
            }
            else
            {
                return;
            }
```

```
        }

        float nearestDistance = Mathf.Infinity;
        Enemy nearestEnemy = null;

        foreach (Enemy enemy in enemies)
        {
            float distance = Vector3.Distance (transform.position,
enemy.transform.position);
            if (distance > _shootDistance)
            {
                continue;
            }

            if (distance < nearestDistance)
            {
                nearestDistance = distance;
                nearestEnemy = enemy;
            }
        }

        _targetEnemy = nearestEnemy;
    }

    // Menembak musuh yang telah disimpan sebagai target
    public void ShootTarget ()
```

```
    {
        if (_targetEnemy == null)
        {
            return;
        }

        _runningShootDelay -= Time.unscaledDeltaTime;
        if (_runningShootDelay <= 0f)
        {
            bool headHasAimed = Mathf.Abs
(_towerHead.transform.rotation.eulerAngles.z -
_targetRotation.eulerAngles.z) < 10f;
            if (!headHasAimed)
            {
                return;
            }

            Bullet bullet = LevelManager.Instance.GetBulletFromPool
(_bulletPrefab);
            bullet.transform.position = transform.position;
            bullet.SetProperties (_shootPower, _bulletSpeed,
_bulletSplashRadius);
            bullet.SetTargetEnemy (_targetEnemy);
            bullet.gameObject.SetActive (true);

            _runningShootDelay = _shootDelay;
        }
```

```
    }


    // Membuat tower selalu melihat ke arah musuh
    public void SeekTarget ()
    {
        if (_targetEnemy == null)
        {
            return;
        }


        Vector3 direction = _targetEnemy.transform.position -
transform.position;
        float targetAngle = Mathf.Atan2 (direction.y, direction.x) *
Mathf.Rad2Deg;
        _targetRotation = Quaternion.Euler (new Vector3 (0f, 0f,
targetAngle - 90f));


        _towerHead.transform.rotation = Quaternion.RotateTowards
(_towerHead.transform.rotation, _targetRotation, Time.deltaTime *
180f);
    }
// ...
```

- Tambahkan juga script berikut pada LevelManager untuk menggerakkan Tower:

```
private void Update ()
    {
        // Counter untuk spawn enemy dalam jeda waktu yang ditentukan
        // Time.unscaledDeltaTime adalah deltaTime yang independent, tidak terpengaruh oleh apapun kecuali game object itu sendiri,
        // jadi bisa digunakan sebagai penghitung waktu
        _runningSpawnDelay -= Time.unscaledDeltaTime;
        if (_runningSpawnDelay <= 0f)
        {
            SpawnEnemy ();
            _runningSpawnDelay = _spawnDelay;
        }

        foreach (Tower tower in _spawnedTowers)
        {
            tower.CheckNearestEnemy (_spawnedEnemies);
            tower.SeekTarget ();
            tower.ShootTarget ();
        }
// ...
```

- Setelah di play, tower akan dapat menyerang musuh yang mendekat.

9. Menambah Audio

Menambahkan Audio

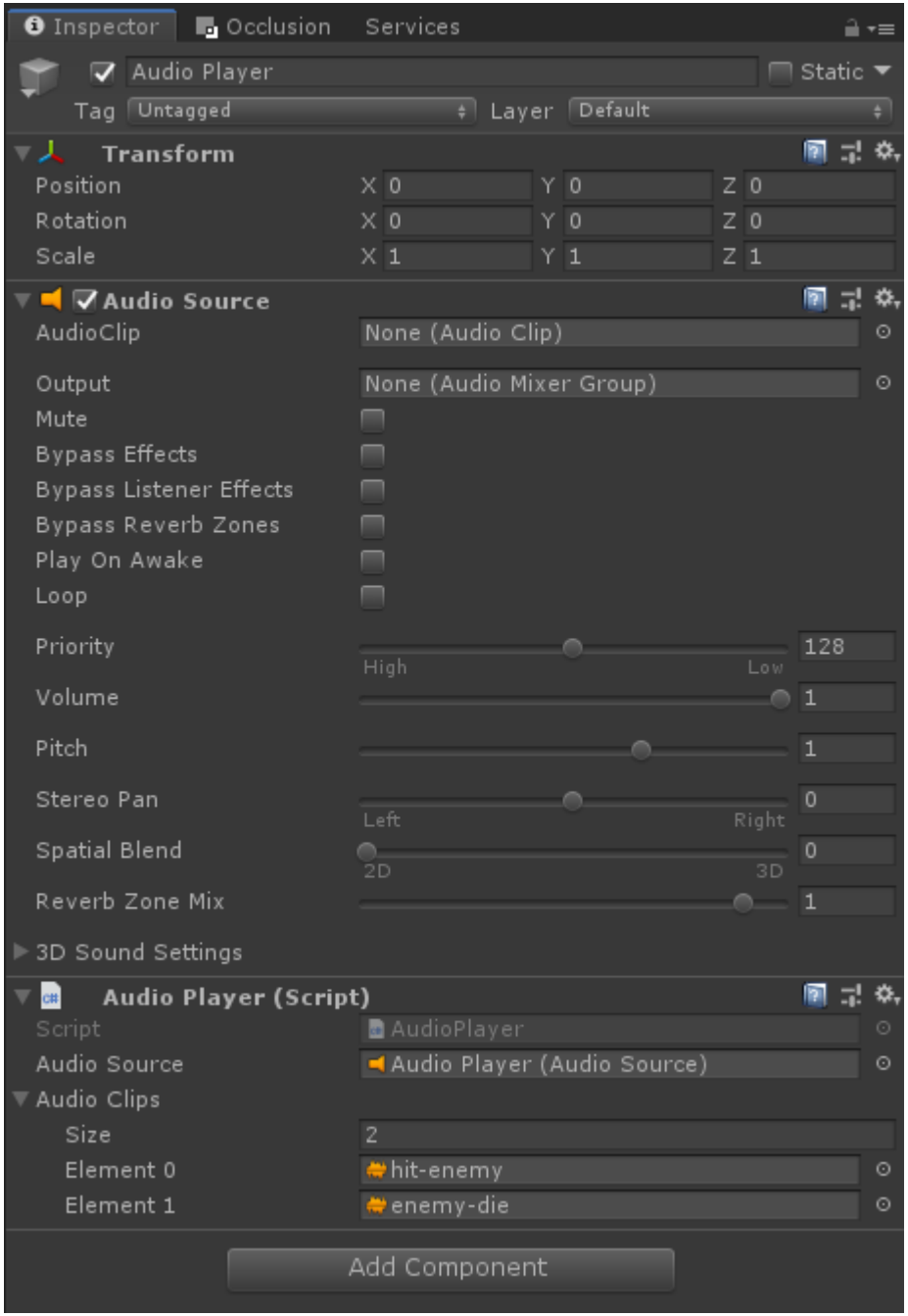
1. Menambahkan Pemutar Audio

- Buatlah game object kosong bernama Audio Player dan tambahkan komponen Audio Source di dalamnya. Uncheck bagian Play On Awake.
- Buatlah script baru bernama AudioPlayer dengan kode berikut.

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

public class AudioPlayer : MonoBehaviour
{
    private static AudioPlayer _instance = null;
    public static AudioPlayer Instance
    {
        get
        {
            if (_instance == null)
            {
                _instance = FindObjectOfType<AudioPlayer> ();
            }

            return _instance;
        }
    }

    [SerializeField] private AudioSource _audioSource;
    [SerializeField] private List<AudioClip> _audioClips;

    public void PlaySFX (string name)
    {
        AudioClip sfx = _audioClips.Find (s => s.name == name);
        if (sfx == null)
```

```
        {
            return;
        }

        _audioSource.PlayOneShot (sfx);
    }
}
```

- Tambahkan script tersebut ke dalam game object Audio Player, lalu assign value Audio Source dengan Audio Source yang baru dibuat dan masukkan setiap Audio Clip pada folder Audios.



2. Memainkan Audio dengan Script.

- Buka script Enemy dan tambahkan kode berikut untuk memutar audio.

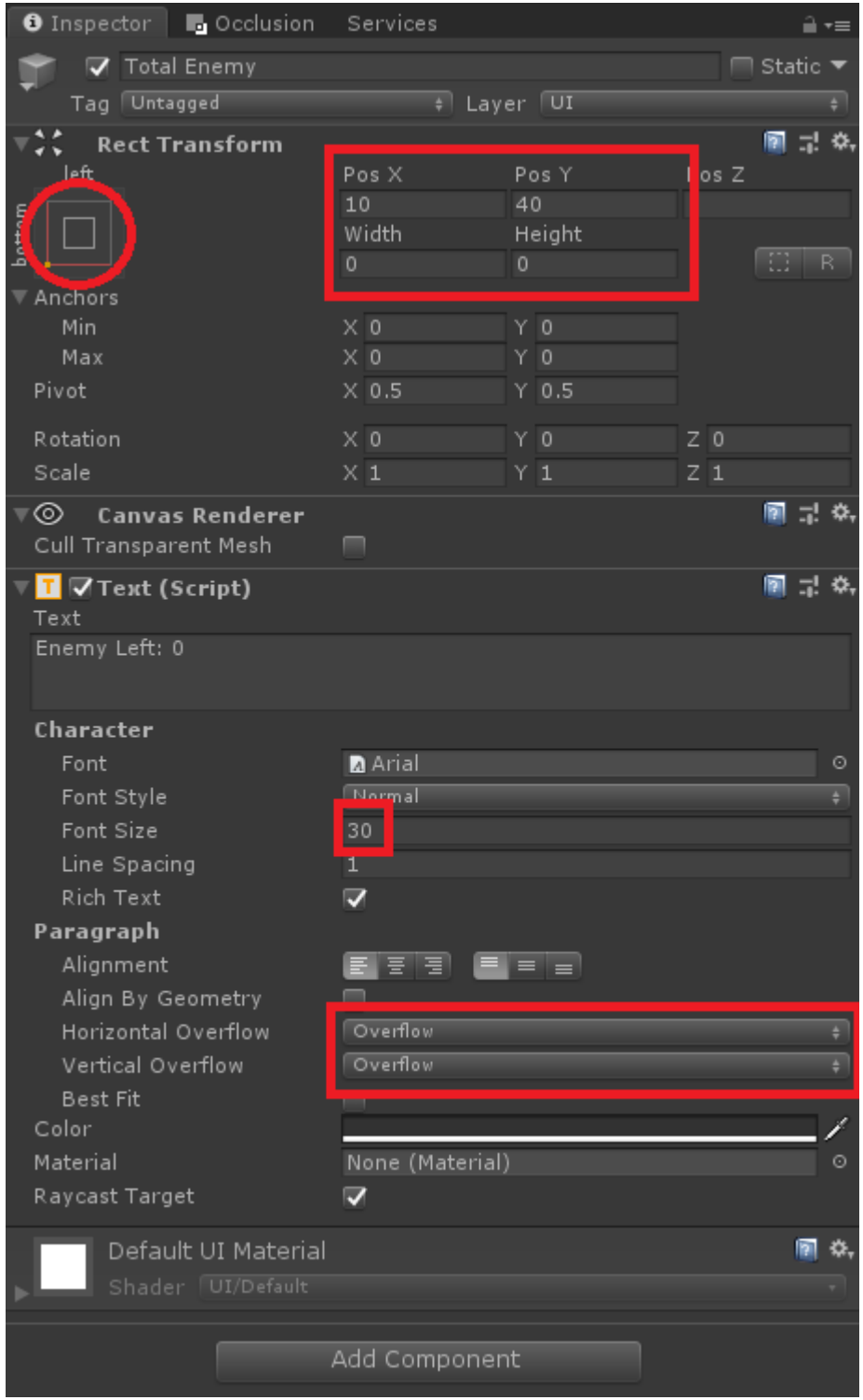
```
// ...
   public void ReduceEnemyHealth (int damage)
   {
      _currentHealth -= damage;
      AudioPlayer.Instance.PlaySFX ("hit-enemy");

      if (_currentHealth <= 0)
      {
         _currentHealth = 0;
         gameObject.SetActive (false);
         AudioPlayer.Instance.PlaySFX ("enemy-die");
      }

      float healthPercentage = (float) _currentHealth / _maxHealth;
      _healthFill.size = new Vector2 (healthPercentage * _healthBar.size.x,
_healthBar.size.y);
   }
// ...
```

- Setelah itu kamu dapat Play game-nya dengan audio yang akan terdengar ketika peluru mengenai musuh dan ketika musuh mati.

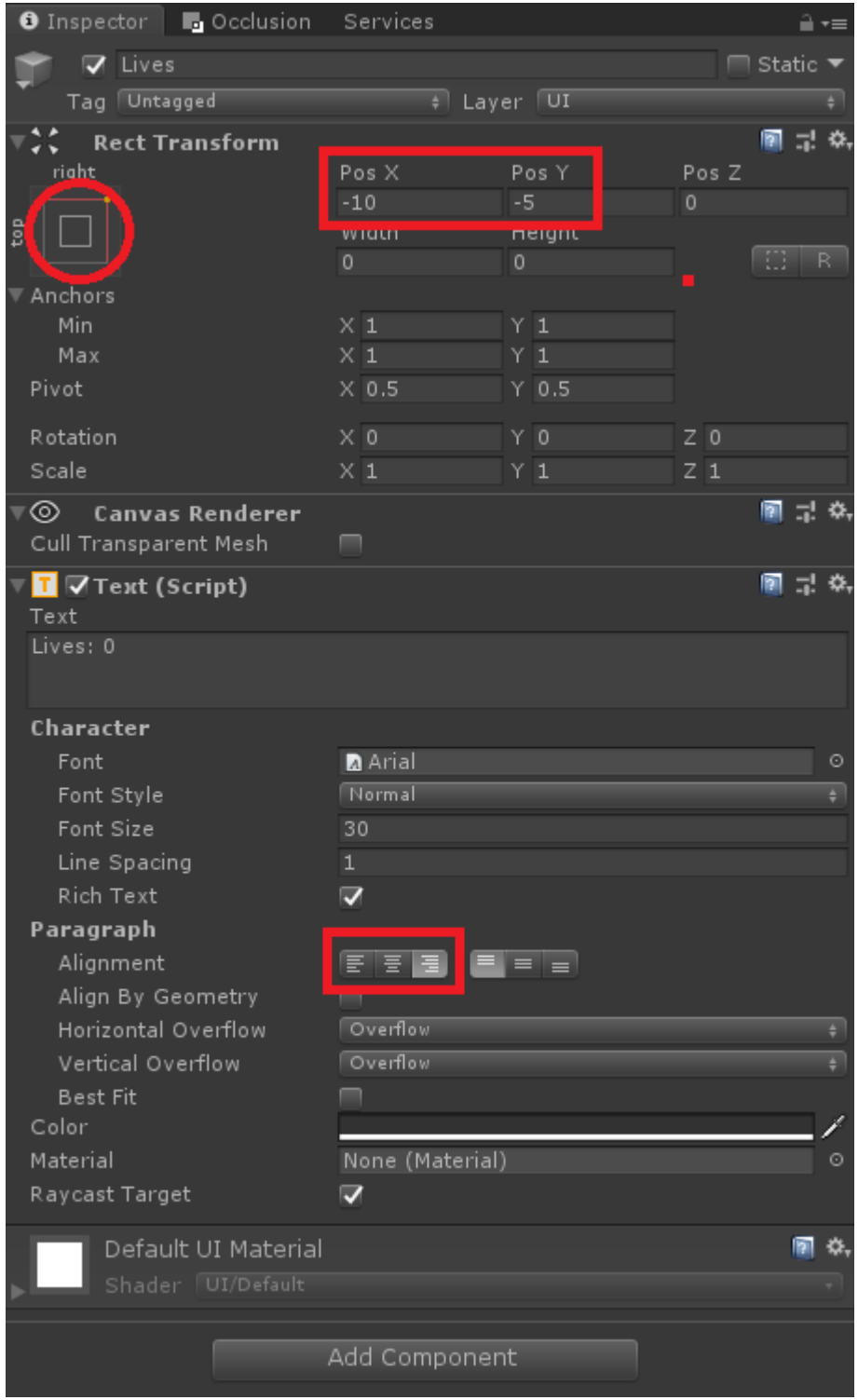
10. Win/Lose Condition

Win/Lose Condition

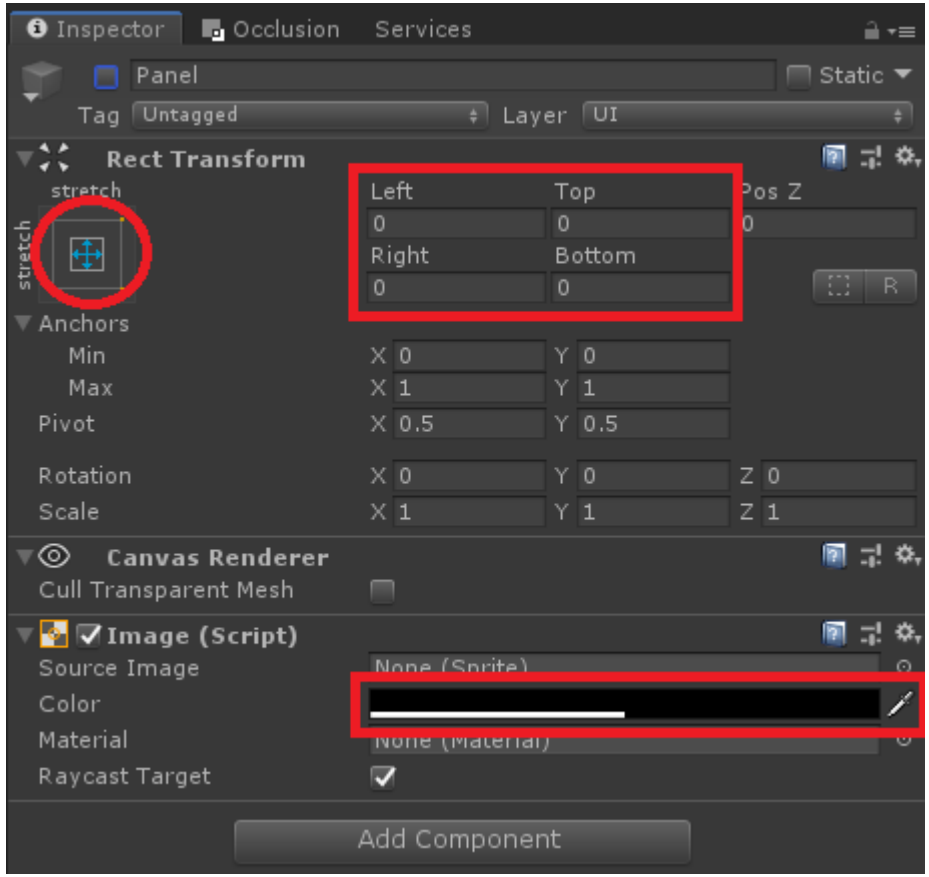
- Buat game object UI Text baru bernama Total Enemy. Lalu ubah properti-nya seperti ini.



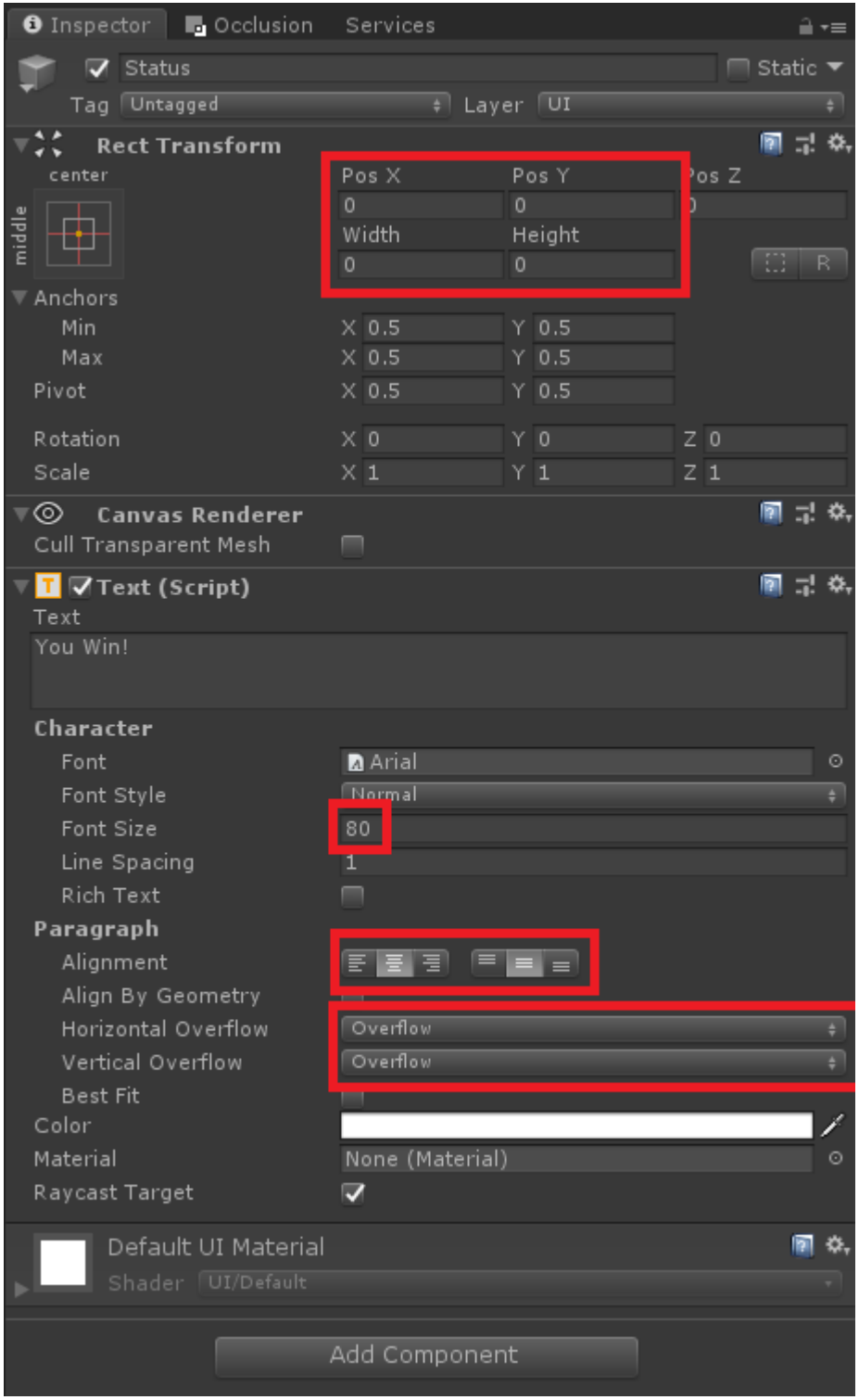
- Duplicate game object Total Enemy dan ubah namanya menjadi Lives. Lalu ubah properti-nya seperti ini.



- Buatlah game object Image baru di dalam Canvas. Ubah properti-nya seperti ini (terlihat agak transparan). Ubah namanya menjadi Panel.



- Buatlah game object Text sebagai child dari Panel. Ubah namanya menjadi Status dan ganti properti-nya menjadi seperti ini.


Inspector
Occlusion
Services
Status
Static
Tag
Untagged
Layer
UI
Rect Transform
center
middle
Pos X
Pos Y
Pos Z
0
0
0
Width
Height
0
0
R
Anchors
Min
X 0.5
Y 0.5
Max
X 0.5
Y 0.5
Pivot
X 0.5
Y 0.5
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Canvas Renderer
Cull Transparent Mesh
Text (Script)
Text
You Win!
Character
Font
Arial
Font Style
Normal
Font Size
80
Line Spacing
1
Rich Text
Paragraph
Alignment
Align By Geometry
Horizontal Overflow
Overflow
Vertical Overflow
Overflow
Best Fit
Color
Material
None (Material)
Raycast Target
Default UI Material
Shader
UI/Default
Add Component

- Untuk sekarang, matikan game object panel agar tidak menutupi layar sebelum game berakhir.
- Tambahkan kode berikut pada script LevelManager untuk dapat merestart kembali gamenya selama dijalankan.

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
// …
   private float _runningSpawnDelay;


   public bool IsOver { get; private set; }
// ...
   private void Update ()
   {
      // Jika menekan tombol R, fungsi restart akan terpanggil
      if (Input.GetKeyDown (KeyCode.R))
      {
         SceneManager.LoadScene (SceneManager.GetActiveScene
().name);
      }


      if (IsOver)
      {
         return;
      }
// ...
```

- Tambahkan kode berikut juga untuk menambahkan nyawa pemain dan jumlah musuh.

```
// ...
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class LevelManager : MonoBehaviour
{
    // Fungsi Singleton
    private static LevelManager _instance = null;
    public static LevelManager Instance
    {
        get
        {
            if (_instance == null)
            {
                _instance = FindObjectOfType<LevelManager> ();
            }

            return _instance;
        }
    }

    [SerializeField] private int _maxLives = 3;
    [SerializeField] private int _totalEnemy = 15;

    [SerializeField] private GameObject _panel;
    [SerializeField] private Text _statusInfo;
```

```
    [SerializeField] private Text _livesInfo;
    [SerializeField] private Text _totalEnemyInfo;
// ...
    private List<Tower> _spawnedTowers = new List<Tower> ();
    private List<Enemy> _spawnedEnemies = new List<Enemy> ();
    private List<Bullet> _spawnedBullets = new List<Bullet> ();

    private int _currentLives;
    private int _enemyCounter;
    private float _runningSpawnDelay;

    public bool IsOver { get; private set; }

    private void Start ()
    {
        SetCurrentLives (_maxLives);
        SetTotalEnemy (_totalEnemy);
// ...
    private void SpawnEnemy ()
    {
        SetTotalEnemy (--_enemyCounter);
        if (_enemyCounter < 0)
        {
            bool isAllEnemyDestroyed = _spawnedEnemies.Find (e =>
e.gameObject.activeSelf) == null;
            if (isAllEnemyDestroyed)
```

```
            {
                SetGameOver (true);
            }

            return;
        }
// ...
    public void ExplodeAt (Vector2 point, float radius, int damage)
    {
        foreach (Enemy enemy in _spawnedEnemies)
        {
            if (enemy.gameObject.activeSelf)
            {
                if (Vector2.Distance (enemy.transform.position, point) <= radius)
                {
                    enemy.ReduceEnemyHealth (damage);
                }
            }
        }
    }

    public void ReduceLives (int value)
    {
        SetCurrentLives (_currentLives - value);
        if (_currentLives <= 0)
        {
```

```
            SetGameOver (false);
        }
    }

    public void SetCurrentLives (int currentLives)
    {
        // Mathf.Max fungsi nya adalah mengambil angka terbesar
        // sehingga _currentLives di sini tidak akan lebih kecil dari 0
        _currentLives = Mathf.Max (currentLives, 0);
        _livesInfo.text = $"Lives: {_currentLives}";
    }

    public void SetTotalEnemy (int totalEnemy)
    {
        _enemyCounter = totalEnemy;
        _totalEnemyInfo.text = $"Total Enemy: {Mathf.Max
(_enemyCounter, 0)}";
    }

    public void SetGameOver (bool isWin)
    {
        IsOver = true;

        _statusInfo.text = isWin ? "You Win!" : "You Lose!";
        _panel.gameObject.SetActive (true);
    }
```

```
// ...
```

- Masukkan game object Total Enemy, Lives, Panel, dan Status pada variable yang baru dibuat di Inspector Level 1.
- Lalu, tambahkan pada scipt Bullet kode berikut agar peluru berhenti ketika game berakhir.

```
private void FixedUpdate ()
   {
      if (LevelManager.Instance.IsOver)
      {
         return;
      }
// ...
```

- Selamat! Kamu sudah bisa memainkan game-nya dengan kondisi menang/kalah.