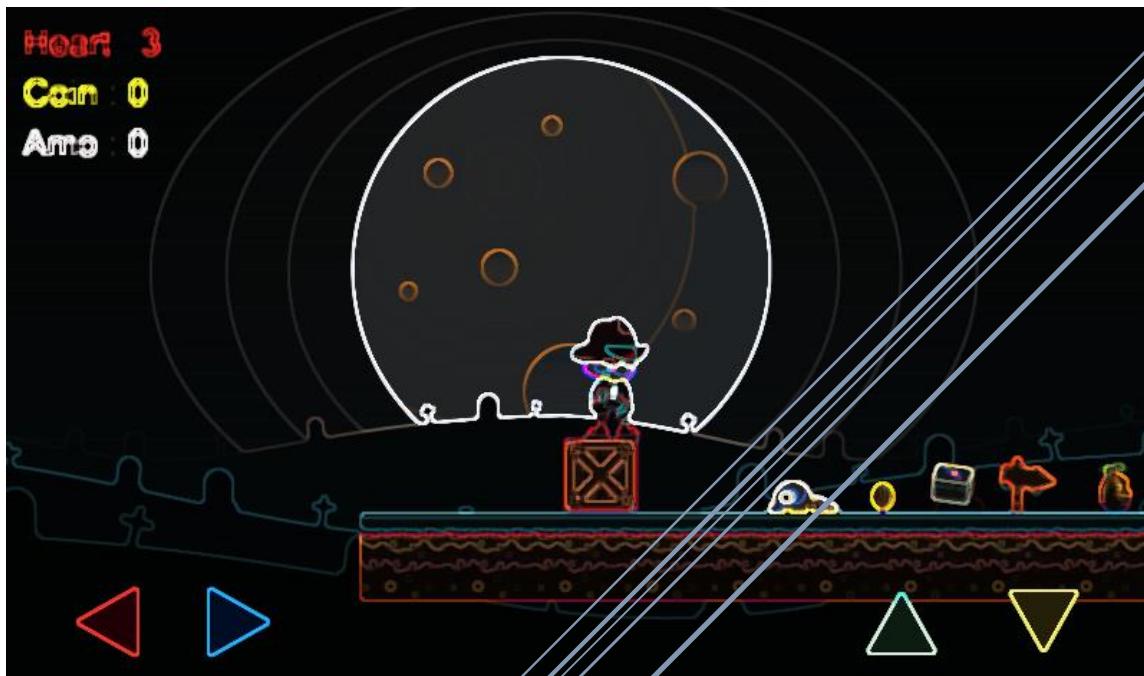


# MODUL UNITY 2019

Membuat Game 2d Platformers



Rio Andriyat Krisdiawan  
2019

<https://staff.uniku.ac.id/rioandriyat>

<https://www.youtube.com/channel/UCqhJbgpBuvhPVUpb-jo-kQ>

## DAFTAR ISI

	Halaman
<b>DAFTAR ISI.....</b>	<b>1</b>
<b>A. Pengenalan Unity .....</b>	<b>4</b>
1. Teori Dasar.....	4
2. Interface Unity 2D .....	5
3. Praktikum.....	5
4. Kesimpulan .....	9
<b>B. Pengenalan C# .....</b>	<b>10</b>
1. Teori Dasar.....	10
2. Praktikum.....	10
3. Kesimpulan .....	14
<b>C. Asset Unity.....</b>	<b>15</b>
1. Teori Dasar.....	15
2. Praktikum.....	16
3. Latihan Project .....	20
<b>D. Rigidbody Dan Collider.....</b>	<b>21</b>
1. Teori Dasar.....	21
<b>Rigidbody .....</b>	<b>21</b>
<b>Tipe Body Rigidbody .....</b>	<b>21</b>
<b>2D collider .....</b>	<b>22</b>
2. Praktikum.....	22
3. Latihan Project .....	26
<b>E. Control Player dan Sensor Tanah (Layer) .....</b>	<b>27</b>
1. Teori Dasar.....	27
2. Praktikum.....	27
3. Latihan Project .....	34
<b>F. Animasi .....</b>	<b>35</b>
1. Teori Dasar.....	35
2. Praktikum.....	36
3. Latihan Project .....	47
<b>G. Camera Follow .....</b>	<b>48</b>

1.	Teori Dasar.....	48
2.	Praktikum.....	48
3.	Latihan Project.....	50
H.	Collider dan Physic .....	51
1.	Teori Dasar.....	51
	<b>2D collider.....</b>	51
	<b>Physic 2D.....</b>	51
	<b>Properti .....</b>	52
2.	Praktikum.....	52
3.	Latihan Project .....	59
I.	Transform/Check Point.....	61
1.	Teori Dasar.....	61
2.	Praktikum.....	61
3.	Latihan Project.....	64
J.	User Interface.....	65
1.	Teori Dasar.....	65
2.	Praktikum.....	65
3.	Latihan Project .....	69
K.	UI Control .....	70
1.	Teori Dasar.....	70
	<b>Kanvas.....</b>	70
	Gambar urutan elemen.....	70
	Mode Render.....	70
	Layar Ruang - Hamparan.....	70
	Layar Space - Kamera.....	71
	Ruang Dunia .....	72
2.	Praktikum.....	72
3.	Latihan Project .....	80
L.	UI Win and Lose.....	81
1.	Teori Dasar.....	81
	The Rect Tool .....	81
	Rect Transform .....	81

Mengubah Ukuran Versus Penskalaan .....	82
Poros .....	82
2. Praktikum.....	82
3. Latihan Project.....	85
M. Main Menu.....	86
1. Praktikum.....	86
2. Latihan Project.....	89
N. Build Game .....	90
1. Teori Dasar.....	90
Pengaturan Build.....	90
Adegan dalam Build .....	90
Daftar platform.....	90
2. Praktikum.....	91
3. Latihan Project.....	94
O. Glossary .....	95
1. Tab Unity .....	95
2. Tab Hirarcy .....	96
3. Transform Tools.....	96
4. Tab Scene.....	96
5. Other .....	96
REFERENCE.....	98

## A. Pengenalan Unity

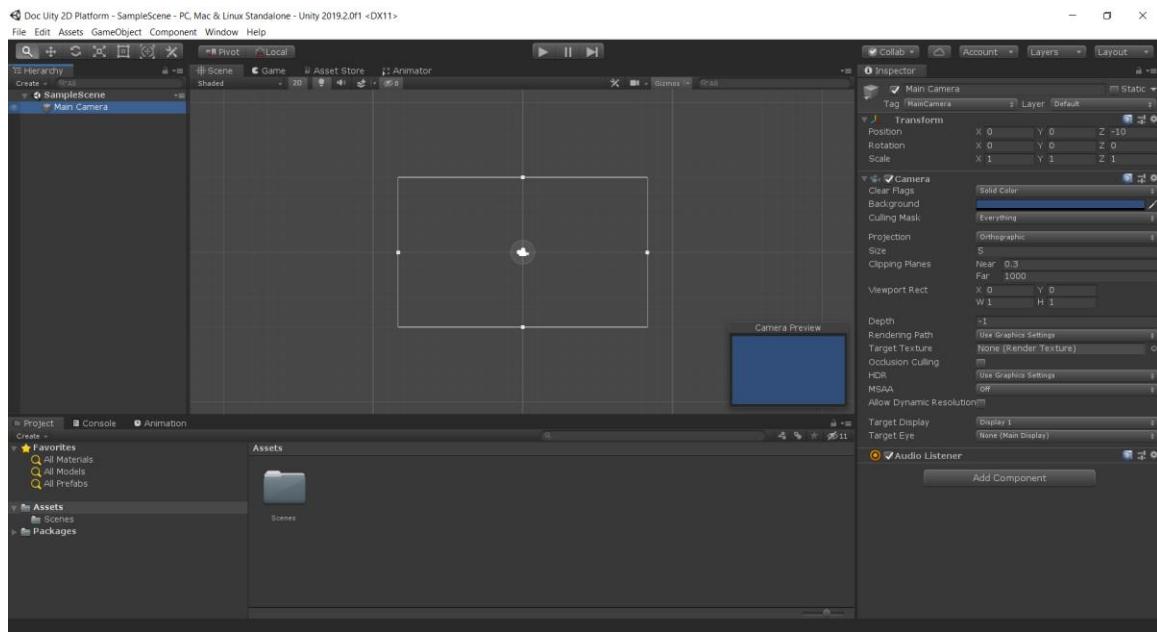
### 1. Teori Dasar

**Unity** adalah lintas platform game engine yang dikembangkan oleh **Unity Technologies**. Unity pertama kali diumumkan dan dirilis pada **Juni 2005** di World Wide Developers Conference milik Apple Inc, sebagai **game engine eksklusif OS X**. Pada 2018, mesin telah diperluas untuk mendukung **27 platform**. Mesin ini dapat digunakan untuk membuat game 3D dan 2D serta simulasi untuk Berbagai platform. Unity telah diadopsi oleh industri di luar permainan video, seperti film , otomotif, arsitektur , teknik dan konstruksi . Beberapa versi utama Unity telah dirilis sejak diluncurkan, dengan versi stabil terbaru adalah 2019.2.12, dirilis pada November 2019.

**Unity** memberi pengguna kemampuan untuk membuat game dalam 2D dan 3D, dan engine menawarkan API skrip yang ditulis dalam **C #**, untuk editor Unity dalam bentuk plugin, dan game itu sendiri, serta fungsionalitas fitur *drag and drop*. Sebelum C # menjadi bahasa pemrograman utama yang digunakan ,sebelumnya mendukung Boo, yang telah dihapus dalam rilis Unity 5 , dan versi JavaScript yang disebut UnityScript, yang dihentikan pada Agustus 2017 setelah rilis Unity 2017.1 mendukung C #.

Dalam game 2D, Unity memungkinkan import **sprite** dan **renderer 2D** yang canggih. Untuk game 3D, Unity memungkinkan spesifikasi kompresi tekstur, *mipmaps*, dan pengaturan resolusi untuk setiap platform yang didukung oleh engine game, dan menyediakan dukungan untuk pemetaan lekukan, pemetaan refleksi, pemetaan *paralaks*, oklusi ruang layar sekeliling (SSAO), bayangan yang dinamis menggunakan peta bayangan, efek render ke tekstur dan pasca-pemrosesan layar penuh. Unity mendukung pembuatan vertex, fragmen (atau piksel) khusus, *tessellation*, *compute shader*, dan shader permukaan Unity sendiri menggunakan CG, versi modifikasi dari Bahasa Shading Tingkat Tinggi Microsoft yang dikembangkan oleh Nvidia.

## 2. Interface Unity 2D

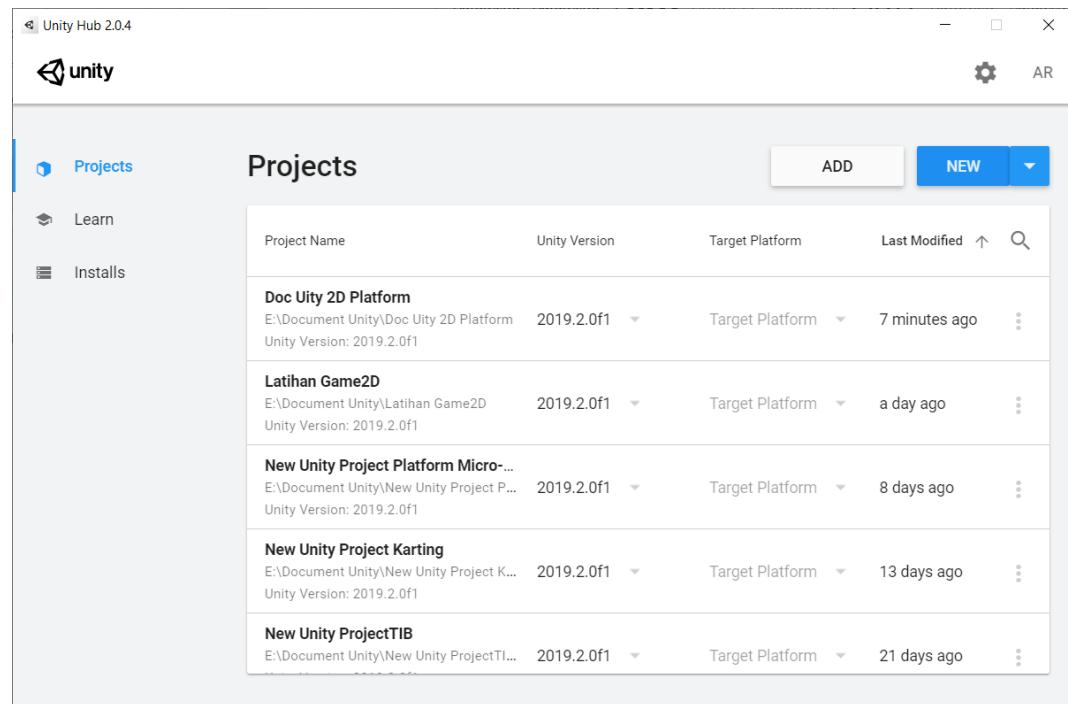


Ketika pertama kali membuka Unity dan setup project, akan muncul halaman yang berisikan beberapa window. Terdapat enam window utama yang akan menjadi fokus dan yang akan sering digunakan ketika mengembangkan game menggunakan Unity. Kelima window tersebut adalah **Hierarchy**, **Inspector**, **Project**, **Console**, **Scene** dan **Game**.

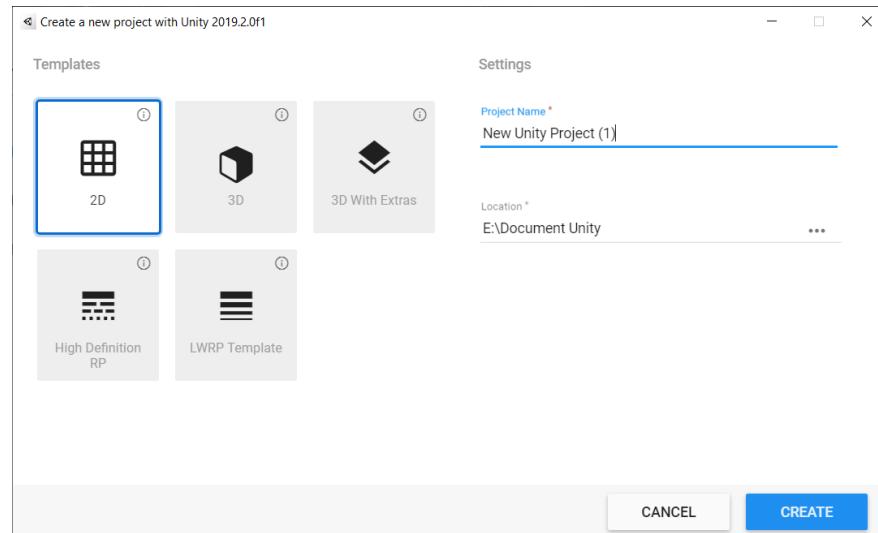
## 3. Praktikum

### a. Membuat project baru unity

- Buatlah Project 2D baru di Unity, dan berinama untuk project tersebut. (*Tampilan awal pembuatan project unity untuk versi unity 5, unity 2018, unity 2019 keatas akan sedikit berbeda pada tampilan menu, tetapi pada dasarnya fungsinya sama.*)



Saat membuat Proyek baru, dapat menentukan apakah akan memulai Unity Editor dalam mode 2D atau mode 3D. Namun, unity juga memiliki opsi untuk mengganti Editor antara mode 2D dan mode 3D kapan saja.



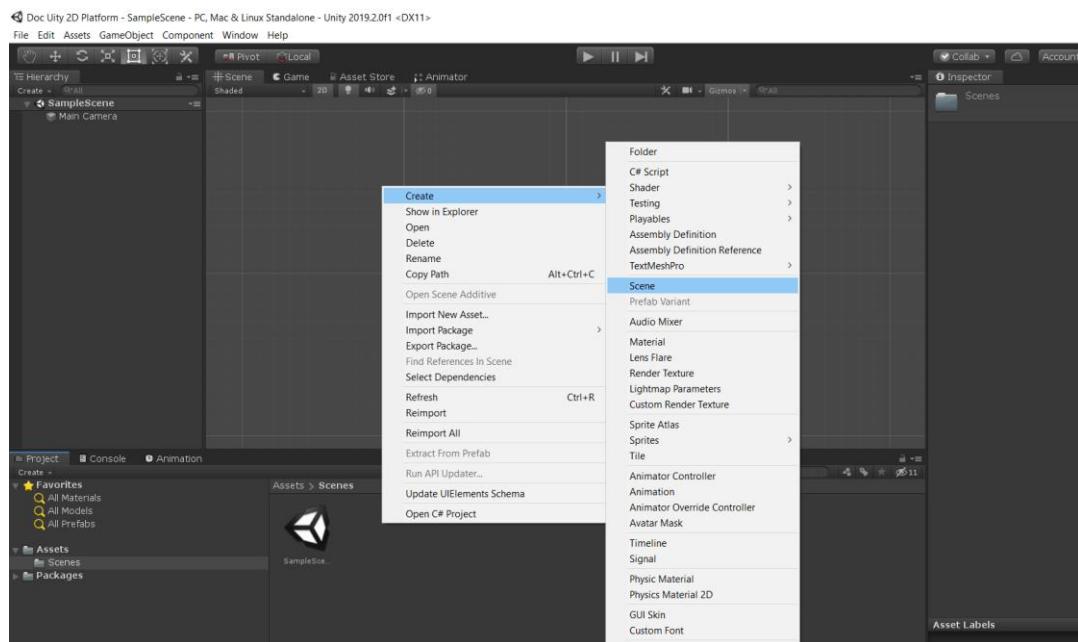
### Mode Project 2D:

- Setiap gambar yang di impor dianggap sebagai gambar 2D ( Objek **Sprite** )
- The **Sprite Packer** akan aktif.
- Tampilan **Scene View** diatur ke 2D.

- Default game object tidak secara realtime view,
- The **kamera** posisi standar adalah 0,0, -10. (0,1, -10 dalam Mode 3D.)
- Kamera diatur menjadi **Orthografis**. (Dalam Mode 3D itu **Perspektif**.)
- Di bagian lighting windows:
  - **Skybox** dinonaktifkan untuk **Scene** baru .
  - **Ambient Source** is set to **Color**. (With the color set as a dark grey: RGB: 54, 58, 66.)
  - **Precomputed Realtime GI** is set to off.
  - **Baked GI** is set to off.
  - **Auto-Building** set to off.

## b. Membuat Scene Baru

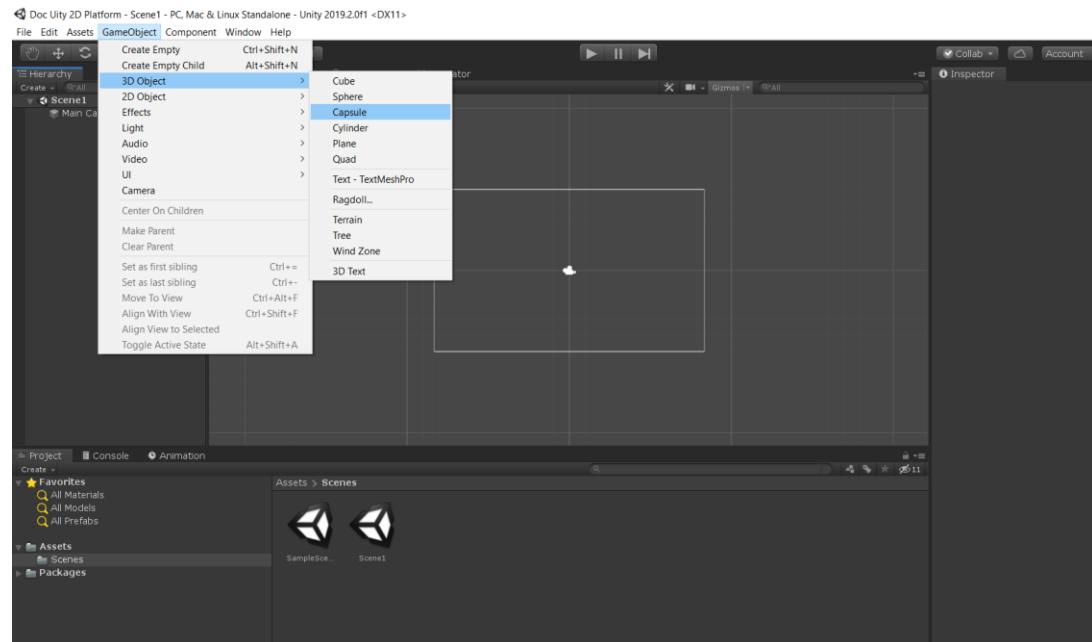
1. Pada jendela project, pilih folder **Asset** → **Scene**
2. Banyak cara untuk langkah membuat Scene baru, salah satunya dengan cara *klik kanan pada jendela project* → *Pilih Create* → **Scene**



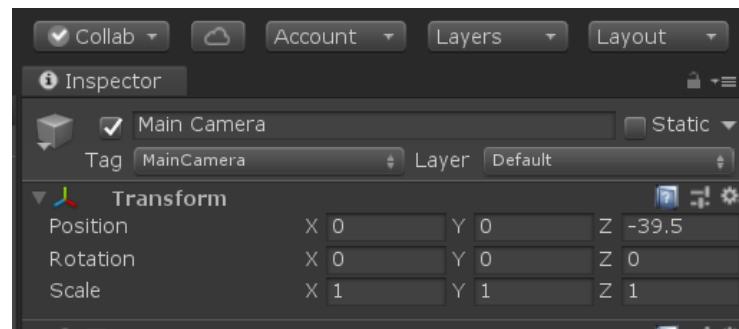
3. Selanjutnya, beri nama scene yang diinginkan → **Scene1**

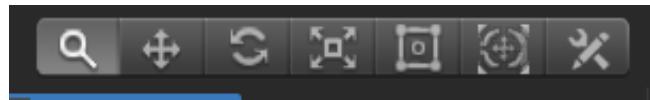
### c. Membuat dan memasukan Objek

1. Pilih *Scene 1*,
2. Pada menu bar, pilih menu game *object* → *3D Object* → *Capsule*



3. Perhatikan, maka akan ada game object Capsule pada hierarchy yang secara otomatis juga akan masuk kedalam scene
4. Pilih game object Capsule, lalu perhatikan detail pada *inspector*.
5. Di dalam windows inspector terdapat property **Transform**. untuk menentukan lokasi object bisa diatur dengan **Transform Tools** pada toolbar atau dengan pengaturan Koordinat pada property transform.

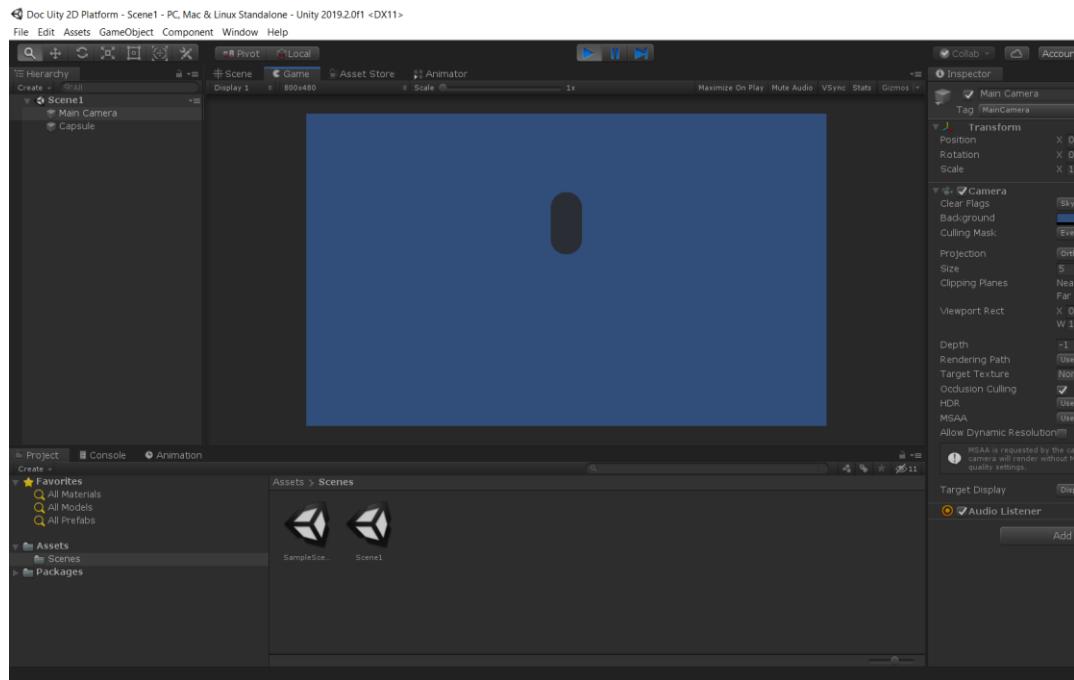




### Properti Transform

Milik:	Fungsi:
<b>Posisi</b>	Posisi Transformasi dalam koordinat X, Y, dan Z.
<b>Rotasi</b>	Rotasi Transform sekitar sumbu X, Y, dan Z, diukur dalam derajat.
<b>Skala</b>	Skala Transformasi sepanjang sumbu X, Y, dan Z. Nilai "1" adalah ukuran asli (ukuran di mana objek diimpor).

#### 6. Play Game, amati dan lihat hasil pada Game View



#### 4. Kesimpulan

- Dalam 1 project unity, didalamnya bisa terdapat berbagai macam File Scene, Script, dan Asset pembangun Game.
- Posisi objek game dalam suatu scene diatur pada property transform.

## B. Pengenalan C#

### 1. Teori Dasar

**Bahasa C#** adalah sebuah **bahasa** pemrograman modern yang bersifat general-purpose, berorientasi objek, yang dapat digunakan untuk membuat program di atas arsitektur Microsoft .NET Framework. **Bahasa C#** ini memiliki kemiripan dengan **bahasa Java**, C dan C++ (selengkapnya dapat dilihat pada Sejarah **Bahasa C#**).

### Struktur Bahasa C#

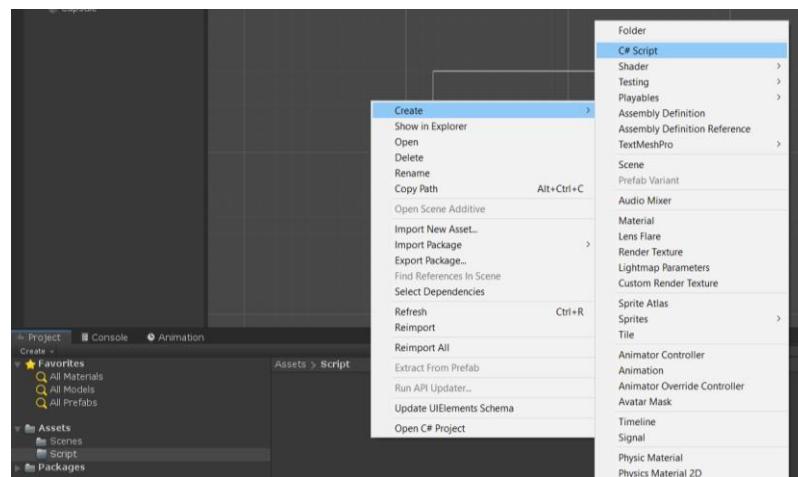
```
using System;
namespace
{
    class program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World C#!");
        }
    }
}
```

### 2. Praktikum

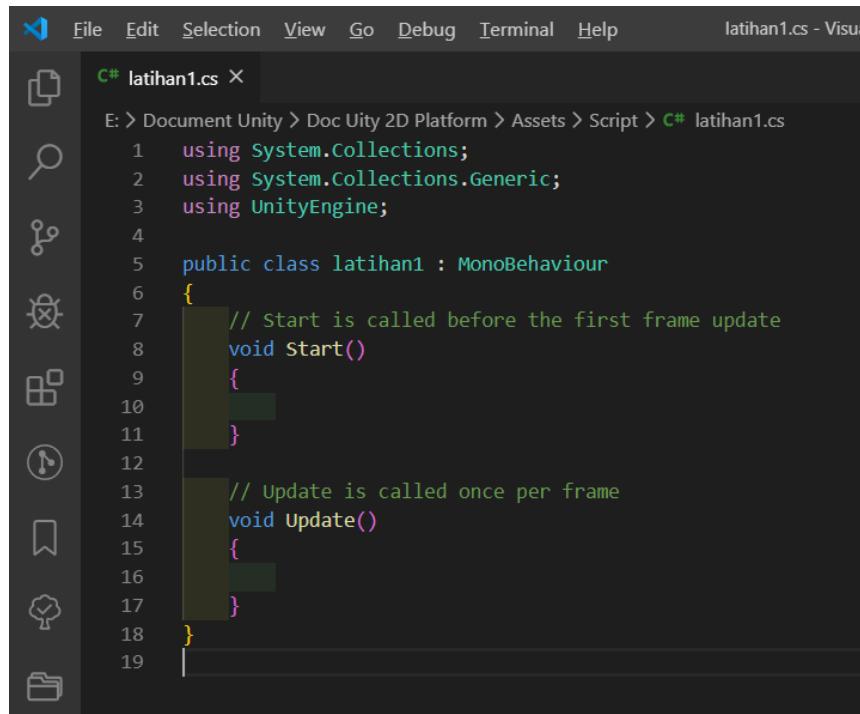
#### a. Untuk Menampilkan text di Console

\*Untuk membuat script pada project unity, alangkah baiknya mempersiapkan folder script pada project asset untuk lebih terstruktur dalam penempatan suatu file dalam suatu project.

1. Pada Folder Script di jendela project, pilih folder *Script*.
2. Klik kanan → *Pilih Create* → *C# Script*
3. Berinama file tersebut → *latihan1*



4. Klik 2x script **Latihan1.cs** untuk membukanya dengan editor.



```

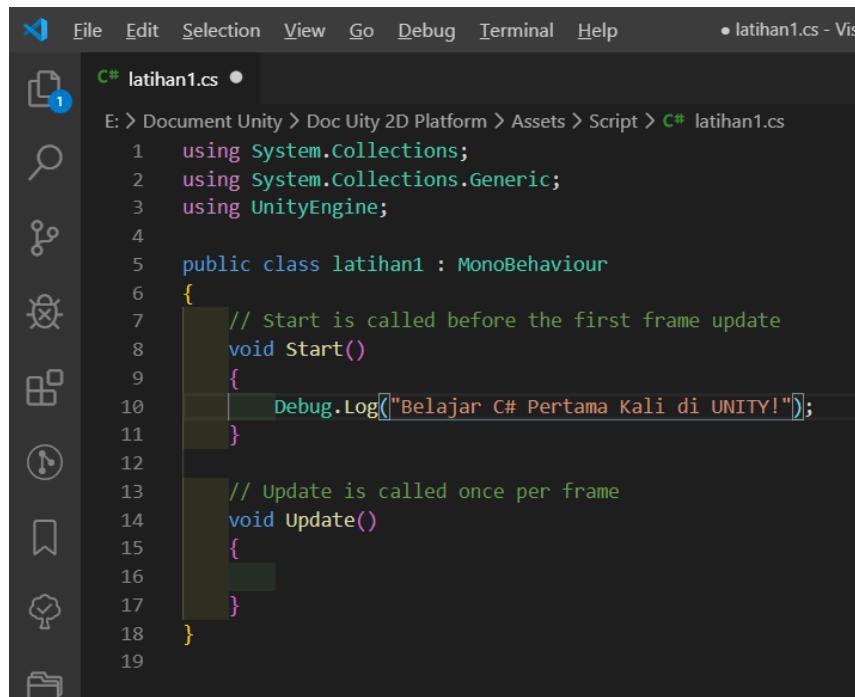
C# latihan1.cs ×
E: > Document Unity > Doc Unity 2D Platform > Assets > Script > C# latihan1.cs
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class latihan1 : MonoBehaviour
6  {
7      // Start is called before the first frame update
8      void Start()
9      {
10
11  }
12
13     // Update is called once per frame
14     void Update()
15     {
16
17  }
18
19

```

5. Kemudian muncul script **Latihan1.cs**
6. Secara default, pada script yang baru dibuat akan ditampilkan 2 buah prosedur (**Monobehaviour**), yaitu:
  - **Start()**  
Prosedur ini akan dipanggil satu kali di awal, pada saat script pertama kali di-enable. Prosedur ini sesuai untuk inisialisasi.
  - **Update()**  
Prosedur ini dipanggil pada setiap frame, apabila script di-enable. Prosedur ini sesuai untuk bagian script yang dieksekusi berulang-ulang.

*Untuk daftar prosedur dan fungsi lengkap yang ada pada kelas MonoBehaviour, silakan lihat pada link: <https://docs.unity3d.com/ScriptReference/MonoBehaviour.html>.*

## 7. Menambahkan/mengubah code di Script *Latihan1.cs*



```

C# latihan1 •
E: > Document Unity > Doc Uity 2D Platform > Assets > Script > C# latihan1.cs
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class latihan1 : MonoBehaviour
6  {
7      // Start is called before the first frame update
8      void Start()
9      {
10         Debug.Log("Belajar C# Pertama Kali di UNITY!");
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16     }
17 }
18
19

```

### Pembahasan Code:

1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;

Library di atas merupakan Library yang digunakan dalam perintah dalam script.

Seperti penggunaan variable GameObject, Monobehavior, dll.

1. public class Latihan1

Nama kelas dari sebuah script. Nama Kelas harus sama dengan nama file C# Script di folder Assets.

1. void Start()

Fungsi di atas hanya dijalankan satu kali dan diawal saat program dijalankan

1. Debug.Log("Belajar C# Pertama Kali di UNITY!");

Debug.Log digunakan untuk menghasilkan output di Jendela Console berupa text atau angka.

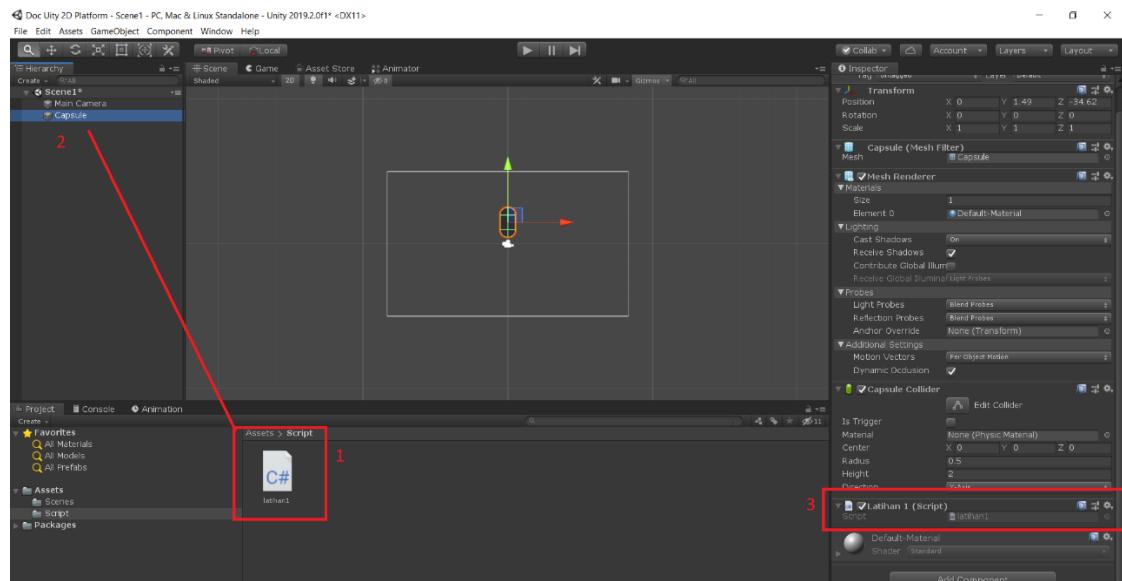
1. void Update()

Dijalankan berulang-ulang setelah menjalankan function Start().

## b. Menjalankan Script

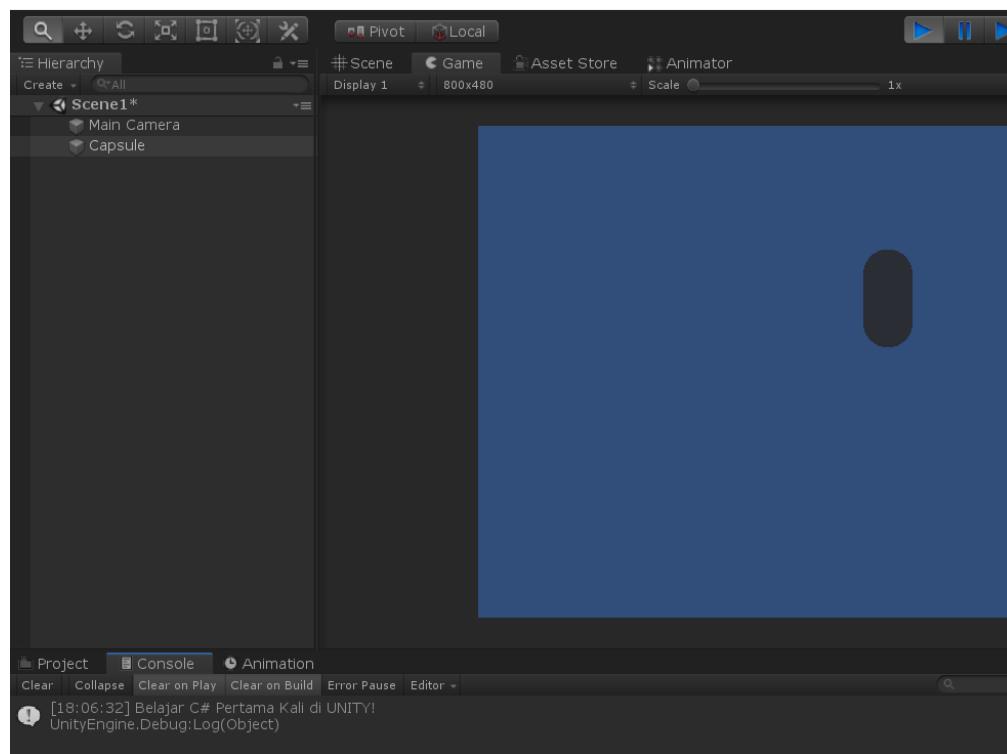
Script akan berjalan dan dapat dijalankan, apabila script ada dalam game object. Sehingga untuk menjalankan suatu script, harus dimasukan terlebih dahulu pada salah satu game object.

### 1. Drag n Drop Script **Latihan1.cs** ke GameObject (**Capsule**) di jendela **Hierarchy**

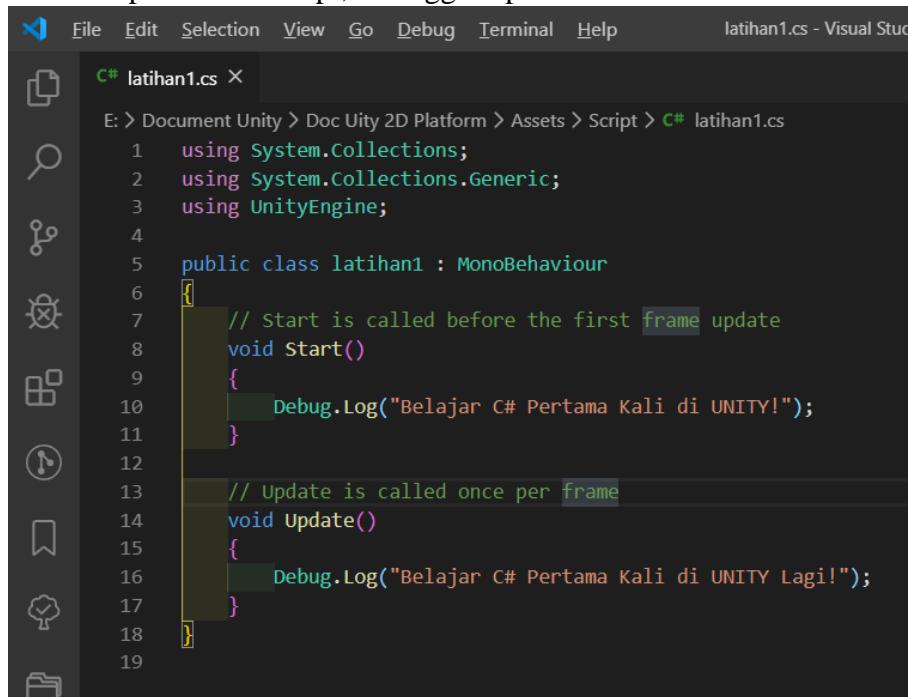


### 2. Klik icon Play pada Unity

### 3. Masuk ke bagian **Console**. Untuk melihat output program.



4. Lakukan perubahan script, sehingga seperti dibawah ini.



```
C# latihan1.cs
E: > Document Unity > Doc Unity 2D Platform > Assets > Script > C# latihan1.cs
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class latihan1 : MonoBehaviour
6  {
7      // Start is called before the first frame update
8      void Start()
9      {
10         Debug.Log("Belajar C# Pertama Kali di UNITY!");
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16         Debug.Log("Belajar C# Pertama Kali di UNITY Lagi!");
17     }
18 }
19
```

5. Simpan dan Amati hasilnya

### 3. Kesimpulan

- Using UnityEgine; , Using System.Collections;, adalah Library default ketika membuat script di unity, yang akan selalu ada sebagai default perintah ke unity engine. Library yang digunakan dalam perintah dalam script. Seperti penggunaan variable GameObject, Monobehavior, dll
- **MonoBehaviour** adalah *class* dasar dari setiap *script* Unity yang dibuat. Ketika menggunakan C# di unity, MonoBehaviour secara otomatis (*default*) akan tercipta. Sebagai kendali ke inspector objek game.

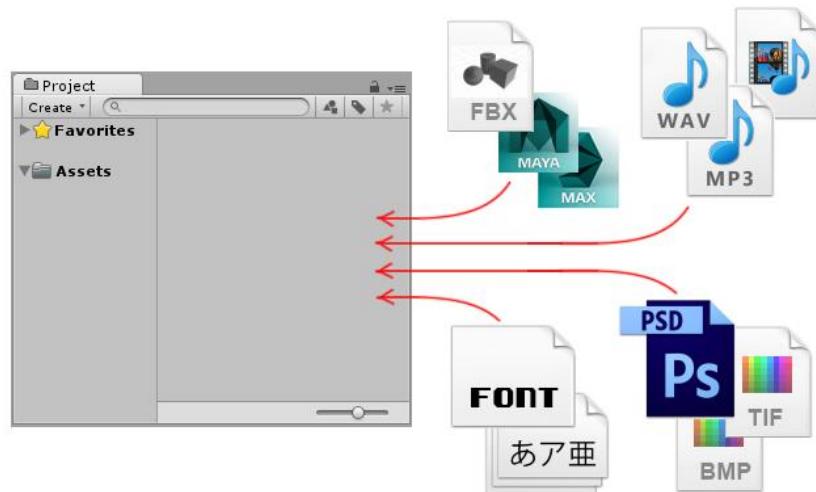
## C. Asset Unity

### 1. Teori Dasar

#### Asset

Aset adalah representasi dari item apa pun yang dapat digunakan dalam game atau Proyek unity. Aset dapat berasal dari file yang dibuat di luar Unity, seperti Model 3D, file audio, gambar, atau jenis file lainnya yang didukung Unity. Ada juga beberapa tipe Aset yang bisa dibuat di Unity, seperti Mesh, an Animator Controller, an Audio Mixer, Render Texture.

#### Asset Workflow



#### Unity Standard Assets

Unity Standard Assets dan Asset Store disediakan dalam Asset Package . Asset Package adalah kumpulan file dan data dari Unity Project, atau elemen Project, yang dikompresi dan disimpan dalam satu file, mirip dengan file zip. Seperti file zip, Asset Package mempertahankan struktur direktori aslinya saat dibongkar, serta metadata asset (seperti pengaturan impor dan tautan ke asset lain).

#### Unity Asset Store

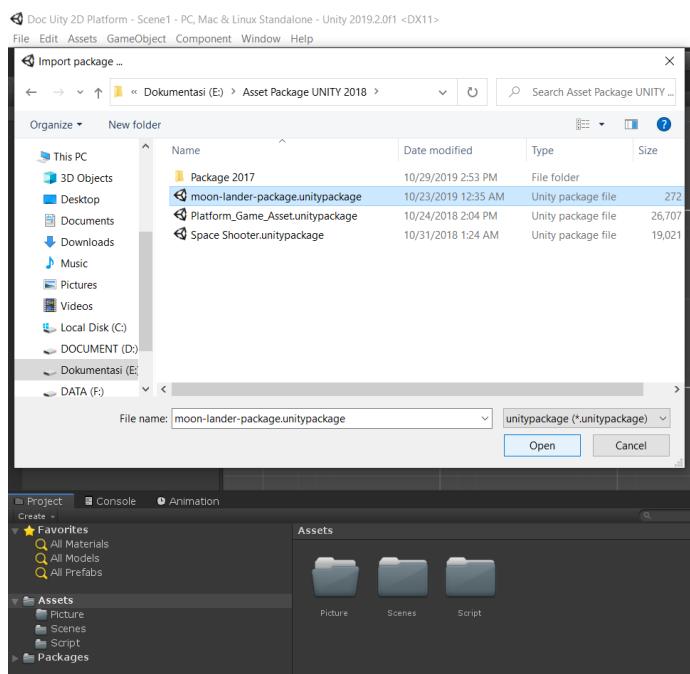
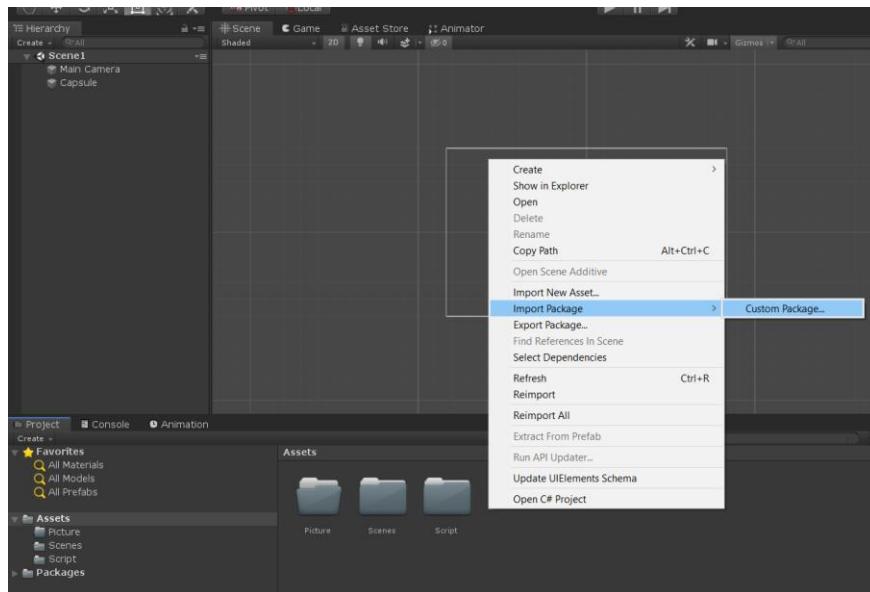
Unity Asset Store adalah rumah bagi perpustakaan yang berkembang dari Aset free dan berbayar yang diciptakan baik oleh Unity Technologies dan juga anggota komunitas. Berbagai macam Aset tersedia, yang mencakup semuanya, mulai dari Tekstur, Model, dan animasi hingga seluruh contoh Proyek, tutorial, dan ekstensi Editor. User dapat mengakses

Aset dari antarmuka sederhana yang ada di Unity Editor yang memungkinkan untuk mengunduh dan mengimpor Aset langsung ke Proyek.

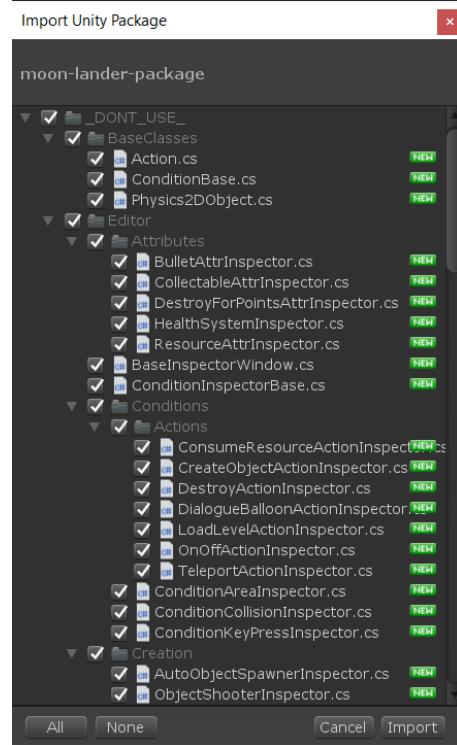
## 2. Praktikum

### a. impor paket Aset:

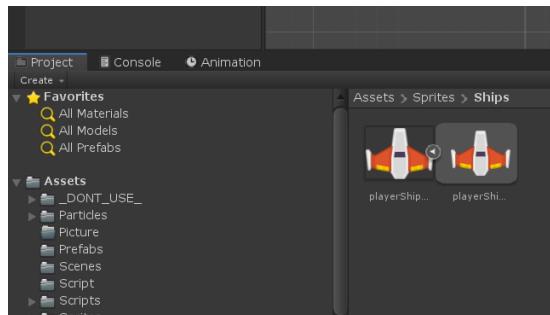
1. Buka Proyek tempat mengimpor Aset.
2. Pilih Aset → Paket Impor → Paket Kustom.



3. Di file explorer, pilih paket yang diinginkan dan kotak dialog Import Unity Package muncul, dengan semua item dalam paket sudah diperiksa sebelumnya, siap dipasang.

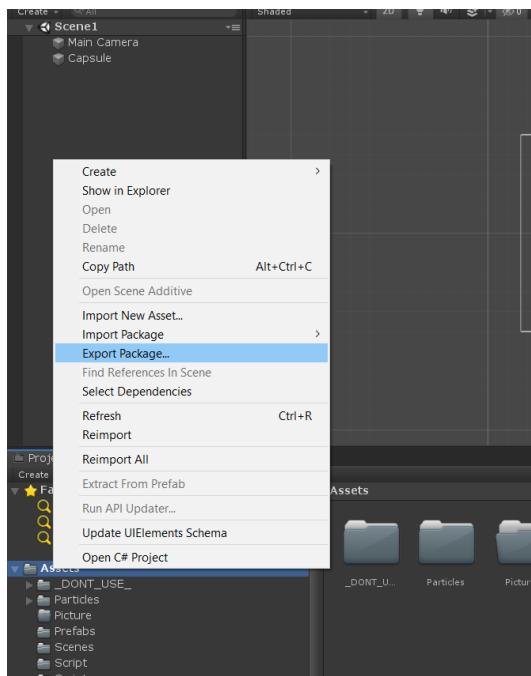


- Pilih Impor dan unity menempatkan konten paket ke folder Aset, yang dapat diakses dari tampilan Proyek.

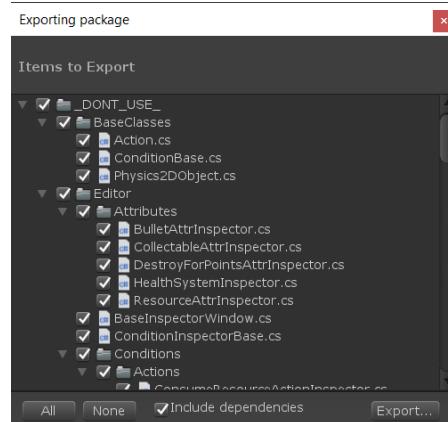


### b. Exsporting Asset packages

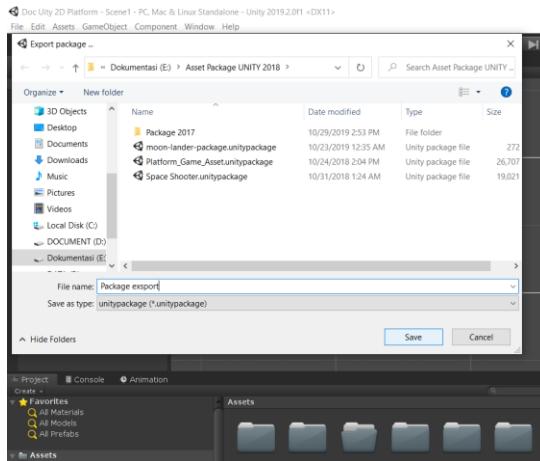
- Buka Proyek tempat asset yang ingin di ekspor.
- Pilih Aset → Paket Ekspor dari menu untuk membuka kotak dialog Paket Ekspor.



3. Di kotak dialog, pilih Aset yang ingin disertakan dalam paket dengan mengklik kotak sehingga mereka dicentang.



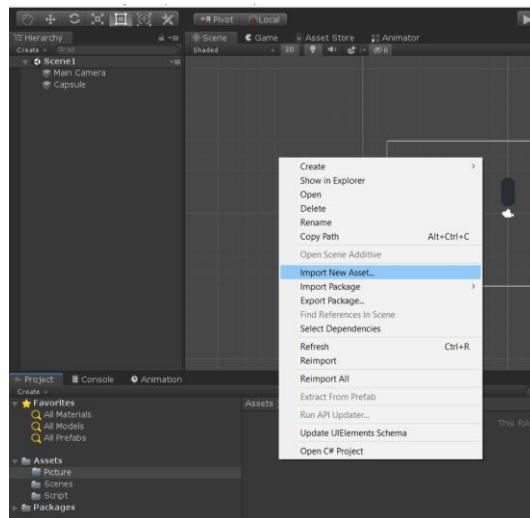
4. Biarkan kotak dependensi dicentang untuk memilih secara otomatis Aset apa pun yang digunakan oleh aset yang telah dipilih.
5. Klik Eksport untuk membuka file explorer, dan pilih di mana lokasi penyimpanan file paket.



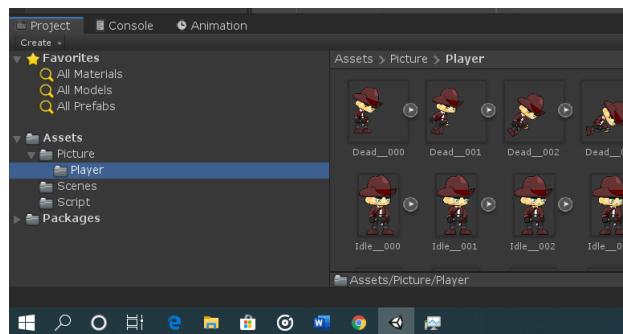
6. Beri nama dan simpan.

### c. Impor Asset Non package

1. Buka project unity,
2. Tentukan terlebih dahulu object yang akan dimasukan kedalam project game.
3. Klik kanan pada windows project → pilih *import new asset*



4. Pilih object sprite, atau png yang akan dimasukan sebagai asset game.



### 3. Latihan Project

Buatlah scene baru dan masukan beberapa asset sebagai awal pembuatan game!

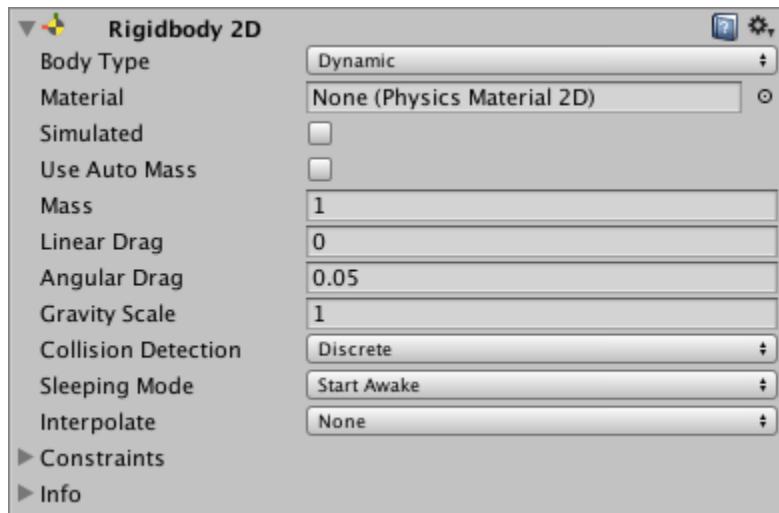


## D. Rigidbody Dan Collider

### 1. Teori Dasar

#### Rigidbody

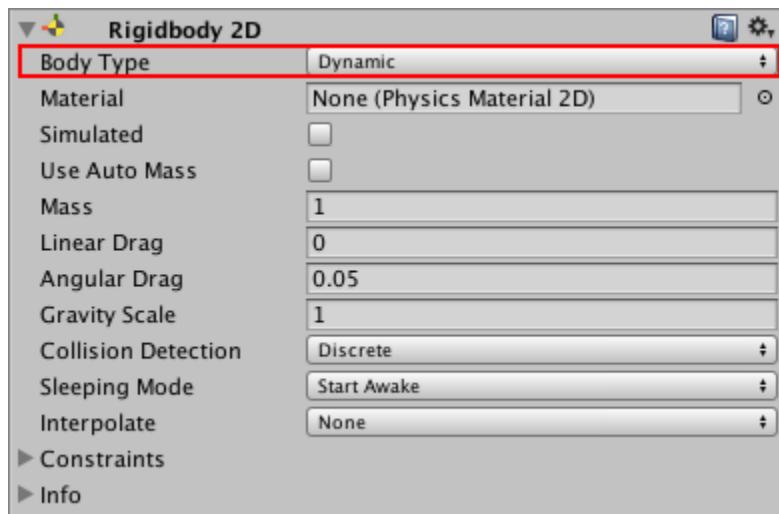
Komponen 2D Rigidbody menempatkan objek di bawah kendali mesin fisika. Rigidbody 2D, objek hanya dapat bergerak di bidang XY dan hanya dapat memutar pada sumbu yang tegak lurus terhadap bidang itu.



Komponen 2D Rigidbody muncul berbeda di Unity Editor tergantung pada Tipe Tubuh yang telah dipilih. Lihat [Tipe Tubuh](#), di bawah, untuk mempelajari lebih lanjut.

#### Tipe Body Rigidbody

Komponen 2D Rigidbody memiliki pengaturan di bagian atas berlabel tipe body . Opsi yang dipilih untuk ini memengaruhi pengaturan lain yang tersedia pada komponen.



Ada tiga opsi untuk **Tipe Body**; masing-masing mendefinisikan perilaku umum dan tetap. Setiap Collider 2D yang dilampirkan pada suatu Rigidbody 2D mewarisi Tipe Body Rigidbody 2D. Tiga opsi tersebut adalah: **Dinamis, Kinematis, Statis.**

### 2D collider

Komponen Collider 2D menentukan bentuk GameObject 2D untuk keperluan tabrakan fisik Tabrakan .

Colliders untuk GameObjects 2D semua memiliki nama yang berakhiran "2D". Collider yang tidak memiliki "2D" di namanya dimaksudkan untuk digunakan pada GameObject 3D.

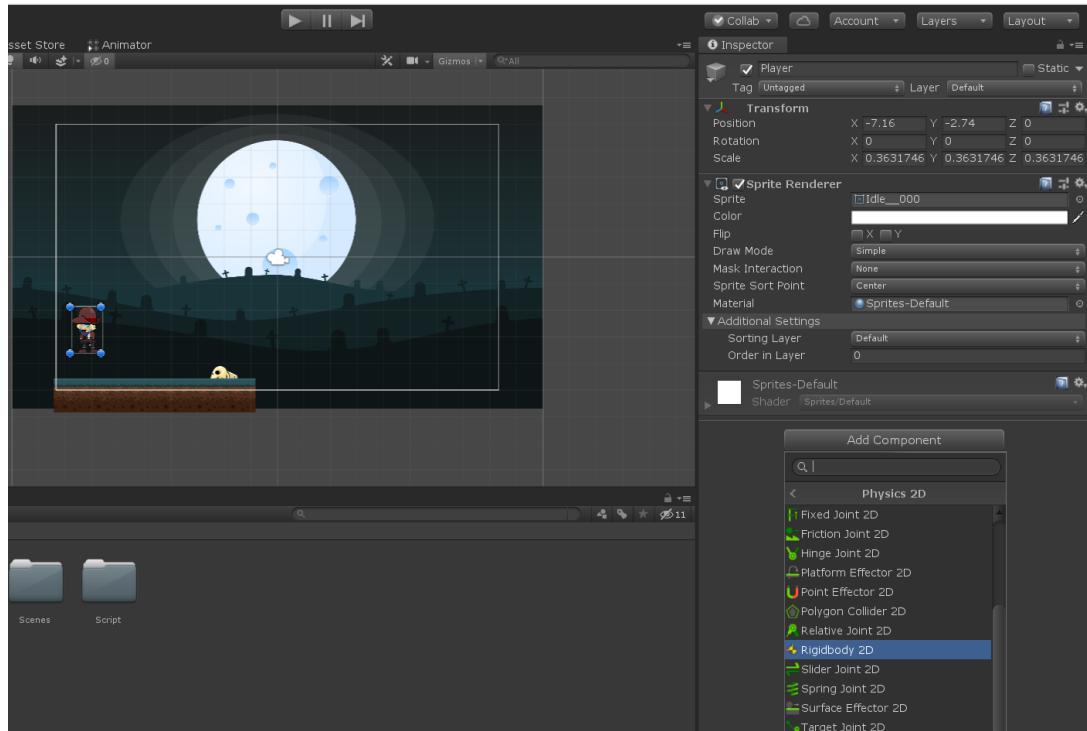
Tipe Collider 2D yang dapat digunakan dengan Rigidbody 2D adalah:

- **Circle Collider 2D** untuk area tumbukan bundar.
- **Box Collider 2D** untuk area tumbukan persegi dan persegi panjang.
- **Polygon Collider 2D** untuk area benturan bentuk bebas.
- **Edge Collider 2D** untuk area tumbukan bentuk unik dan area yang tidak sepenuhnya tertutup (seperti sudut cembung bulat).
- **Capsule Collider 2D** untuk area tumbukan bundar atau berbentuk permen.
- **Composite Collider 2D** untuk menggabungkan Box Collider 2D dan 2D Poligon Collider.

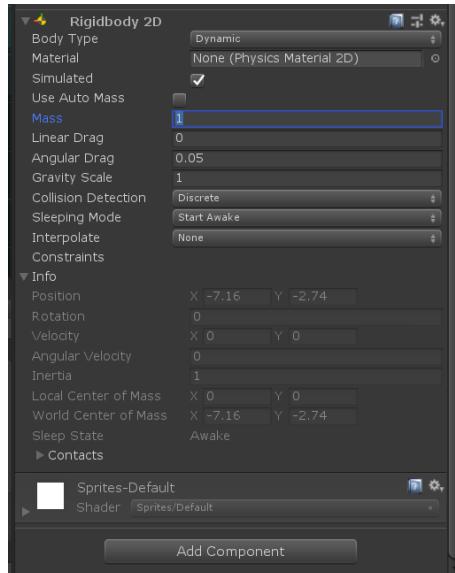
## 2. Praktikum

### a. Memberikan Rigidbody ke Player:

1. Buka Proyek Sebelumnya.
2. Pilih *game object player* → pada bagian *inspector*, klik *add component* → *Physics2D* → *Rigidbody2D*



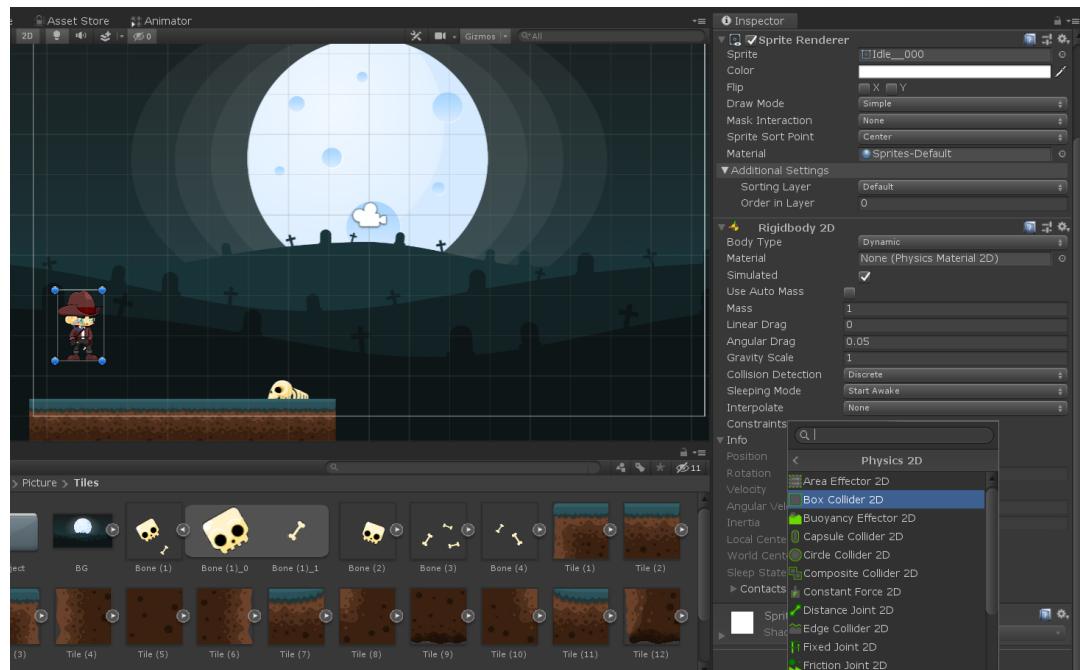
3. Perhatikan inspector dari game object player, terdapat Rigidbody yang dikenai pada game object tersebut. Perhatikan atribut rigidbodynya.



4. Play Game, dan Amati.

#### b. Memberikan Collider ke Player:

- Pilih game object player → pada bagian inspector, klik add component → Physics2D → Box Collider 2D



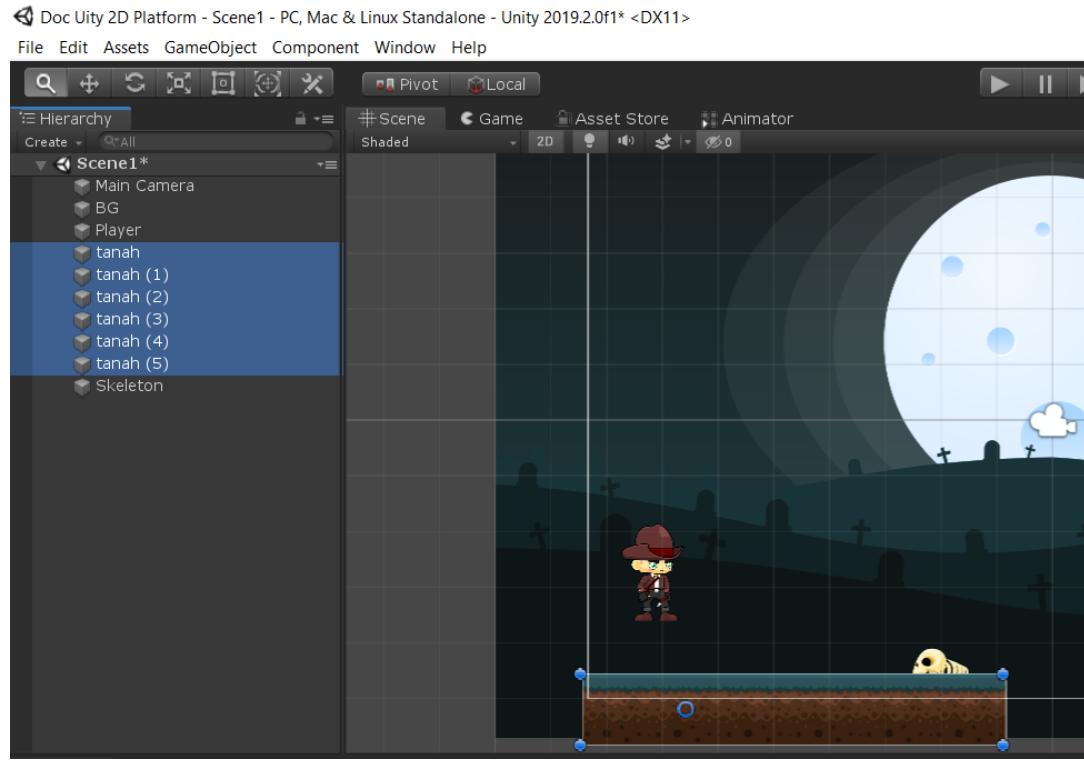
2. Perhatikan inspector dari game object player, terdapat Box Collider 2D yang dikenai pada game object tersebut. Perhatikan atributnya,



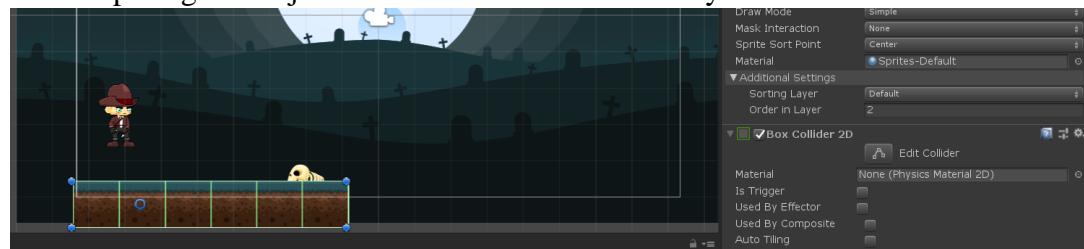
3. Play Game, dan Amati. (Object player masih jatuh dari area game scene).

### c. Memberikan Collider ke objek Tanah:

- Pilih semua *game object tanah* → pada bagian *inspector*, klik *add component* → *Physics2D* → *Box Collider 2D*



2. Perhatikan inspector dari game object player, terdapat Box Collider 2D yang dikenai pada game object tersebut. Perhatikan atributnya.



3. Play Game, dan Amati.

### 3. Latihan Project

Buatlah scene baru dan masukan beberapa asset sebagai awal pembuatan game!.



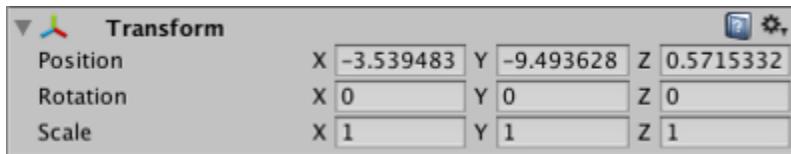
Berikan Rigidbody/Collider dan setting propertinya apabila diperlukan.!

## E. Control Player dan Sensor Tanah (Layer)

### 1. Teori Dasar

#### Transform

Komponen Transform menentukan Posisi, Rotasi, dan Skala masing-masing objek dalam game. Setiap GameObject memiliki Transform.



Property:	Function:
<b>Position</b>	Position of the Transform in X, Y, and Z coordinates.
<b>Rotation</b>	Rotation of the Transform around the X, Y, and Z axes, measured in degrees.
<b>Scale</b>	Scale of the Transform along X, Y, and Z axes. Value “1” is the original size (size at which the object was imported).

Posisi, rotasi, dan nilai skala dari Transform diukur relatif terhadap induk Transform. Jika Transform tidak memiliki induk, properti diukur dalam ruang dunia.

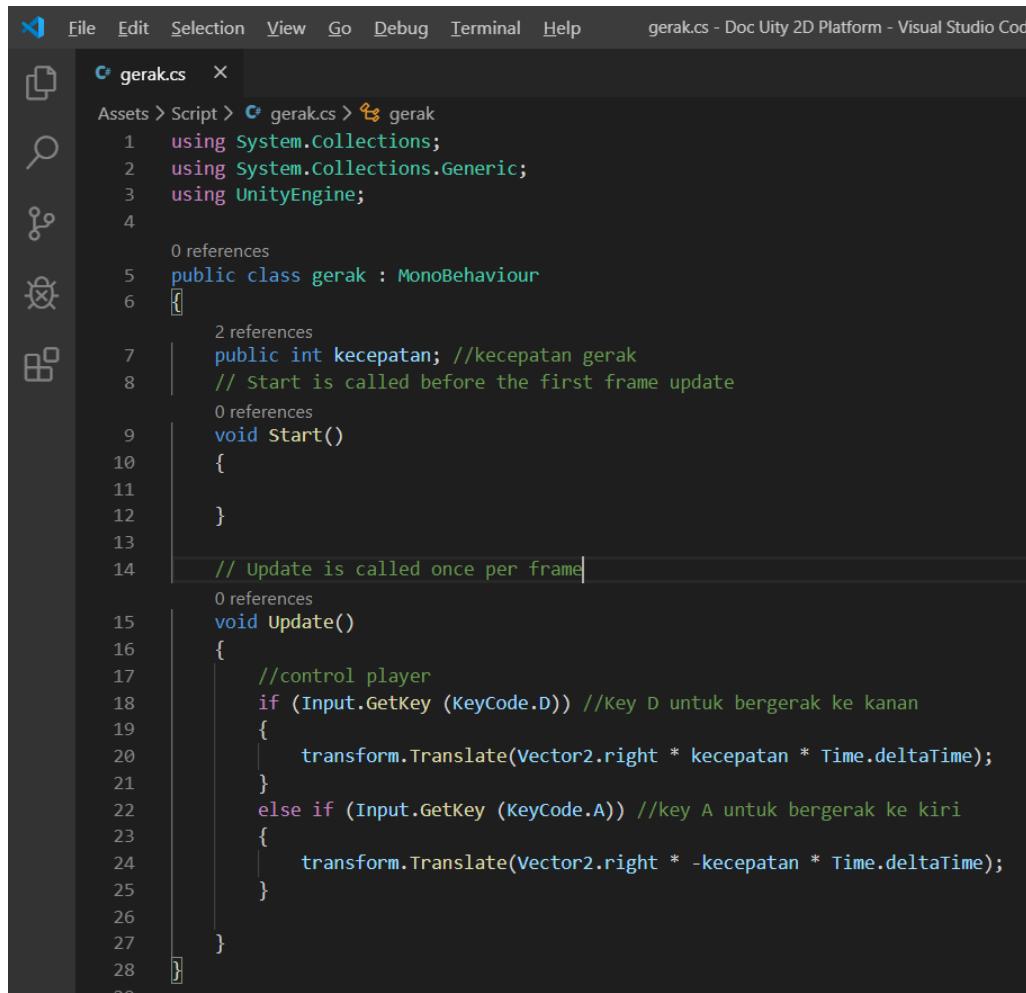
#### Rect Transform

Komponen Rect Transform adalah rekanan tata letak 2D dari komponen Transform. Di mana Transform mewakili satu titik, Rect Transform mewakili persegi panjang yang merupakan UI, elemen dapat ditempatkan di dalam. Jika induk dari Rect Transform juga merupakan Rect Transform, anak Rect Transform juga dapat menentukan bagaimana seharusnya diposisikan dan berukuran relatif terhadap persegi panjang induk.

### 2. Praktikum

#### a. Memberikan Script Control Player (Gerak ke Kiri dan ke Kanan):

1. Buka Proyek Sebelumnya.
2. Buatlah script c# baru dengan nama *gerak.cs* dengan script sebagai berikut :



```

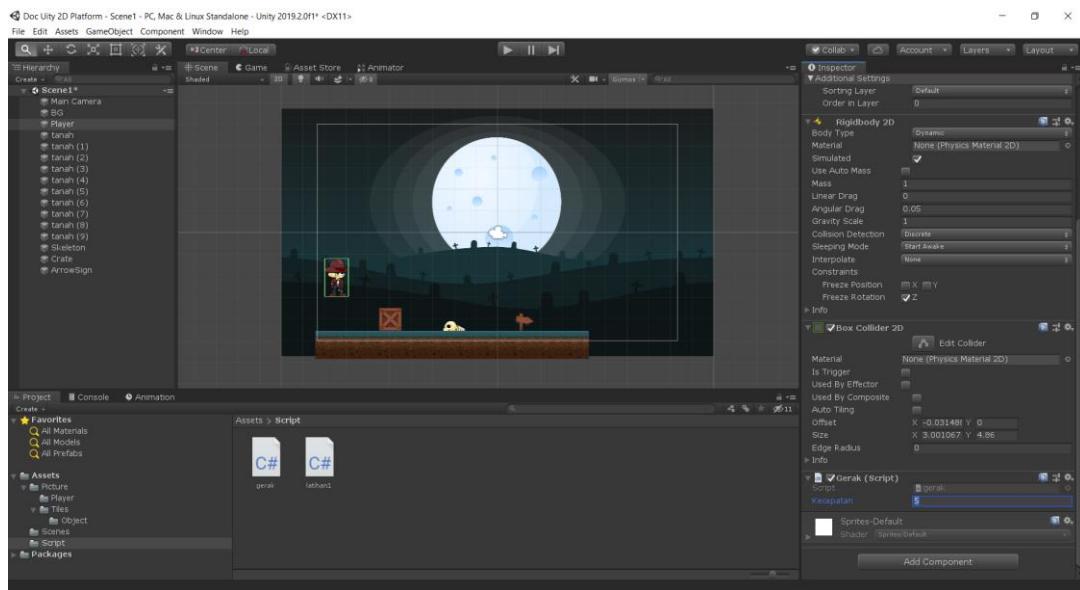
File Edit Selection View Go Debug Terminal Help
gerak.cs - Doc Unity 2D Platform - Visual Studio Code

gerak.cs X
Assets > Script > gerak.cs > gerak
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class gerak : MonoBehaviour
6 {
7     public int kecepatan; //kecepatan gerak
8     // Start is called before the first frame update
9     void Start()
10    {
11    }
12
13    // Update is called once per frame
14    void Update()
15    {
16        //control player
17        if (Input.GetKey(KeyCode.D)) //Key D untuk bergerak ke kanan
18        {
19            transform.Translate(Vector2.right * kecepatan * Time.deltaTime);
20        }
21        else if (Input.GetKey(KeyCode.A)) //key A untuk bergerak ke kiri
22        {
23            transform.Translate(Vector2.right * -kecepatan * Time.deltaTime);
24        }
25    }
26
27 }
28

```

3. Save, dan masukan script tersebut ke game object player.

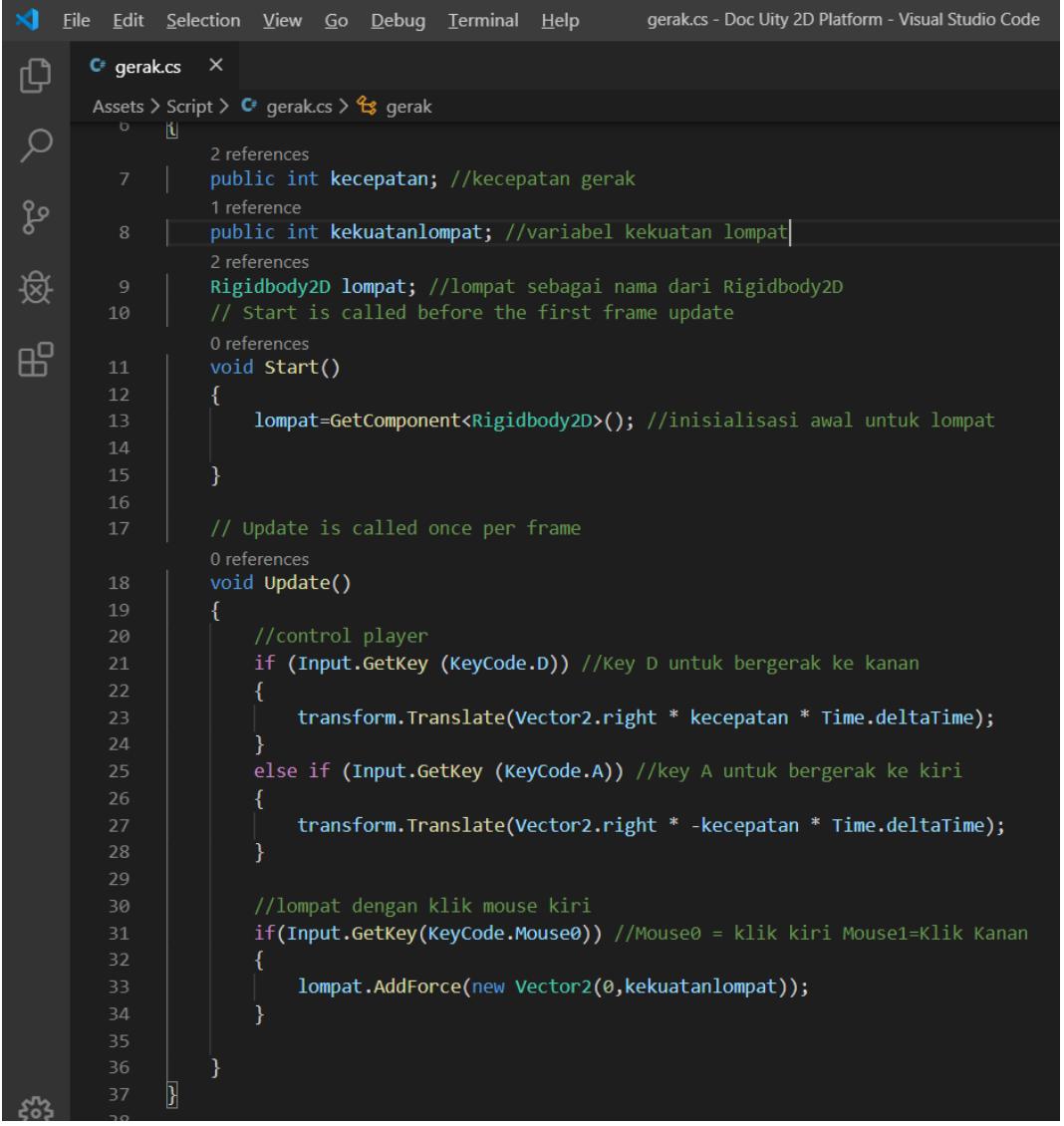
4. Kemudian atur atribut kecepatannya.



5. Play game, lakukan pergerakan ke kiri dan ke kanan dengan tombol *A* dan *D* pada Keyboard. dan amati hasilnya

**b. Memberikan Script Control Player (Melompat):**

1. Buka Script *gerak.cs* sebelumnya.
2. Tambahkan script sebagai berikut :



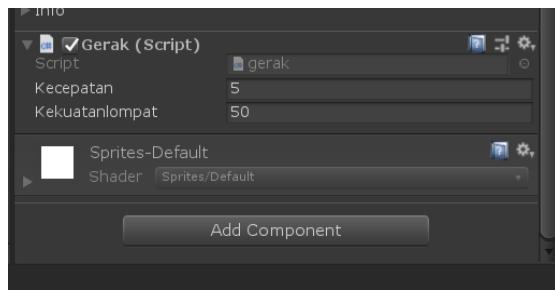
```

File Edit Selection View Go Debug Terminal Help gerak.cs - Doc Unity 2D Platform - Visual Studio Code
gerak.cs
Assets > Script > gerak.cs > gerak
    2 references
    public int kecepatan; //kecepatan gerak
    1 reference
    public int kekuatanlompat; //variabel kekuatan lompat
    2 references
    Rigidbody2D lompat; //lompat sebagai nama dari Rigidbody2D
    // Start is called before the first frame update
    0 references
    void Start()
    {
        lompat=GetComponent<Rigidbody2D>(); //inisialisasi awal untuk lompat
    }
    // Update is called once per frame
    0 references
    void Update()
    {
        //control player
        if (Input.GetKey(KeyCode.D)) //Key D untuk bergerak ke kanan
        {
            transform.Translate(Vector2.right * kecepatan * Time.deltaTime);
        }
        else if (Input.GetKey(KeyCode.A)) //key A untuk bergerak ke kiri
        {
            transform.Translate(Vector2.right * -kecepatan * Time.deltaTime);
        }

        //lompat dengan klik mouse kiri
        if(Input.GetKey(KeyCode.Mouse0)) //Mouse0 = klik kiri Mouse1=klik Kanan
        {
            lompat.AddForce(new Vector2(0,kekuatanlompat));
        }
    }
}

```

3. Save, Kemudian atur nilai kekuatan lompatnya.



4. Play game, lakukan pergerakan ke kiri, kanan dan melompat dengan tombol *A* dan *D* pada Keyboard. Serta klik kiri pada mouse untuk melompat dan amati hasilnya

### c. Memberikan Script Control Player (Flip Objek):

1. Buka Script *gerak.cs* sebelumnya.
2. Tambahkan script sebagai berikut :

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

public class gerak : MonoBehaviour
{
    public int kecepatan; //kecepatan gerak
    public int kekuatanlompat; //variabel kekuatan lompat
    public bool balik;
    public int pindah;
    Rigidbody2D lompat; //lompat sebagai nama dari Rigidbody2D
    // Start is called before the first frame update
    void Start()
    {
        lompat=GetComponent<Rigidbody2D>(); //inisialisasi awal untuk lompat
    }

    // Update is called once per frame
    void Update()
    {
        //control player
        if (Input.GetKeyDown(KeyCode.D)) //Key D untuk bergerak ke kanan
        {
            transform.Translate(Vector2.right * kecepatan * Time.deltaTime);
            pindah=-1;
        }
        else if (Input.GetKeyDown(KeyCode.A)) //key A untuk bergerak ke kiri
        {
            transform.Translate(Vector2.right * -kecepatan * Time.deltaTime);
            pindah=1;
        }

        //lompat dengan klik mouse kiri
        if(Input.GetKeyDown(KeyCode.Mouse0)) //Mouse0 = klik kiri Mouse1=Klik Kanan
        {
            lompat.AddForce(new Vector2(0,kekuatanlompat));
        }
    }
}
```

```

    //logik balik badan
    if(pindah > 0 && !balik)
    {
        flip();
    }
    else if(pindah < 0 && balik)
    {
        flip();
    }

}

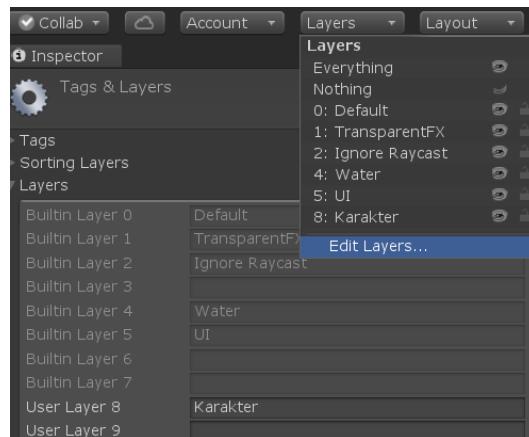
//fungsi balik badan
void flip()
{
    balik = !balik;
    Vector3 Player = transform.localScale;
    Player.x *= -1;
    transform.localScale = Player;
}
}

```

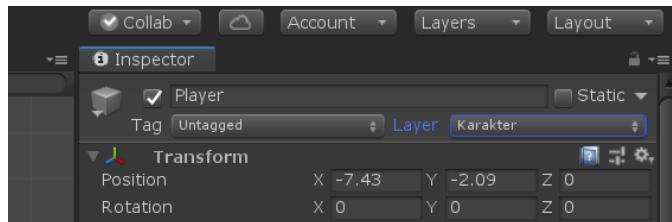
3. Save, Play game, lakukan pergerakan ke kiri, kanan dan melompat dengan tombol *A* dan *D* pada Keyboard. Serta klik kiri pada mouse untuk melompat dan amati hasilnya.
4. Game objek player sudah bisa berbalik badan/flip mengikuti control pergerakan.

#### d. Membuat Deteksi Objek Tanah:

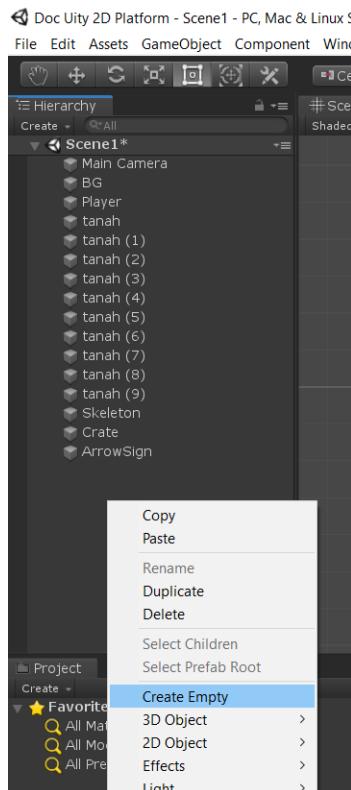
1. Buatlah layer baru dengan nama *karakter*
2. Pada menu layer, pilih →edit layer. Lalu masukan nama layer *Karakter* pada User Layer 8. Perhatikan gambar



3. Setelah membuat layer. Posisikan game object Player di layer Karakter, dengan cara :
4. Pilih Game Object Player → Pilih Layer *Karakter* pada inspector layer.



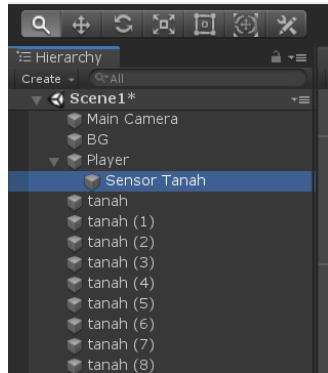
5. Buatlah game object baru di hierarchy, dengan cara klik kanan → create empty-->



6. Lalu beri nama game object tersebut dengan nama *Sensor Tanah*  
 7. Lalu tempatkan posisi Game Object *Sensor Tanah* pada bagian bawah character player.



8. Dan selanjutnya, pada hierarchy, posisi sensor tanah sebagai *Child* dari Player.



9. Tambahkan script sensor tanah sebagai berikut :

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

public class gerak : MonoBehaviour
{
    public int kecepatan; //kecepatan gerak
    public int kekuatanlompat; //variabel kekuatan lompat
    public bool balik;
    public int pindah;
    Rigidbody2D lompat; //lompat sebagai nama dari Rigidbody2D
    //Variabel sensor tanah
    public bool tanah;
    public LayerMask targetlayer;
    public Transform deteksitanah;
    public float jangkauan;
    // Start is called before the first frame update
    void Start()
    {
        lompat=GetComponent<Rigidbody2D>(); //inisialisasi awal untuk lompat
    }

    // Update is called once per frame
    void Update()
    {
        //sensor tanah
        tanah = Physics2D.OverlapCircle(deteksitanah.position, jangkauan, targetlayer);
        //control player
        if (Input.GetKey(KeyCode.D)) //Key D untuk bergerak ke kanan
        {
            transform.Translate(Vector2.right * kecepatan * Time.deltaTime);
            pindah=-1;
        }
        else if (Input.GetKey(KeyCode.A)) //key A untuk bergerak ke kiri
        {
            transform.Translate(Vector2.right * -kecepatan * Time.deltaTime);
            pindah=1;
        }

        //lompat dengan klik mouse kiri
        if(tanah==true && Input.GetKeyDown(KeyCode.Mouse0)) //Mouse0 = klik kiri Mouse1=Klik Kanan
        {
            lompat.AddForce(new Vector2(0,kekuatanlompat));
        }
    }
}
```

```

        }

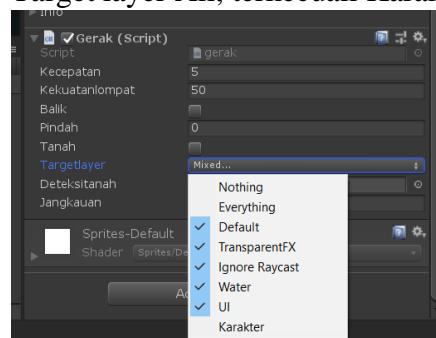
    //logik balik badan
    if(pindah > 0 && !balik)
    {
        flip();
    }
    else if(pindah < 0 && balik)
    {
        flip();
    }
}

//fungsi balik badan
void flip()
{
    balik = !balik;
    Vector3 Player = transform.localScale;
    Player.x *= -1;
    transform.localScale = Player;
}
}

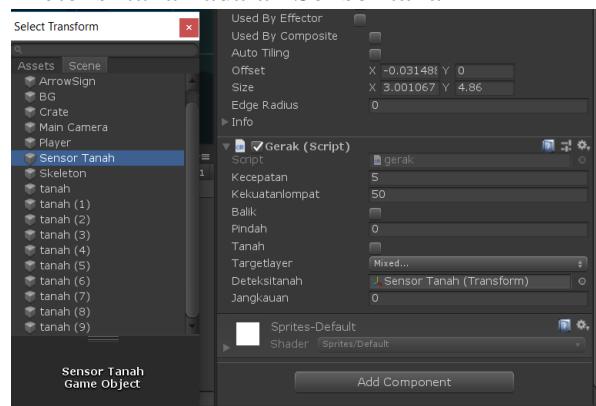
```

10. Lakukan setting nilai sensor tanah, pilih objek Player, perhatikan setting inspector berikut :

- Target layer All, terkecuali Karakter



- Deteksi tanah adalah Sensor tanah



11. Play Game, dan Amati.

### 3. Latihan Project

Lakukan modifikasi nilai, tombol dan control yang lain.

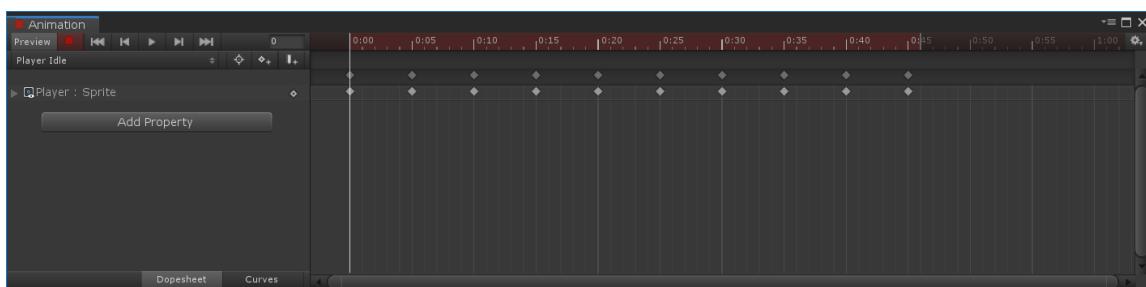
## F. Animasi

### 1. Teori Dasar

#### Animasi

Ada dua cara untuk memberikan animasi pada objek berbentuk 2 Dimensi, yang pertama adalah menggunakan metode skeletal atau bone-based animation dan yang kedua adalah sprite. Pada metode skeletal, kita akan menggerakan tiap objek dari karakter yang ingin dianimasikan frame per frame. Sedangkan untuk metode sprite, animasi dilakukan dengan mengganti gambar untuk tiap frame.

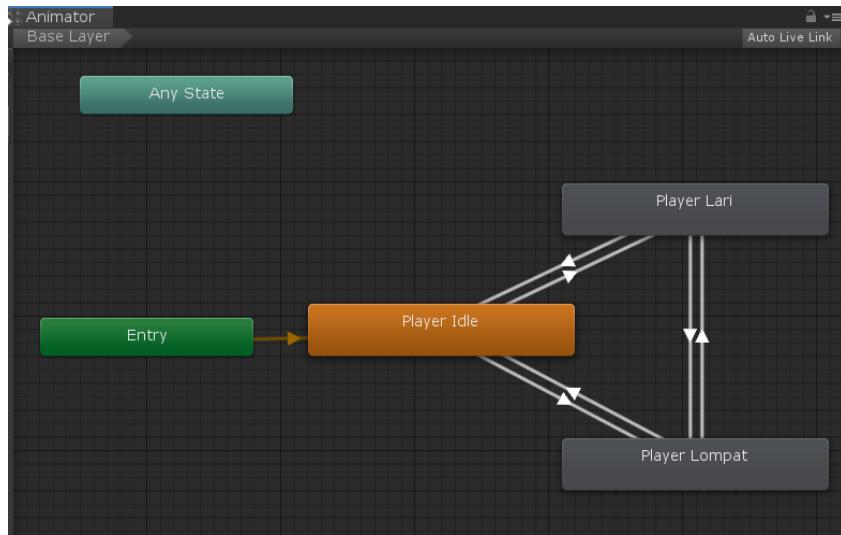
#### Jendela Animasi



**Animation windows** di Unity memungkinkan untuk membuat dan memodifikasi **Animasi Klip** langsung di dalam Unity. Ini dirancang untuk bertindak sebagai alternatif yang kuat dan langsung ke program animasi 3D eksternal. Selain gerakan animasi, editor juga memungkinkan untuk menghidupkan variabel-variabel bahan dan komponen dan menambah Klip Animasi dengan **Event Animasi**, fungsi-fungsi yang dipanggil pada titik-titik yang ditentukan di sepanjang **Timeline**.

#### Animator

**Animator** adalah game **component** yang berfungsi untuk mengendalikan **state** dari objek yang ingin kita animasikan. **State** ini contohnya dari **state** diam, jalan, melompat, dan seterusnya.

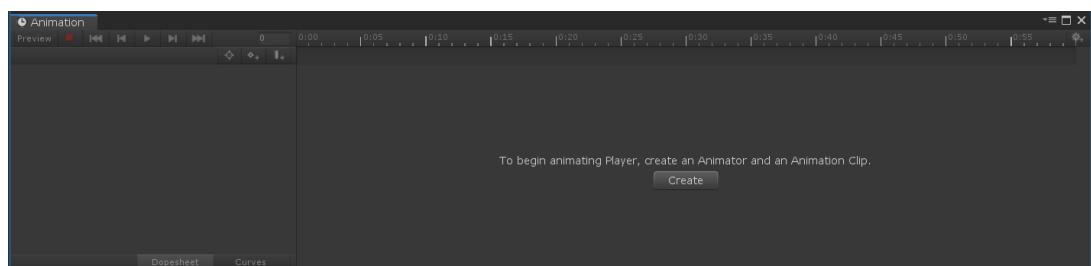
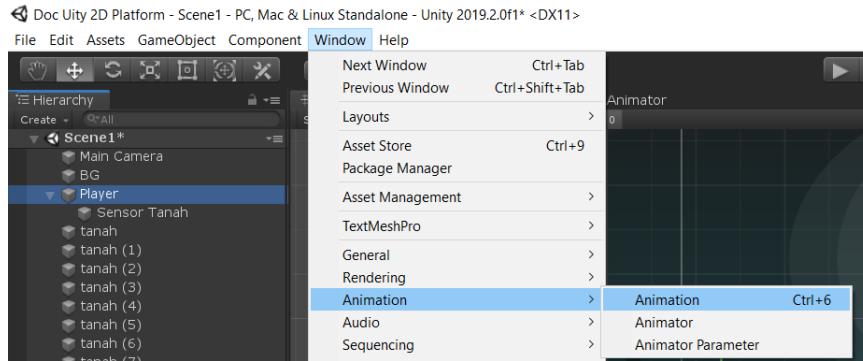


Lebih Detail dari documentasi animasi dan animator dapat diakses di unity documentations : <https://docs.unity3d.com/Manual/AnimationSection.html>

## 2. Praktikum

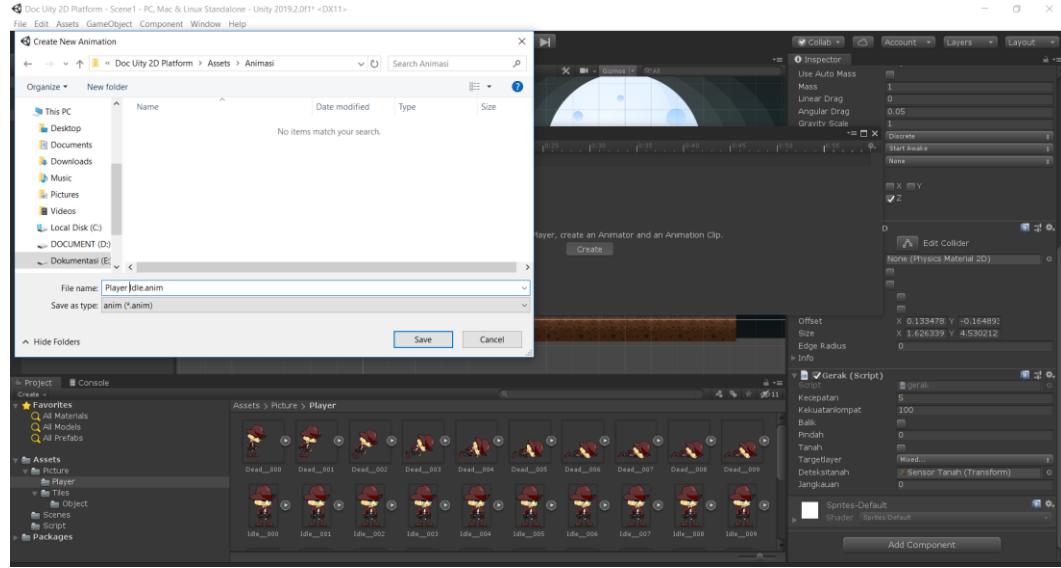
### a. Membuat animasi idle Player:

1. Buka tools animasi, dengan cara klik menu windows → animation.

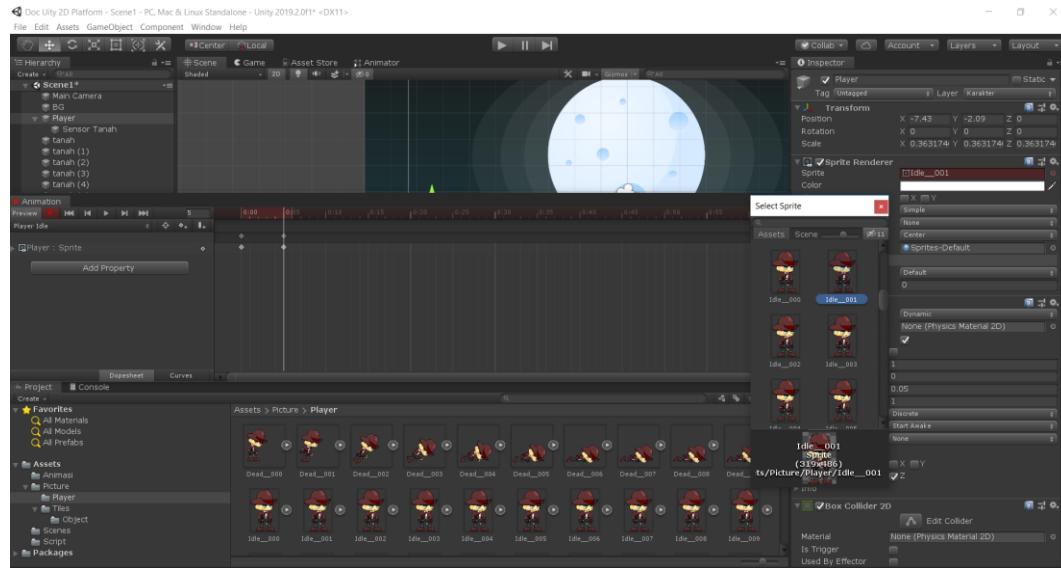


2. Klik create animasi clip untuk membuat animasi, lalu tentukan lokasi penyimpanan.

Berinama *Player Idle*



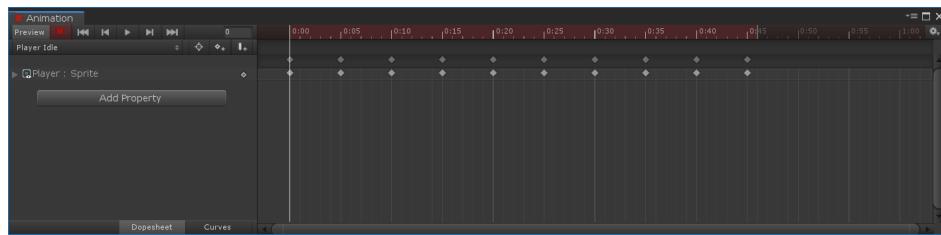
3. Perhatikan timeline, dan inspector layer (Sprite renderer). Untuk animasi ini menggunakan animasi frame by frame dari object sprite yg sudah ada.



4. Untuk detail timeline :

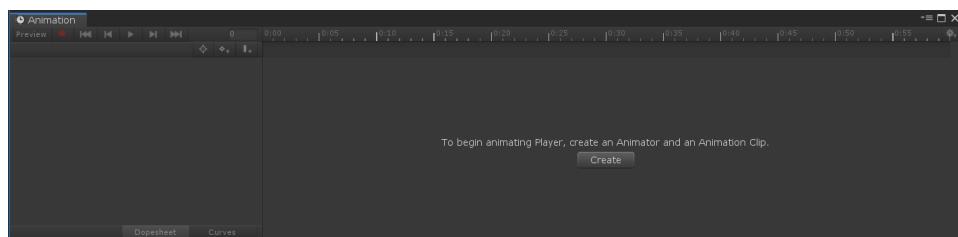
Timeline	Object sprite
05	Idle001
10	Idle002
15	Idle003
20	Idle004
25	Idle005
30	Idle006
35	Idle007
40	Idle008
45	Idle009

Sehingga didapatkan timeline animasi sebagai berikut :



### b. Membuat animasi Jump Player:

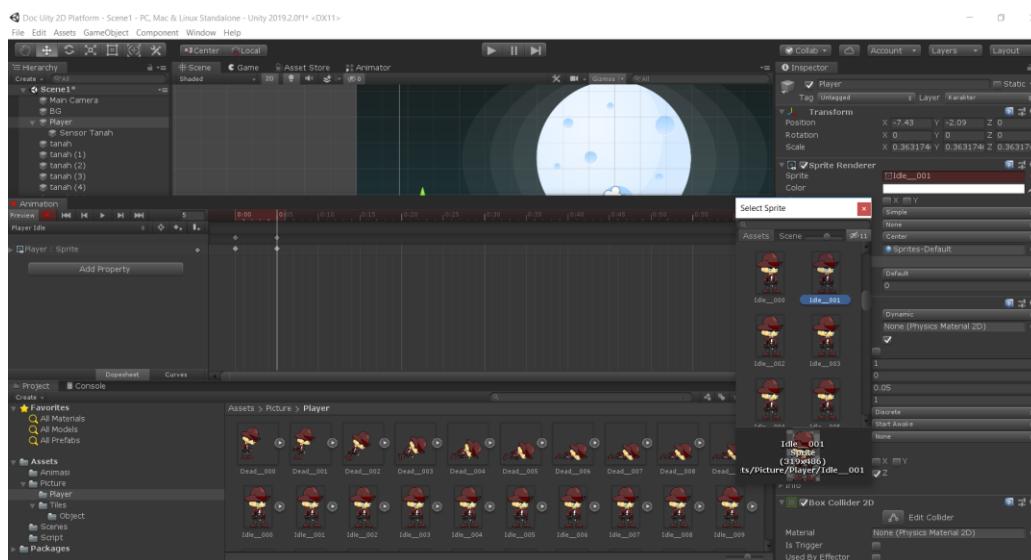
1. Masih di windows animation.



2. Klik create new clip untuk membuat animasi baru, lalu tentukan lokasi penyimpanan. Berinama *Player lompat*



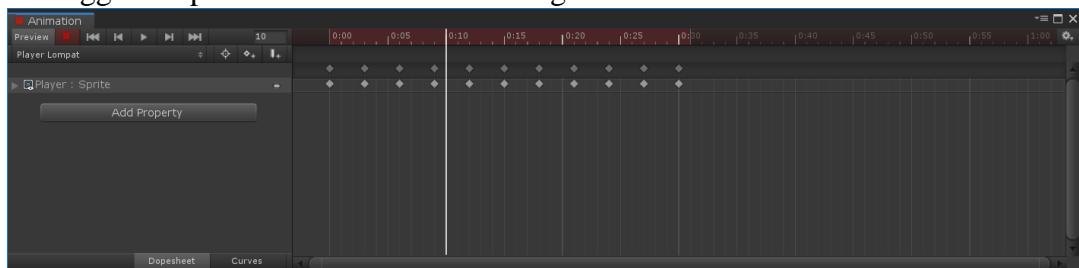
3. Perhatikan timeline, dan inspector layer (Sprite renderer). Untuk animasi ini menggunakan animasi frame by frame dari object sprite yg sudah ada.



4. Untuk detail timeline :

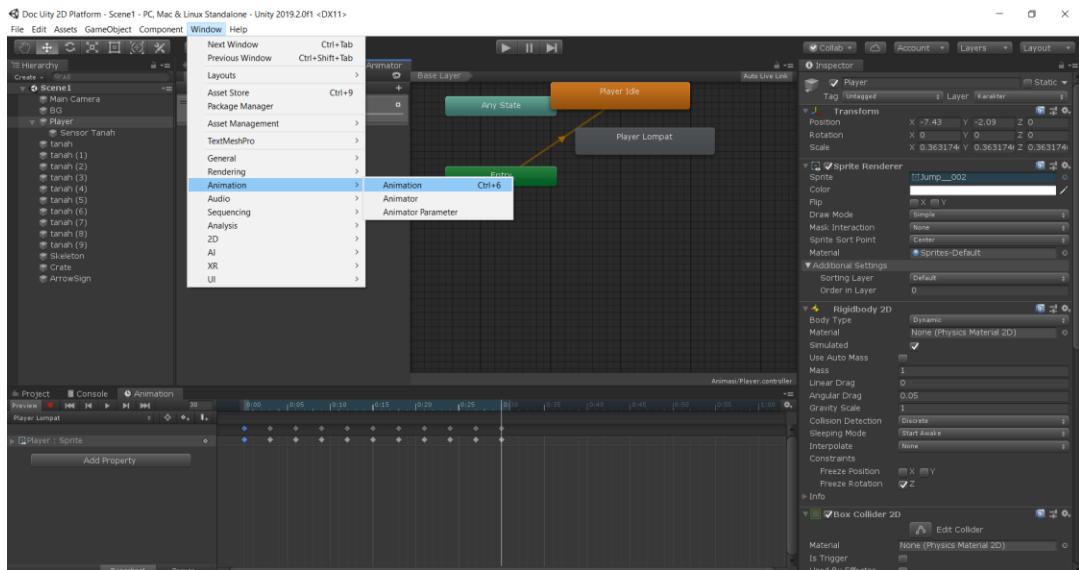
Timeline	Object sprite
03	Jump01
06	Jump02
09	Jump03
12	Jump04
15	Jump05
18	Jump06
21	Jump07
24	Jump08
27	Jump09

Sehingga didapatkan timeline animasi sebagai berikut :

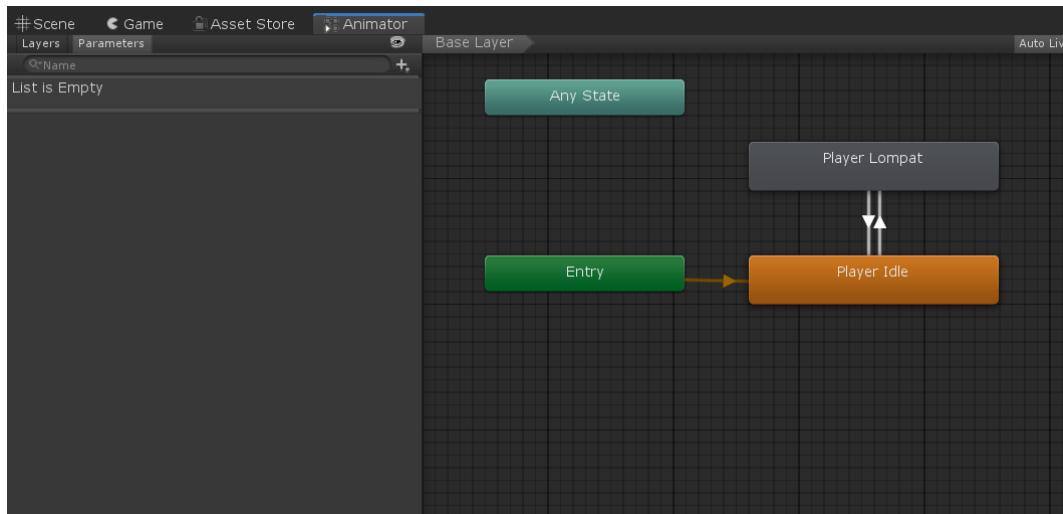


### c. Mengatur transisi animasi Player:

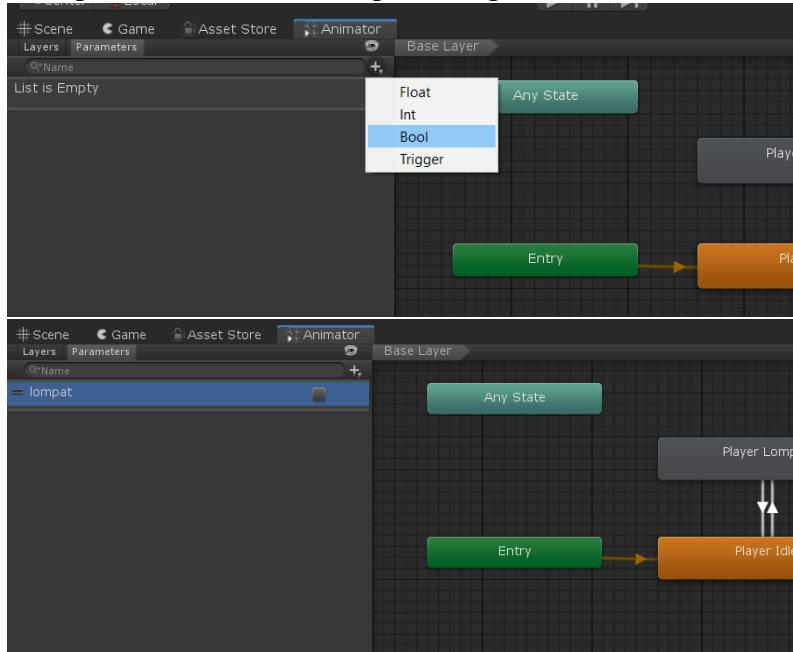
1. Buka windows animator, dengan cara klik menu windows → animation→ animator.



2. Setting transisi animasi dengan gambaran transisi sebagai berikut :

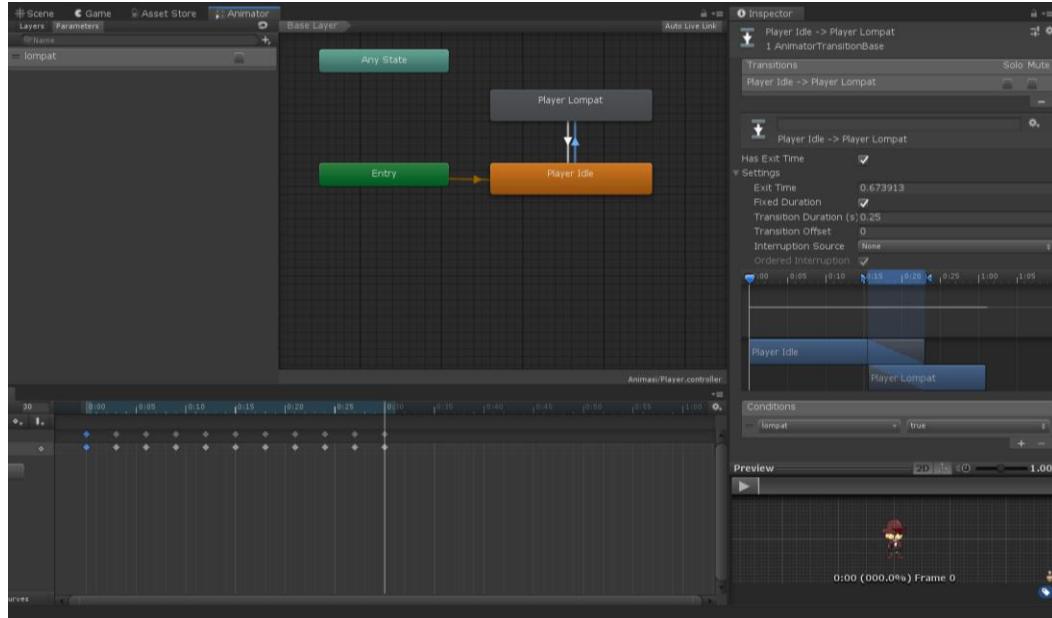


### 3. Buat parameter transisi (lompat) bertipe data bool

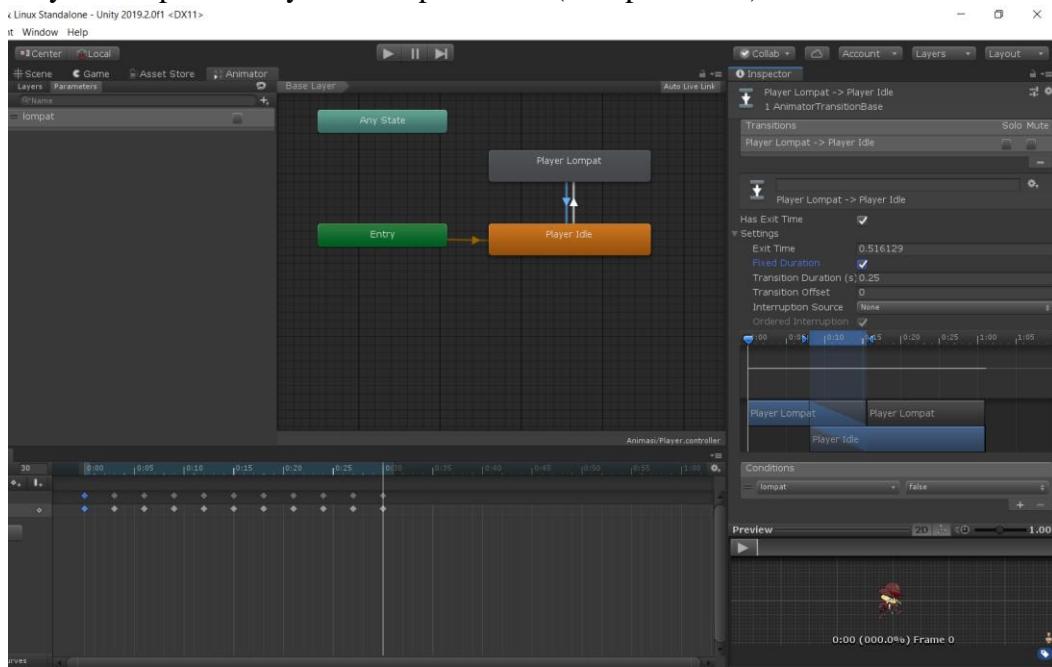


### 4. Setting parameternya

Player Idle → Player Lompat = parameter (Lompat= True)



Player Lompat → Player Idle = parameter (Lompat=False)



#### d. Membuat script logika untuk switching parameter animasi/perpindahan animasi:

1. Buka script gerak, lalu masukan script sebagai berikut :  
Variabel :

```
Animator anim; //sebagai variabel Animator
```

Inisialisasi Animator (Start Behaviour)

```
anim=GetComponent<Animator>(); //Inisialisasi Componen Animasi
```

### Logik Parameter (Void Update)

```
//Logik untuk Animasi
if(tanah == false)
{
    anim.SetBool("lompat", true);
}
else
{
    anim.SetBool("lompat", false);
}
```

2. Script lengkap sebagai berikut ;

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

public class gerak : MonoBehaviour
{
    public int kecepatan; //kecepatan gerak
    public int kekuatanlompat; //variabel kekuatan lompat
    public bool balik;
    public int pindah;
    Rigidbody2D lompat; //lompat sebagai nama dari Rigidbody2D
    //Variabel sensor tanah
    public bool tanah;
    public LayerMask targetlayer;
    public Transform deteksitanah;
    public float jangkauan;
    // Start is called before the first frame update

    //Animasi
    Animator anim; //sebagai variabel Animator
    void Start()
    {
        lompat=GetComponent<Rigidbody2D>(); //inisialisasi awal untuk lompat
        anim=GetComponent<Animator>(); //Inisialisasi Componen Animasi
    }

    // Update is called once per frame
    void Update()
    {
        //Logik untuk Animasi
        if(tanah == false)
        {
            anim.SetBool("lompat", true);
        }
        else
        {
            anim.SetBool("lompat", false);
        }

        //sensor tanah
        tanah = Physics2D.OverlapCircle(deteksitanah.position, jangkauan, targetlayer);
        //control player
    }
}
```

```

if (Input.GetKey (KeyCode.D)) //Key D untuk bergerak ke kanan
{
    transform.Translate(Vector2.right * kecepatan * Time.deltaTime);
    pindah=-1;
}
else if (Input.GetKey (KeyCode.A)) //key A untuk bergerak ke kiri
{
    transform.Translate(Vector2.right * -kecepatan * Time.deltaTime);
    pindah=1;
}

//lompat dengan klik mouse kiri
if(tanah == true && Input.GetKeyDown(KeyCode.Mouse0)) //Mouse0 = klik kiri Mouse1=Klik Kanan
{
    lompat.AddForce(new Vector2(0,kekuatanlompat));
}

//logik balik badan
if(pindah > 0 && !balik)
{
    flip();
}
else if(pindah < 0 && balik)
{
    flip();
}

}

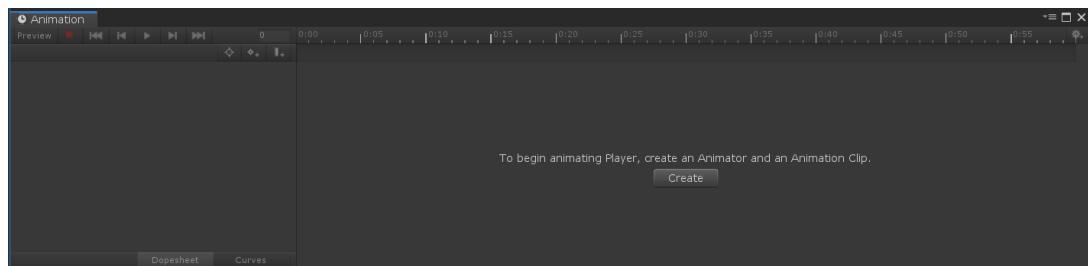
//fungsi balik badan
void flip()
{
    balik = !balik;
    Vector3 Player = transform.localScale;
    Player.x *= -1;
    transform.localScale = Player;
}
}

```

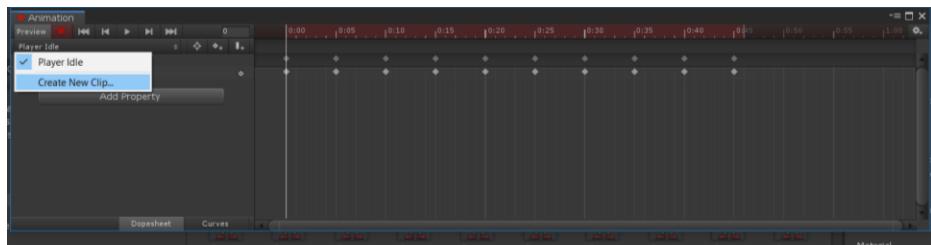
3. Simpan dan Play Game, Amati hasilnya.

#### e. Membuat animasi Lari Player:

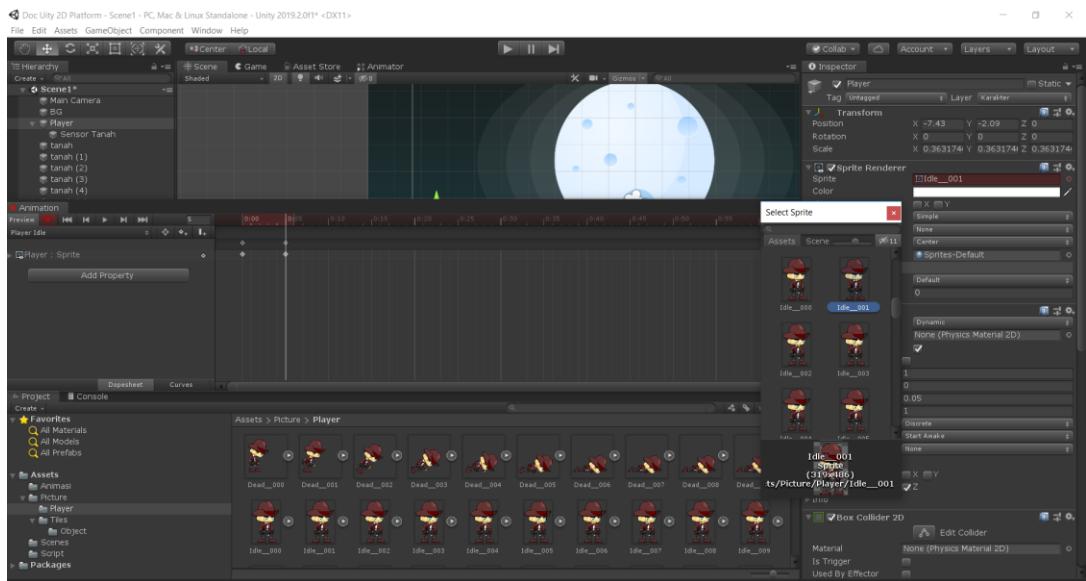
1. Masih di windows animation.



2. Klik create new clip untuk membuat animasi baru, lalu tentukan lokasi penyimpanan. Berinama *Player Lari*



3. Perhatikan timeline, dan inspector layer (Sprite renderer). Untuk animasi ini menggunakan animasi frame by frame dari object sprite yg sudah ada.



4. Untuk detail timeline :

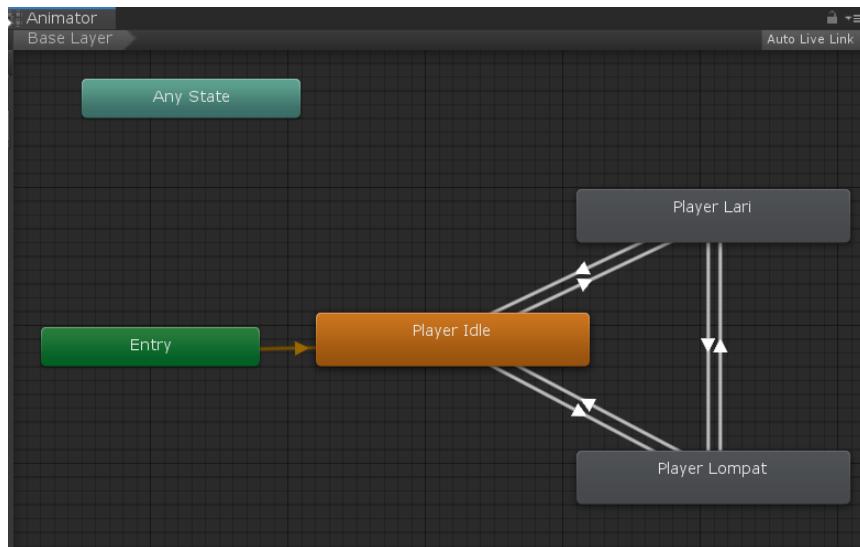
Timeline	Object sprite
00	Run01
03	Run02
06	Run03
09	Run04
12	Run05
15	Run06
18	Run07
21	Run08
24	Run09

Sehingga didapatkan timeline animasi sebagai berikut :



**f. Mengatur transisi animasi Player:**

1. Buka windows animator, dengan cara klik menu windows → animation→ animator.
2. Setting transisi animasi dengan gambaran transisi sebagai berikut :



3. Buat parameter transisi (lari) bertipe data bool

4. Setting parameternya

Player Idle → Player Lompat = parameter (Lompat= True)

Player Lompat → Player Idle = parameter (Lompat=False)

Player Lompat → Player Lari = parameter (Lari=True)

Player Lari → Player Lompat = parameter (Lari=False)

Player Idle → Player Lari = parameter (Lari= True)

Player Lari → Player Idle = parameter (Lari= False)

**g. Membuat script logika untuk switching parameter animasi/perpindahan animasi:**

1. Buka script gerak, lalu masukan script sebagai berikut :

Logik Parameter (Void Update)

Ditempatkan pada logika control tombol A dan D

```
anim.SetBool("lari", true); //aimasi lari
```

Logika saat tidak berlari.

```
anim.SetBool("lari", false);
```

2. Script lengkap sebagai berikut ;

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

public class gerak : MonoBehaviour
{
    public int kecepatan; //kecepatan gerak
    public int kekuatanlompat; //variabel kekuatan lompat
    public bool balik;
    public int pindah;
    Rigidbody2D lompat; //lompat sebagai nama dari Rigidbody2D
    //Variabel sensor tanah
    public bool tanah;
    public LayerMask targetlayer;
    public Transform deteksitanah;
    public float jangkauan;
    // Start is called before the first frame update

    //Animasi
    Animator anim; //sebagai variabel Animator
    void Start()
    {
        lompat=GetComponent<Rigidbody2D>(); //inisialisasi awal untuk lompat
        anim=GetComponent<Animator>(); //Inisialisasi Komponen Animasi
    }

    // Update is called once per frame
    void Update()
    {
        //Logik untuk Animasi
        if(tanah == false)
        {
            anim.SetBool("lompat", true);
        }
        else
        {
            anim.SetBool("lompat", false);
        }

        //sensor tanah
        tanah = Physics2D.OverlapCircle(deteksitanah.position, jangkauan, targetlayer);
        //control player
        if (Input.GetKey (KeyCode.D)) //Key D untuk bergerak ke kanan
        {
            transform.Translate(Vector2.right * kecepatan * Time.deltaTime);
            pindah=-1;
            anim.SetBool("lari", true); //animasi lari
        }
        else if (Input.GetKey (KeyCode.A)) //key A untuk bergerak ke kiri
        {
            transform.Translate(Vector2.right * -kecepatan * Time.deltaTime);
            pindah=1;
            anim.SetBool("lari", true); //animasi lari
        }
    }
}
```

```

    else
    {
        anim.SetBool("lari", false); //Tidak Berlari
    }

    //lompat dengan klik mouse kiri
    if(tanah == true && Input.GetKeyDown(KeyCode.Mouse0)) //Mouse0 = klik kiri Mouse1=Klik Kanan
    {
        lompat.AddForce(new Vector2(0,kekuatanlompat));
    }

    //logik balik badan
    if(pindah > 0 && !balik)
    {
        flip();
    }
    else if(pindah < 0 && balik)
    {
        flip();
    }
}

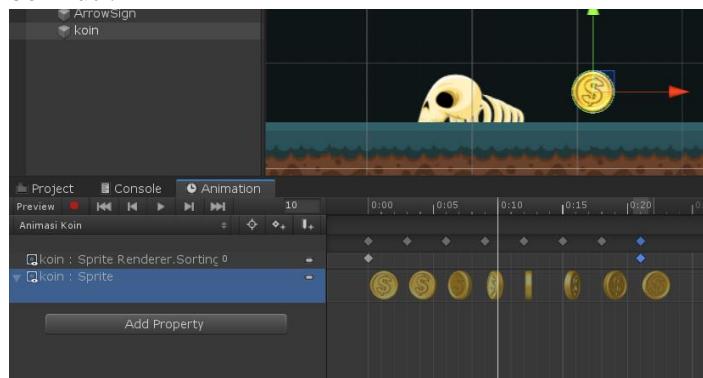
//fungsi balik badan
void flip()
{
    balik = !balik;
    Vector3 Player = transform.localScale;
    Player.x *= -1;
    transform.localScale = Player;
}
}

```

3. Simpan dan Play Game. Amati Hasilnya.

### 3. Latihan Project

1. Buat animasi Slide, Tentukan Transisi dan parameteranya. Slide untuk Tombol Mouse Klik Kanan.
2. Tambahkan objek baru, berupa coin dan buat animasi koin berputar, seperti berikut :



## G. Camera Follow

### 1. Teori Dasar

#### Camera

Adalah perangkat yang menangkap dan menampilkan dunia kepada pemain. Dengan menyesuaikan dan memanipulasi kamera, camera dapat membuat presentasi game benar-benar unik. Setiap game dapat memiliki jumlah kamera yang tidak terbatas dalam sebuah **Scene**. Mereka dapat diatur untuk menyajikan dalam urutan apa pun, di mana saja di layar, atau hanya bagian layar tertentu.

### 2. Praktikum

#### a. Membuat Cammera Follow:

1. Untuk camera follow disini, yaitu camera yang selalu mengikuti pergerakan daripada player.
2. Masih di project yang sama sebelumnya, tambahkan script baru dengan nama *CameraFollow.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class CameraFollow : MonoBehaviour
{
    public float xMargin = 0.5f;
    public float yMargin = 0.5f;
    public float xSmooth = 4f;
    public float ySmooth = 4f;
    public Vector2 maxXAndY;
    public Vector2 minXAndY;
    private Transform player;
    void Awake()
    {
        player=GameObject.FindGameObjectWithTag("Player").transform;
    }
    bool CheckXMargin()
    {
        return Mathf.Abs(transform.position.x - player.position.x) > xMargin;
    }
    bool CheckYMargin()
    {
        return Mathf.Abs(transform.position.y - player.position.y) > yMargin;
    }
    void FixedUpdate()
    {
        TrackPlayer();
    }
    void TrackPlayer()
    {
        float targetX = transform.position.x;
```

```

float targetY = transform.position.y;
if(CheckXMargin())
    targetX=Mathf.Lerp(transform.position.x, player.position.x, xSmooth*time.deltaTime);
if(CheckYMargin())
    targetY=Mathf.Lerp(transform.position.y, player.position.y, ySmooth*time.deltaTime);
targetX=Mathf.Clamp(targetX, minXAndY.x, maxXAndY.x);
targetY=Mathf.Clamp(targetY, minYAndY.y, maxYAndY.y);
transform.position=new Vector3(targetX, targetY, transform.position.z);
}
}

```

3. Simpan, lalu masukan script CameraFollow ke Game Object Main Camera.

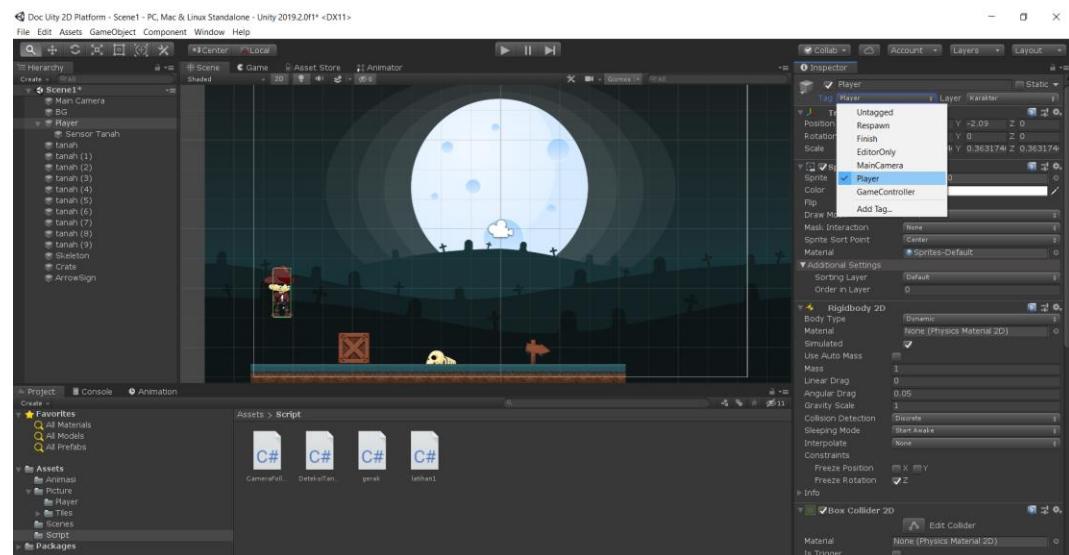
*Pembahasan :*

`player=GameObject.FindGameObjectWithTag("Player").transform;`  
*player adalah game object dengan tag : Player (Penamaan game object harus sama) dan harus di arahkan tag ke Player.*

4. Lakukan setting nilai radius camera follow sebagai berikut :



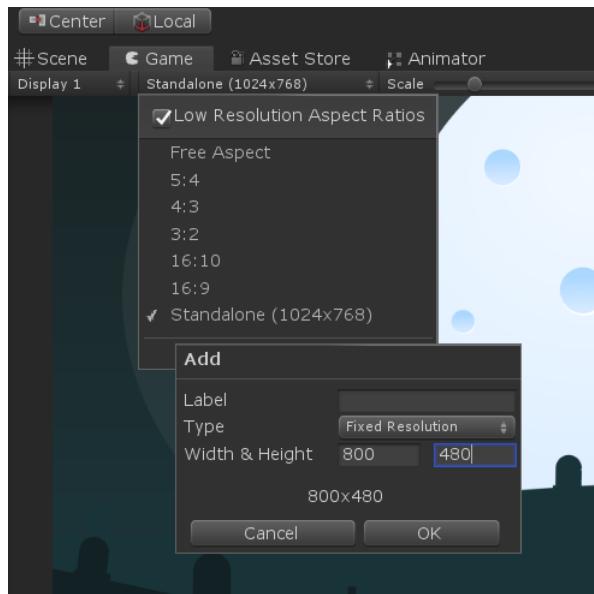
### Setting tag Player pada Game Object Player



5. Play Game, dan Amati hasilnya (Pergerakan camera mengikuti Player)

**b. Setting Display dan Background mengikuti Cammera Follow:**

- Untuk setting display yang diinginkan, masuk ke bagian tab game, dan tambahkan aspect ratio display (contoh : 800 x 480 px).



- Selanjutnya agar background menjadi tetap, tidak mengikuti camera follow, maka background harus ditempatkan menjadi child dari camera.



- Play game, dan amati hasilnya.

### 3. Latihan Project

Tambahkan asset lainnya dan setting camera yang diinginkan, lakukan modifikasi setting camera follow sesuai kebutuhan.

## H. Collider dan Physic

### 1. Teori Dasar

#### 2D collider

Komponen Collider 2D menentukan bentuk GameObject 2D untuk keperluan tabrakan fisik Tabrakan .

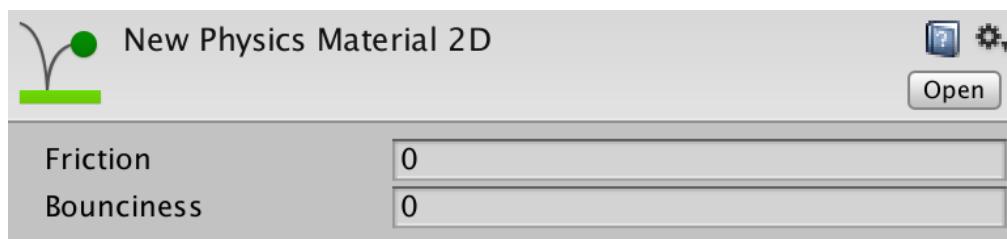
Colliders untuk GameObjects 2D semua memiliki nama yang berakhiran "2D". Collider yang tidak memiliki "2D" di namanya dimaksudkan untuk digunakan pada GameObject 3D.

Tipe Collider 2D yang dapat digunakan dengan Rigidbody 2D adalah:

- **Circle Collider 2D** untuk area tumbukan bundar.
- **Box Collider 2D** untuk area tumbukan persegi dan persegi panjang.
- **Polygon Collider 2D** untuk area benturan bentuk bebas.
- **Edge Collider 2D** untuk area tumbukan bentuk unik dan area yang tidak sepenuhnya tertutup (seperti sudut cembung bulat).
- **Capsule Collider 2D** untuk area tumbukan bundar atau berbentuk permen.
- **Composite Collider 2D** untuk menggabungkan Box Collider 2D dan 2D Poligon Collider.

#### Physic 2D

Sebuah **Physic Material 2D** digunakan untuk mengatur gesekan dan mental yang terjadi antara 2D objek physic ketika mereka bertabrakan. Anda dapat membuat Bahan Physic 2D dari menu Aset ( **Aset> Buat> Bahan Physic 2D** ).



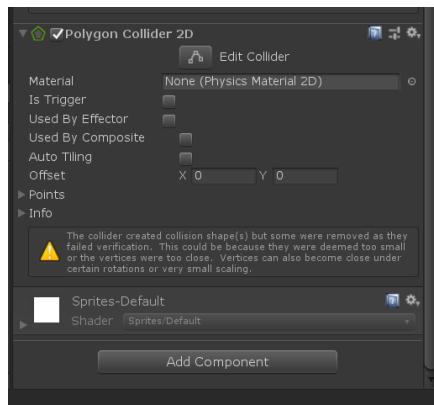
## Properti

<i>Property:</i>	<i>Function:</i>
<b>Friction</b>	Coefficient of friction for this <b>collider</b>
<b>Bounciness</b>	The degree to which <b>collisions</b> rebound from the surface. A value of 0 indicates no bounce while a value of 1 indicates a perfect bounce with no loss of energy.

## 2. Praktikum

### a. Membuat Enemies:

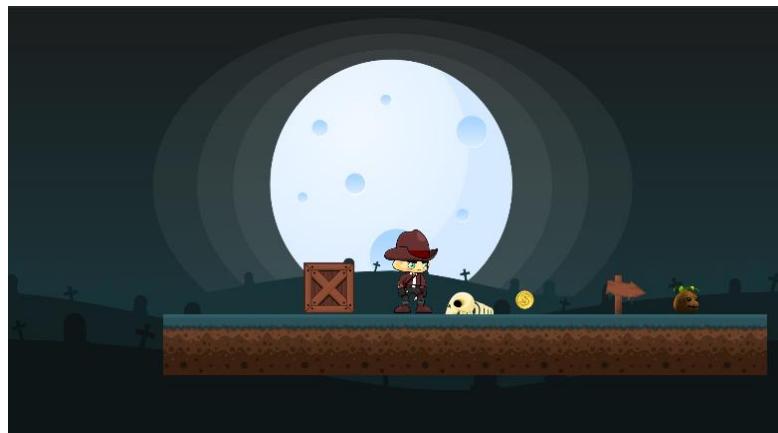
1. Buatlah enemies dilengkapi dengan collider polygon sebagai berikut :



2. Buat juga animasi enemies sebagai berikut :



3. Play game, dan hasilnya seperti berikut:



4. Sudah ada object Player, Koin, dan Enemies.

#### b. Membuat Logic untuk enemies (Output Console)

1. Untuk logic yang akan dibuat adalah, ketika player menyentuh enemies maka akan ada peringatan,
2. Script program sebagai berikut berinama *enemies.cs* :

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

public class Enemies : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

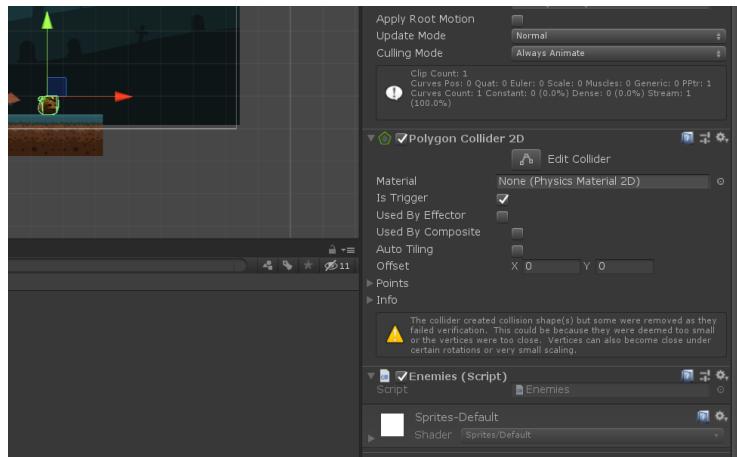
    }

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

    }

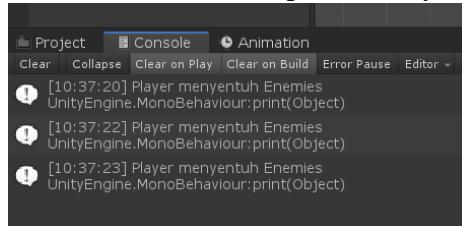
    //Fungsi ketika menyentuh enemies
    void OnTriggerEnter2D(Collider2D other) {
        if(other.transform.tag == "Player")
        {
            print ("Player menyentuh Enemies");
        }
    }
}
```

3. Masukan script tersebut ke game object Enemies



Enemies memiliki Polygon Collider dan Triger.

- Play Game, amati hasilnya. (ketika player menyentuh enemies, maka di bagian console akan muncul pesan “Player menyentuh Enemies”.



### c. Membuat Logic untuk enemies (Enemies Destroyed)

- Untuk logic yang akan dibuat adalah, ketika player menyentuh enemies maka Enemies Destroyed,
- Adapun script program *enemies.cs* sebagai berikut :

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

public class Enemies : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

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

    }

    //Fungsi ketika menyentuh enemies
```

```

void OnTriggerEnter2D(Collider2D other) {
    if(other.transform.tag == "Player")
    {
        Destroy(gameObject);
    }
}

```

3. Play Game, amati hasilnya. (*ketika player menyentuh enemies, maka Enemies akan hilang*).

#### d. Menghubungkan Script dan Komponen Game Object Player (Logic Player Destroyed)

1. Untuk logic yang akan dibuat adalah, ketika player menyentuh enemies maka Player Destroyed, tetapi sebelumnya akan dihubungkan terlebih dahulu script dan game object Player (Script *gerak.cs*) yang dipanggil di Script *Enemies.cs*.

```

gerak KomponenGerak; // Variabel gerak dan Semua komponennya di Script Gerak
// Start is called before the first frame update
void Start()
{
    KomponenGerak = GameObject.Find("Player").GetComponent<gerak>();
}

```

2. Adapun script *enemies.cs* adalah sebagai berikut :

```

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

public class Enemies : MonoBehaviour
{
    gerak KomponenGerak; // Variabel gerak dan Semua komponennya di Script Gerak
    // Start is called before the first frame update
    void Start()
    {
        KomponenGerak = GameObject.Find("Player").GetComponent<gerak>();
    }

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

    }

    //Fungsi ketika menyentuh enemies
    void OnTriggerEnter2D(Collider2D other) {
        if(other.transform.tag == "Player")
    }
}

```

```
        {
            Destroy(KomponenGerak.gameObject);
        }
    }
```

3. Play Game, amati hasilnya. (*ketika player menyentuh enemies, maka Player akan hilang*).

e. Membuat Variabel Heart untuk Player (Logic Heart akan berkurang ketika menyentuh Enemies)

1. Buka Script gerak.cs (Buatlah variable heart dengan type data int).

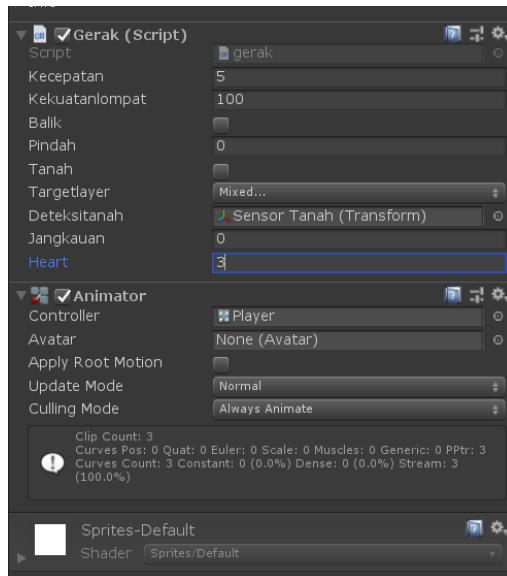
***Kenapa di Script gerak?*** Karena agar atribut heart ada di inspector player.

Dikarenakan script *gerak.cs* dikenai pada objek Player.

```
public int heart; // Variabel nyawa Player
```

Save Script *gerak.cs*, pilih game object player, perhatikan pada inspectornya.

Terdapat atribut variable *Heart* pada inspector. Isikan dengan nilai 3



2. Logic ketika heart kurang dari 0, maka player destroy. Tambahkan script logic dibawah ini pada lifecycle update pada script *gerak.cs*.

```
//Logik Heart  
    if(Heart<0)  
    {
```

```

        Destroy(gameObject);
    }
}
```

3. Save, Untuk script *enemies.cs* scriptnya sebagai berikut :

```

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

public class Enemies : MonoBehaviour
{
    gerak KomponenGerak; // Variabel gerak dan Semua komponennya di Script Gerak
    // Start is called before the first frame update
    void Start()
    {
        KomponenGerak = GameObject.Find("Player").GetComponent<gerak>();
    }

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

    }

    //Fungsi ketika menyentuh enemies
    void OnTriggerEnter2D(Collider2D other) {
        if(other.transform.tag == "Player")
        {
            KomponenGerak.Heart--;
        }
    }
}
```

4. Save dan Play game, lalu amati hasilnya. Seharusnya ketika di play maka akan berjalan logic seperti dibawah ini :

- Ketika player menyentuh enemies, maka heart akan berkurang 1 dan seterusnya (*Decrement*)
- Ketika Heart Player 0, maka Player Destroy

## f. Membuat Variabel Koin (Logic Koint akan Bertambah ketika menyentuh Game Object Koin)

1. Buka Script *gerak.cs* (Buatlah variable koin dengan type data int).
2. Buat Script baru dengan nama *koin.cs*, lalu masukan kode program berikut :

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

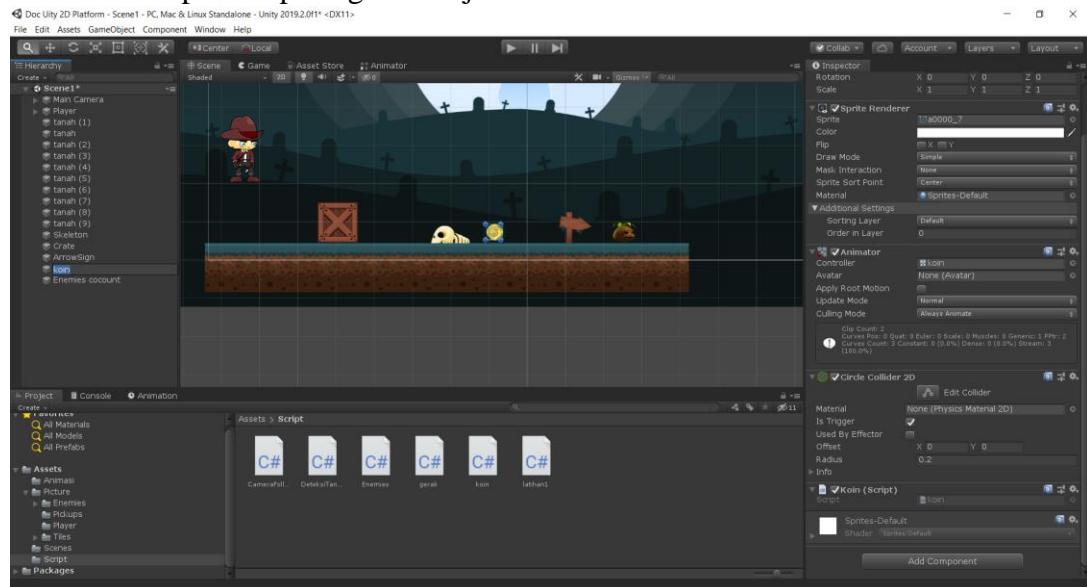
public class koin : MonoBehaviour
{
    gerak KomponenGerak; // Variabel gerak dan Semua komponennya di Script Gerak
    // Start is called before the first frame update
    void Start()
    {
        KomponenGerak = GameObject.Find("Player").GetComponent<gerak>();
    }

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

    }

    //Fungsi ketika menyentuh enemies
    void OnTriggerEnter2D(Collider2D other) {
        if(other.transform.tag == "Player")
        {
            KomponenGerak.Koin++;
            Destroy(gameObject);
        }
    }
}
```

3. Masukan script koin pada game Object Koin.

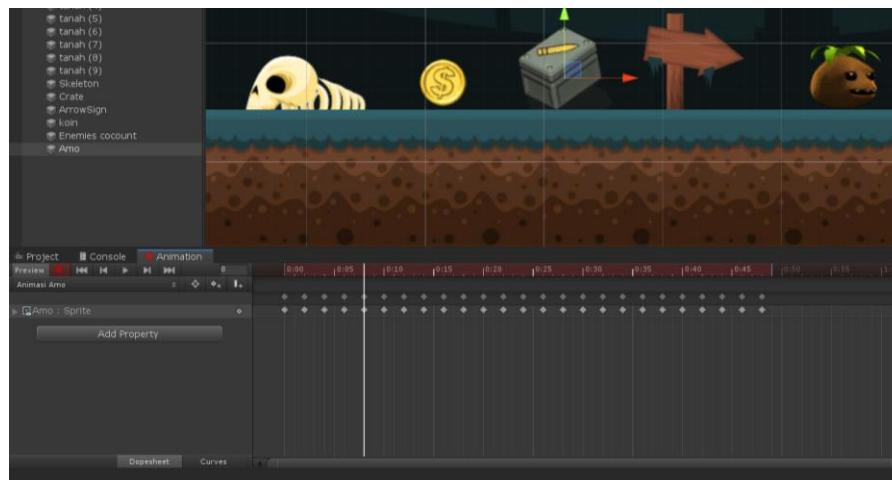


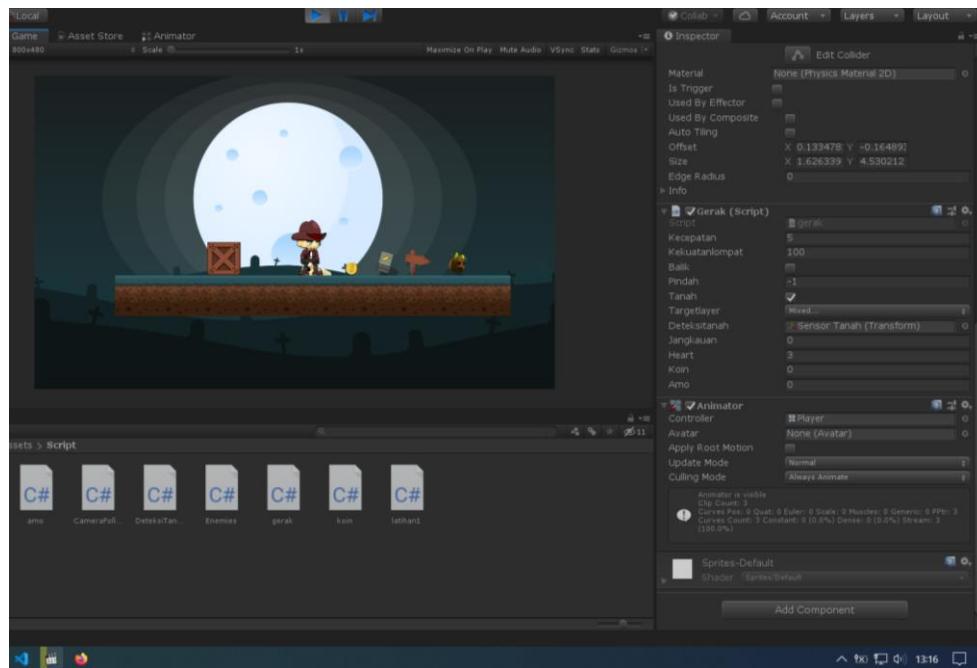
4. Play Game, dan Amati hasilnya pada inspector player Atribut Heart dan Nyawa.



### 3. Latihan Project

Tambahkan game object lain, misalnya Amo yang memiliki nilai variable Amo, yang nantinya bisa digunakan untuk menembak. Setiap kali mendapatkan object Amo, maka Variabel Amo bertambah 10.





## I. Transform/Check Point

### 1. Teori Dasar

Transformasi digunakan untuk perpindahan object game pada suatu tempat tertentu yang sudah di tentukan. Misalnya pada game 2d ini, perpindahan dimungkinkan dengan cara mensetting vector 2D sebagai posisi dari game objek.

Contohnya ketika player menyentuh enemies, maka player dikembalikan ke posisi awal object player memulai game.

Transformasi biasanya digunakan juga sebagai check point dalam pembangunan game, untuk membuat pemain menjadi lebih menarik dalam menyelesaikan misi dari permainan tersebut.

### 2. Praktikum

#### a. Membuat Transformasi Player ketika Menyentuh Enemies

1. Buka Script *gerak.cs* (Buatlah variable Vector untuk Posisi awal player dan Variabel play\_again bertipe data Boolean untuk nilai kembali).

```
Vector2 play; //variabel vector untuk posisi start
public bool play_again; //Variabel nilai play kembali
```

2. Inisialisasikan play pada lifecycle start dengan script sebagai berikut :

```
play=transform.position; //start sebagai object transform posisi
```

3. Buatlah kondisi pada lifecycle update untuk kondisi transformasi player ke posisi awal. Scriptnya sebagai berikut :

```
//Logic Cek Point 1
if(play_again == true)
{
    transform.position = play;
    play_again=false;
}
```

4. Buka script *enemies.cs* lalu tambahkan script pada fungsi on trigger untuk menset nilai tru pada variabel play\_again.

```
KomponenGerak.play_again=true; //player kembali start dari awal
```

5. Untuk Script lengkap *gerak.cs* dan *enemies.cs* adalah sebagai berikut :
   
*Gerak.cs*

```

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

public class gerak : MonoBehaviour
{
    public int kecepatan; //kecepatan gerak
    public int kekuatanLompat; //variabel kekuatan Lompat
    public bool balik;
    public int pindah;
    Rigidbody2D Lompat; //Lompat sebagai nama dari Rigidbody2D
    //Variabel sensor tanah
    public bool tanah;
    public LayerMask targetLayer;
    public Transform deteksitanah;
    public float jangkauan;

    public int Heart; // Variabel nyawa Player
    public int Koin; //Variabel Koin
    public int Amo; //Variabel Amo

    // Start is called before the first frame update
    //Animasi
    Animator anim; //sebagai variabel Animator

    Vector2 play; //variabel vector untuk posisi start
    public bool play_again; //Variabel nilai play kembali
    void Start()
    {
        Lompat=GetComponent<Rigidbody2D>(); //inisialisasi awal untuk Lompat
        anim=GetComponent<Animator>(); //Inisialisasi Componen Animasi
        play=transform.position; //start sebagai object transform posisi
    }

    // Update is called once per frame
    void Update()
    {
        //Logic Cek Point 1
        if(play_again == true)
        {
            transform.position = play;
            play_again=false;
        }
        //Logik Heart
        if(Heart<0)
        {
            Destroy(gameObject);
        }

        //Logik untuk Animasi
        if(tanah == false)
        {
            anim.SetBool("Lompat", true);
        }
        else
        {
            anim.SetBool("Lompat", false);
        }

        //sensor tanah
        tanah = Physics2D.OverlapCircle(deteksitanah.position, jangkauan, targetLayer);
    }
}

```

```

//control player
if (Input.GetKey (KeyCode.D)) //Key D untuk bergerak ke kanan
{
    transform.Translate(Vector2.right * kecepatan * Time.deltaTime);
    pindah=-1;
    anim.SetBool("lari", true); //animasi Lari
}
else if (Input.GetKey (KeyCode.A)) //key A untuk bergerak ke kiri
{
    transform.Translate(Vector2.right * -kecepatan * Time.deltaTime);
    pindah=1;
    anim.SetBool("lari", true); //animasi Lari
}
else
{
    anim.SetBool("lari", false); //Tidak Berlari
}

//Lompat dengan klik mouse kiri
if(tanah == true && Input.GetKeyDown(KeyCode.Mouse0)) //Mouse0 = klik kiri Mouse1=Klik Kanan
{
    Lompat.AddForce(new Vector2(0,kekuatanLompat));
}

//Logik balik badan
if(pindah > 0 && !balik)
{
    flip();
}
else if(pindah < 0 && balik)
{
    flip();
}

}

//fungsi balik badan
void flip()
{
    balik = !balik;
    Vector3 Player = transform.localScale;
    Player.x *= -1;
    transform.localScale = Player;
}
}

```

### Enemies.cs

```

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

public class Enemies : MonoBehaviour
{
    gerak KomponenGerak; // Variabel gerak dan Semua komponennya di Script Gerak
    // Start is called before the first frame update
}

```

```

void Start()
{
    KomponenGerak = GameObject.Find("Player").GetComponent<gerak>();
}

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

//Fungsi ketika menyentuh enemies
void OnTriggerEnter2D(Collider2D other) {
    if(other.transform.tag == "Player")
    {
        KomponenGerak.Heart--; //Mengurangi nilai Heart -1 dst
        KomponenGerak.play_again=true; //player kembali start dari awal
    }
}

```

6. Play game, dan amati hasilnya. Adapun kondisi yang terjadi adalah :

- Player ketika menyentuh enemies akan kembali ke posisi awal
- Dan ketika heart sama dengan 0, maka player akan destroy.

### 3. Latihan Project

Tambahkan Enemies yang lain dengan pengurangan heart yang berbeda.

## J. User Interface

### 1. Teori Dasar

#### User Interface

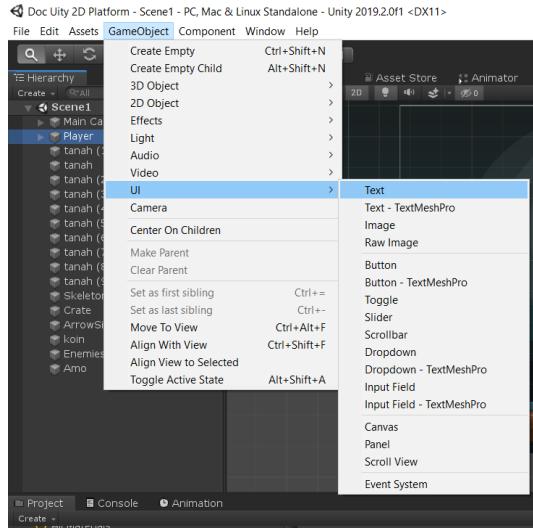
Unity menyediakan toolkit antarmuka pengguna (UI) berikut untuk membuat UI baik di Unity Editor atau di aplikasi:

- **UIElements** : Elemen Antarmuka Pengguna (UIElements) adalah toolkit UI mode dipertahankan untuk mengembangkan antarmuka pengguna di Unity Editor. UIElements didasarkan pada teknologi web yang diakui dan mendukung stylesheet, penanganan acara dinamis dan kontekstual, dan kegigihan data.
- **Unity UI** : Unity User Interface (Unity UI) adalah toolkit UI sederhana untuk mengembangkan antarmuka pengguna untuk game dan aplikasi. Unity UI adalah sistem UI berbasis GameObject yang menggunakan komponen dan Tampilan Game untuk mengatur, memosisikan, dan menata antarmuka pengguna. Anda tidak dapat menggunakan Unity UI untuk antarmuka pengguna dalam Unity Editor.
- **IMGUI** : Immediate Mode Graphical User Interface adalah toolkit UI berbasis kode yang terutama dimaksudkan sebagai alat untuk pengembang. IMGUI menggunakan `OnGUI` fungsi (dan skrip yang mengimplementasikan fungsi `OnGUI`) untuk menggambar dan mengelola antarmuka penggunanya. Anda dapat menggunakan IMGUI untuk membuat tampilan debugging dalam game, khusus **Inspektor** untuk komponen skrip, dan jendela atau alat yang memperluas Unity Editor. Ini bukan opsi terbaik untuk membangun UI game atau aplikasi Anda.

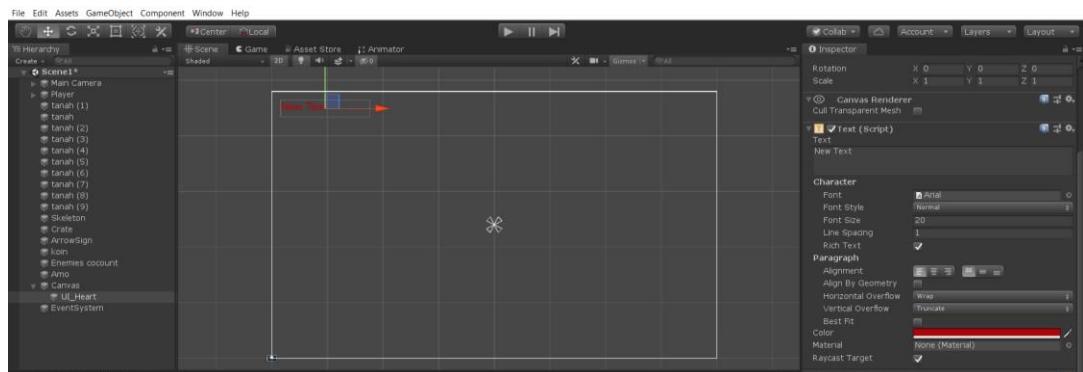
### 2. Praktikum

#### a. Membuat UI untuk Heart

1. Buat game object baru untuk User Interface,
2. Pilih menu Game Object → UI → Text.



3. Berinama Game Object UI tersebut : UI\_Heart
4. Perhatikan pada bagian inspector, lakukan setting sesuai dengan kebutuhan, (Jenis Text, Ukuran text, warna, dll)



5. Setting juga untuk penempatan UI\_Heart.
6. Pada pembahasan sebelumnya terkait variabel heart, sudah ada pada game object Player. Sekarang menyiapkan script coding untuk menampilkan nilai variabel Heart pada UI\_Heart.
7. Buka script gerak.cs, karena akan menambahkan komponen UI, maka tambahkan library UI yaitu :

```
using UnityEngine.UI;
```

8. Selanjutnya, tambahkan script untuk menampilkan nilai Herat ke game object UI\_heart sebagai berikut :

```
Text info_Heart; // Variabel Heart
```

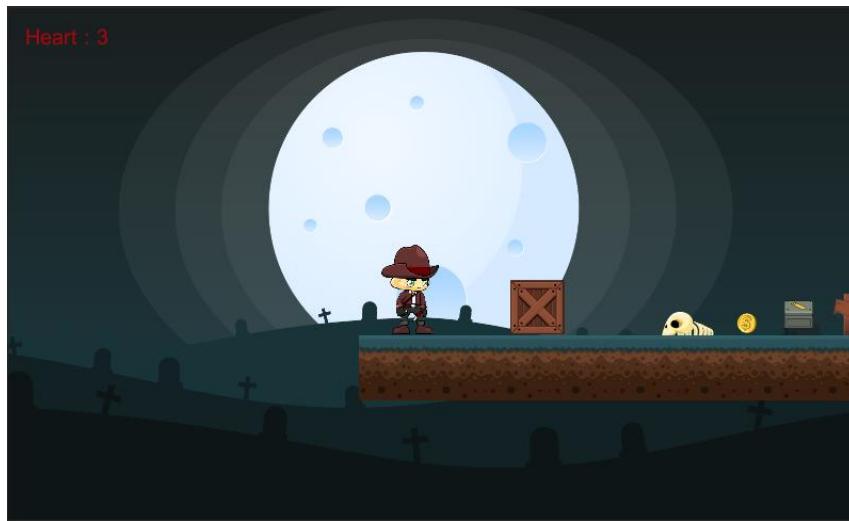
### Script pada Lifecycle Start

```
info_Heart = GameObject.Find("UI_Heart").GetComponent<Text>(); // Pendefinisian UI Heart sebagai komponen Text
```

### Script pada Lifecycle Update

```
info_Heart.text = "Heart : " + Heart.ToString(); //Heart yaitu Variabel di Atribut Player
```

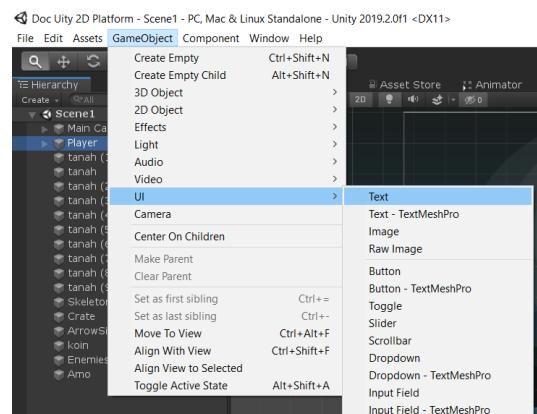
9. Simpan dan Play Game, Amati Hasilnya



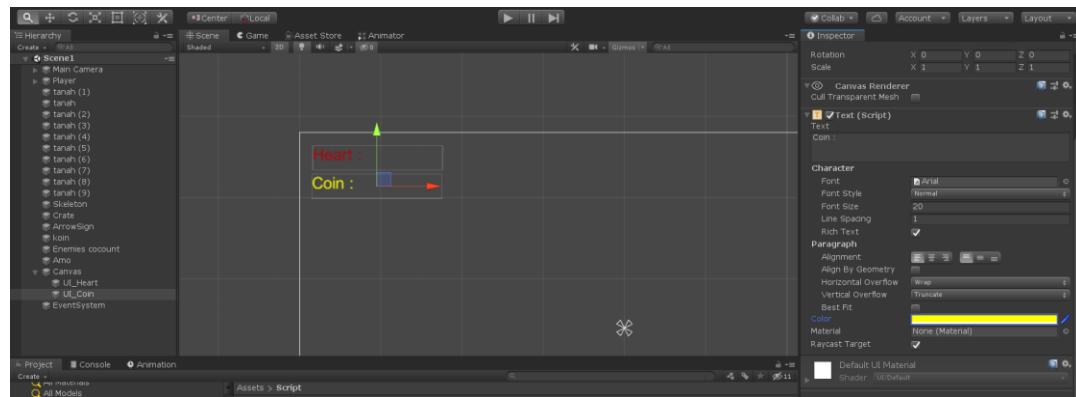
10. Nilai Heart, sudah bisa ditampilkan pada UI Game, yaitu pada game object UI\_Heart.

### b. Membuat UI untuk Coin

1. Buat game object baru untuk User Interface Coin,
2. Pilih menu Game Object → UI → Text.



3. Berinama Game Object UI tersebut : UI\_Coin
4. Select game Object UI\_Coin, Perhatikan pada bagian inspector, lakukan setting sesuai dengan kebutuhan, (Jenis Text, Ukuran text, warna, dll)



5. Setting juga untuk penempatan UI\_Coin.
6. Pada pembahasan sebelumnya terkait variabel Koin, sudah ada pada game object Player. Sekarang menyiapkan script coding untuk menampilkan nilai variabel Koin pada UI\_Coin.
7. Buka script gerak.cs, tambahkan script untuk menampilkan nilai Koin ke game object UI\_Coin sebagai berikut :

```
Text info_Coin; // Variabel untuk Koin
```

#### Script pada Lifecycle Start

```
info_Heart = GameObject.Find("UI_Heart").GetComponent<Text>(); // Pendefinisian UI Heart sebagai komponen Text
```

#### Script pada Lifecycle Update

```
info_Coin.text = "Coin : " + Koin.ToString(); //Coin yaitu Variabel di Atribut Player
```

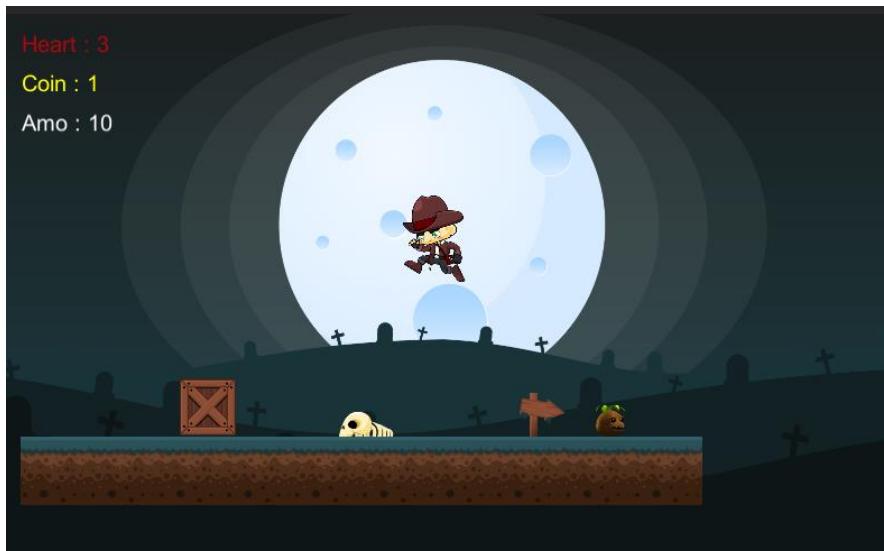
8. Simpan dan Play Game, Amati Hasilnya



9. Nilai Koin, sudah bisa ditampilkan pada UI Game, yaitu pada game object UI\_Coin.

### 3. Latihan Project

Lakukan hal yang sama untuk menampilkan Nilai Amo.



## K. UI Control

### 1. Teori Dasar

#### Kanvas

The **Canvas** adalah daerah bahwa semua **UI** elemen harus di dalam. Canvas adalah Obyek Game dengan komponen Canvas di atasnya, dan semua elemen **UI** harus anak-anak dari Canvas tersebut.

Membuat elemen UI baru, seperti Gambar menggunakan menu **GameObject**> **UI**> **Gambar**, secara otomatis membuat Kanvas, jika belum ada Kanvas dalam adegan. Elemen UI dibuat sebagai anak dari Kanvas ini.

Area Canvas ditampilkan sebagai persegi panjang di **Scene View Tampilan**. Ini memudahkan untuk memposisikan elemen UI tanpa perlu agar Tampilan Game terlihat setiap saat.

**Canvas** menggunakan objek EventSystem untuk membantu Sistem Pesan.

#### Gambar urutan elemen

Elemen UI di Kanvas digambar dalam urutan yang sama dengan yang muncul di Hierarki. Anak pertama ditarik pertama, anak kedua berikutnya, dan seterusnya. Jika dua elemen UI tumpang tindih, yang kemudian akan muncul di atas yang sebelumnya.

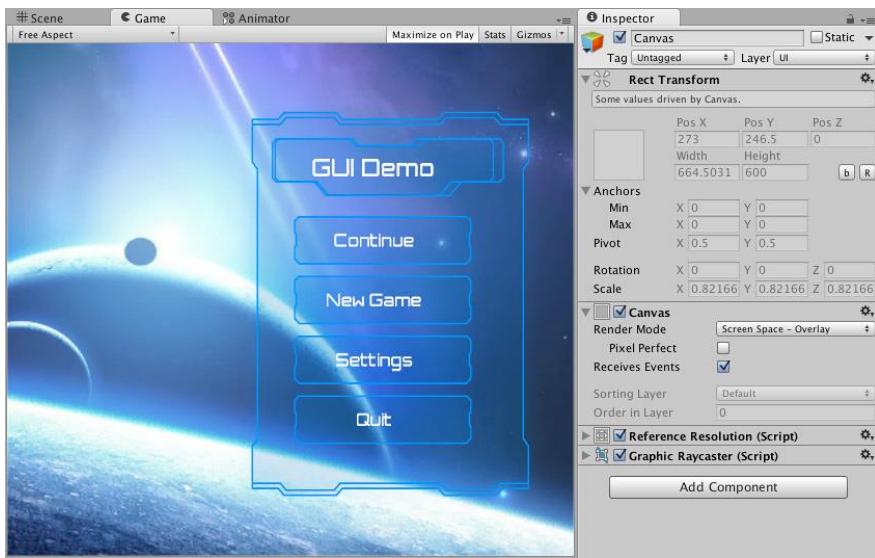
Untuk mengubah elemen mana yang muncul di atas elemen lain, cukup susun ulang elemen dalam Hirarki dengan menyeretnya. Urutan juga dapat dikontrol dari skrip dengan menggunakan metode ini pada **komponen**: SetAsFirstSibling, SetAsLastSibling, dan SetSiblingIndex.

#### Mode Render

Canvas memiliki pengaturan **Mode Render** yang dapat digunakan untuk membuatnya ditampilkan di ruang layar atau ruang dunia.

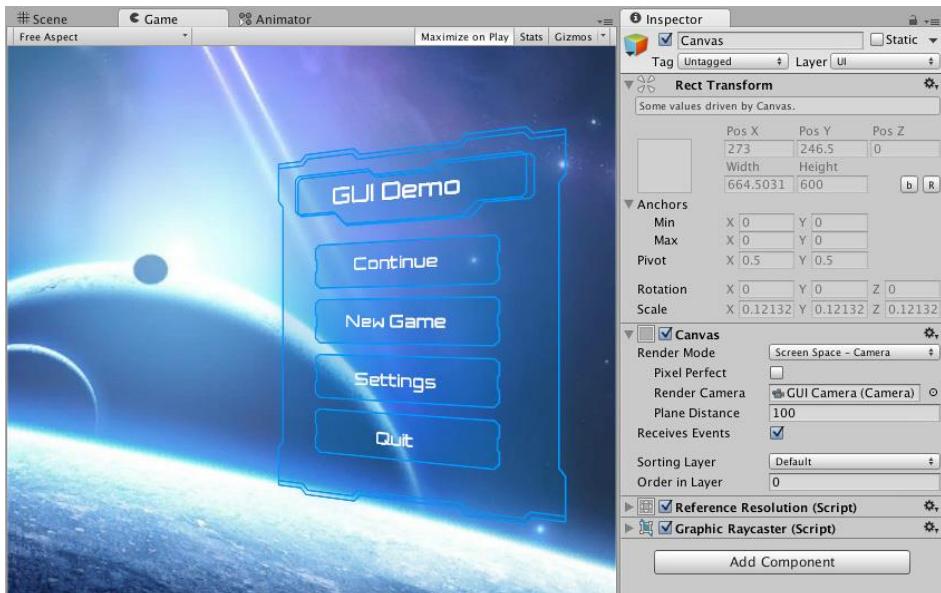
#### Layar Ruang - Hamparan

Mode render ini menempatkan elemen-elemen UI di layar yang ditampilkan di atas **layar**. Jika layar diubah ukurannya atau mengubah resolusi, kanvas akan secara otomatis mengubah ukuran untuk mencocokkan ini.



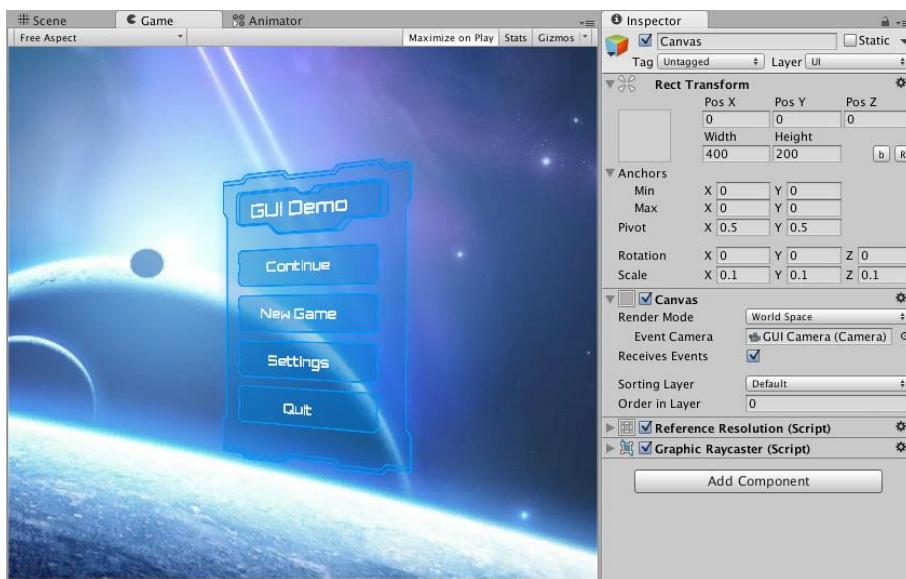
### Layar Space - Kamera

Ini mirip dengan **Screen Space - Overlay**, tetapi dalam mode render ini Canvas ditempatkan pada jarak tertentu di depan **Camera** ditentukan . Elemen UI diberikan oleh kamera ini, yang berarti bahwa pengaturan Kamera memengaruhi tampilan UI. Jika Kamera diatur ke **Perspektif**, elemen UI akan ditampilkan dengan perspektif, dan jumlah distorsi perspektif dapat dikontrol oleh **Bidang Bidang Tampilan Kamera**. Jika layar diubah ukurannya, perubahan resolusi, atau perubahan kamera frustum, kanvas akan secara otomatis mengubah ukuran agar sesuai.



## Ruang Dunia

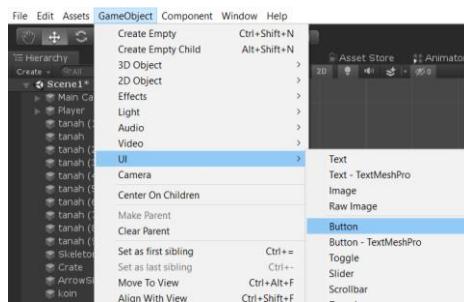
Dalam mode render ini, Kanvas akan berperilaku seperti objek lain di TKP. Ukuran Kanvas dapat diatur secara manual menggunakan Rect Transform-nya, dan elemen UI akan ditampilkan di depan atau di belakang objek lain dalam adegan berdasarkan penempatan 3D. Ini berguna untuk UI yang dimaksudkan untuk menjadi bagian dari dunia. Ini juga dikenal sebagai "antarmuka diegetik".



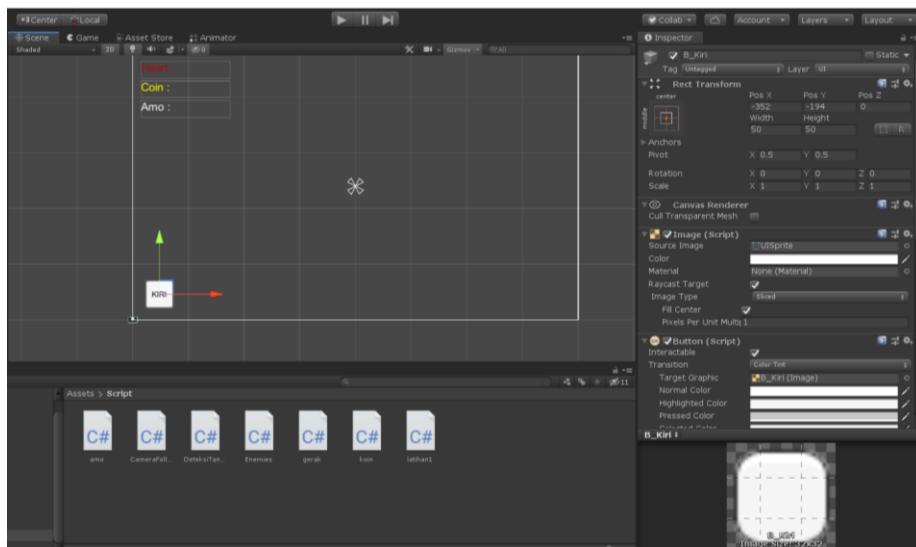
## 2. Praktikum

### a. Membuat UI untuk Control (Tombol Pergerakan ke Kiri)

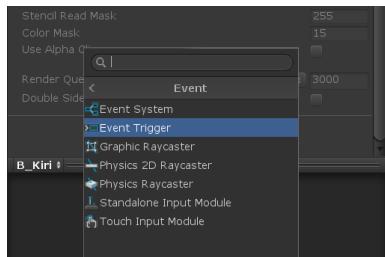
1. Buat game object baru untuk User Interface Button,
2. Pilih menu Game Object → UI → Button.



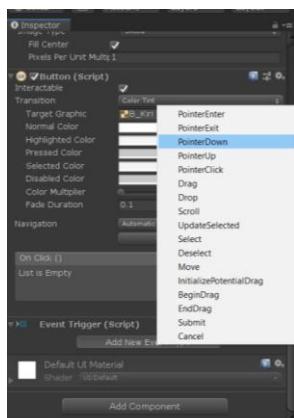
3. Berinama Game Object UI tersebut : B\_kiri
4. Select game Object B\_Kiri, Perhatikan pada bagian inspector, lakukan setting sesuai dengan kebutuhan, (Jenis Text, Ukuran text, warna, Picture dll)



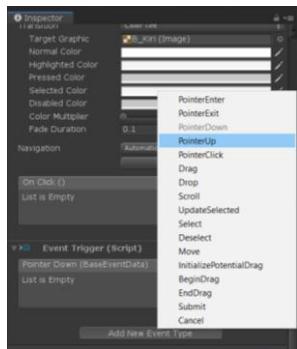
5. Setting juga untuk penempatan B\_Kiri.
6. Logic untuk button ini yaitu, ketika button di tekan, akan menggerakkan player bergerak ke kiri. Adapun untuk scrip program ditempatkan pada script *gerak.cs*.
7. Sebelumnya tambahkan komponen atribut event pada game object UI B\_kiri, yaitu Add Event Trigger : Pilih Game Object B\_kiri, pada inspector tambahkan add component → Event Triger



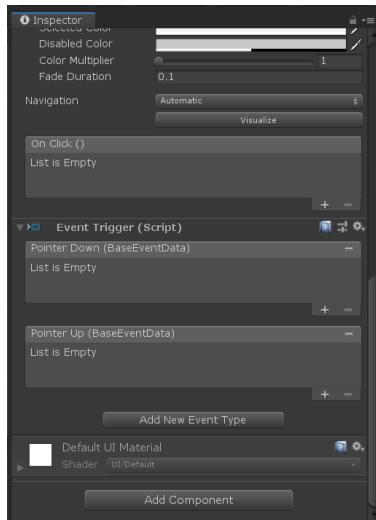
Event Button Down (Ketika Button di Tekan/Klik)



Event Button Up (Ketika Button tidak di Tekan/Klik)



Komponen Event Triger sebagai berikut:



8. Lakukan penambahan fungsi untuk event button kiri pada script *gerak.cs* sebagai berikut :

#### Variabel Button :

```
public bool Button_kiri; //Variabel untuk Button Kiri
```

#### Fungsi Baru untuk Button Up dan Button Down :

```
//Fungsi Button kiri
public void tekan_kiri()
{
    Button_kiri = true; //Ketika di Tekan
}
public void Lepas_kiri()
{
    Button_kiri = false; //Ketika dilepas
}
```

**Tambahkan Script untuk logic pada script sebelumnya**  
Sebelumnya :

```
else if (Input.GetKey (KeyCode.A)) //key A untuk bergerak ke kiri
```

Menjadi :

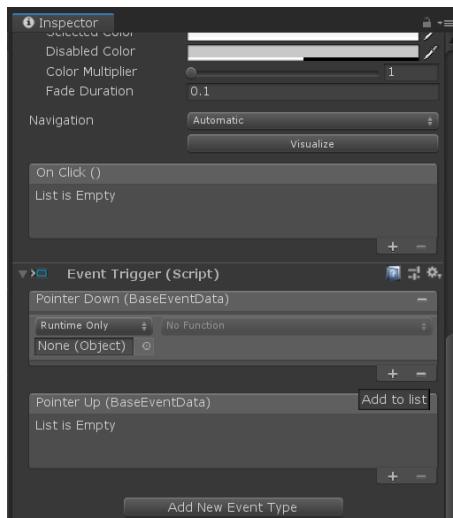
```
else if (Input.GetKey (KeyCode.A) || (Button_kiri == true))
```

### Hapus Script Klik Kiri

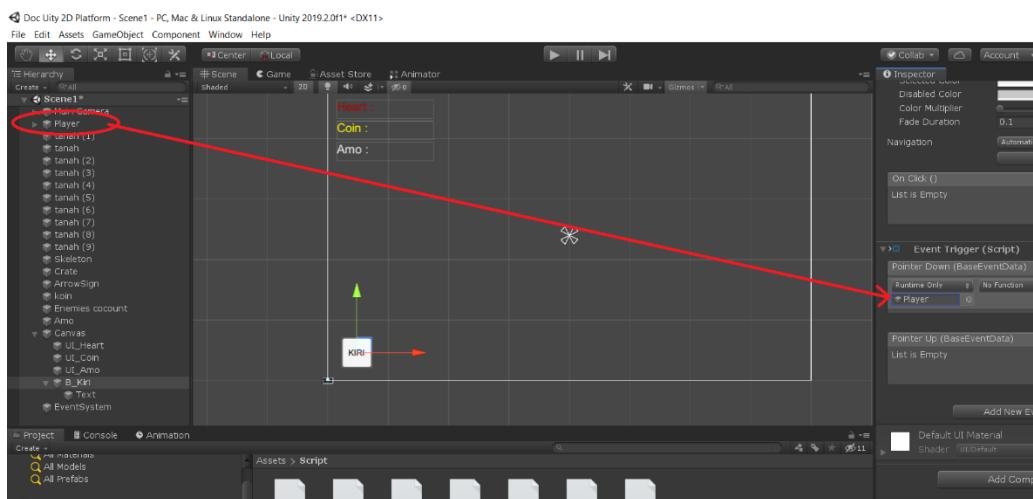
```
if(tanah == true && Input.GetKeyDown(KeyCode.Mouse0)) //Mouse0 = klik kiri
Mouse1=Klik Kanan
{
    lompat.AddForce(new Vector2(0,kekuatanlompat));
}
```

9. Lakukan Konfigurasi di inspector, Event Triger pointer down sebagai berikut :

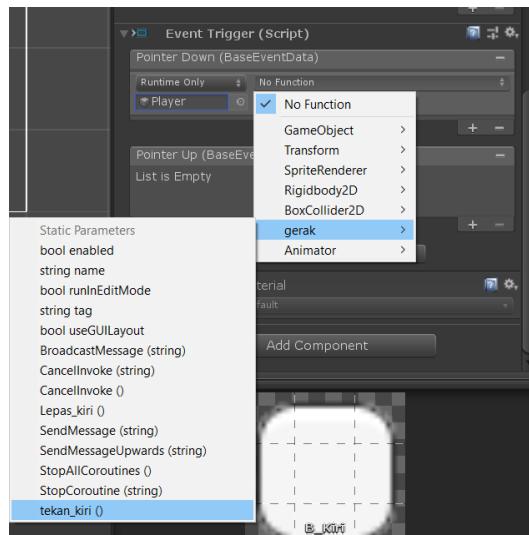
- Add to list pada event triger script.



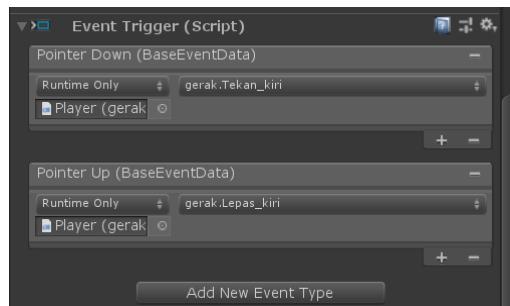
- Drag Game Object Player ke bagian Object list.



- Masukan Fungsi tekan kiri, dengan cara sebagai berikut :



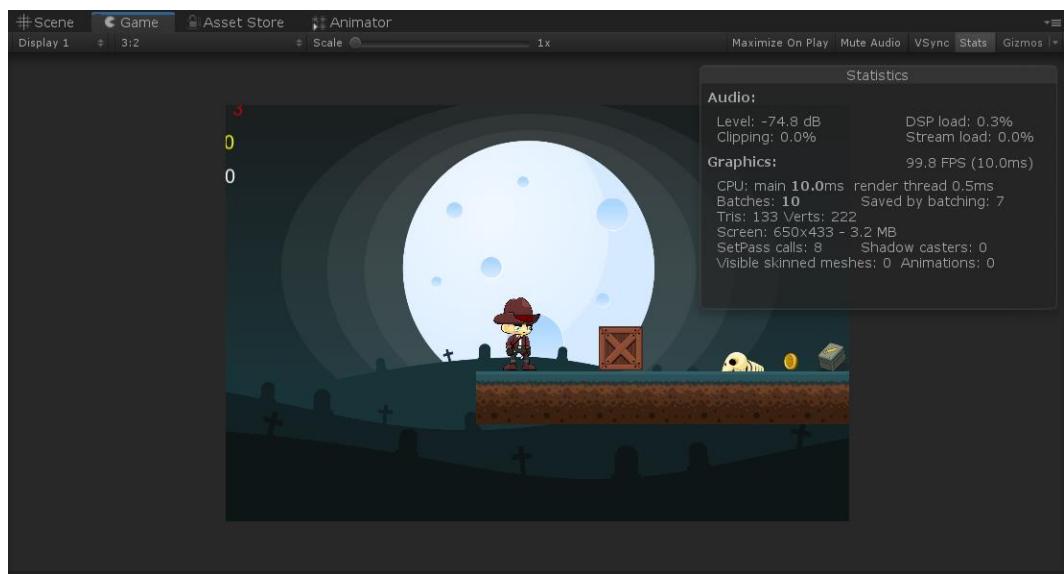
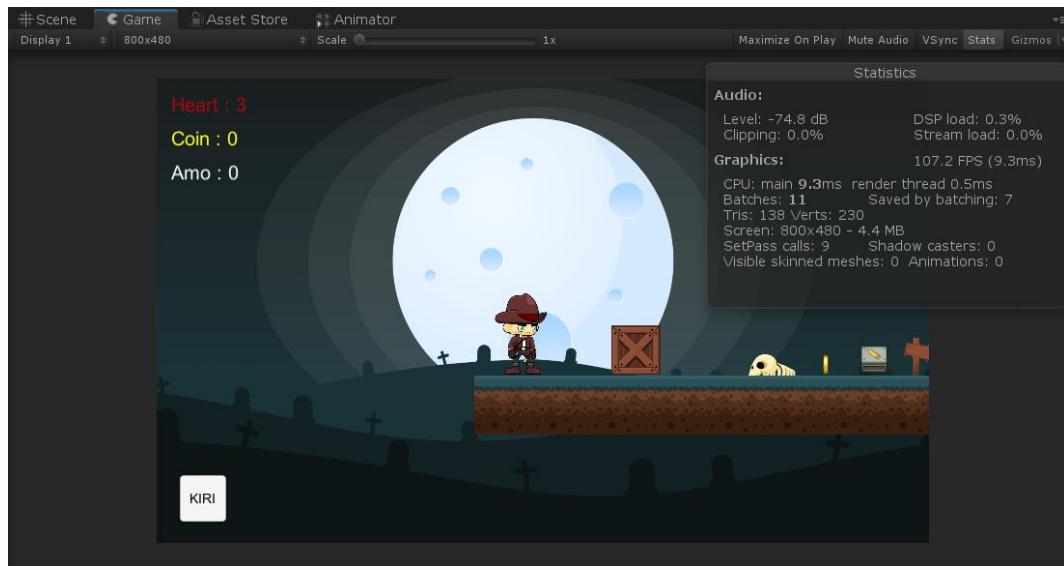
10. Ulangi langkah 9 untuk pointer Up, sehingga hasil even triger pointer sebagai berikut :



11. Play Game, Amati Hasilnya. (ketika tombol kiri di klik, maka player akan bergerak ke kiri.

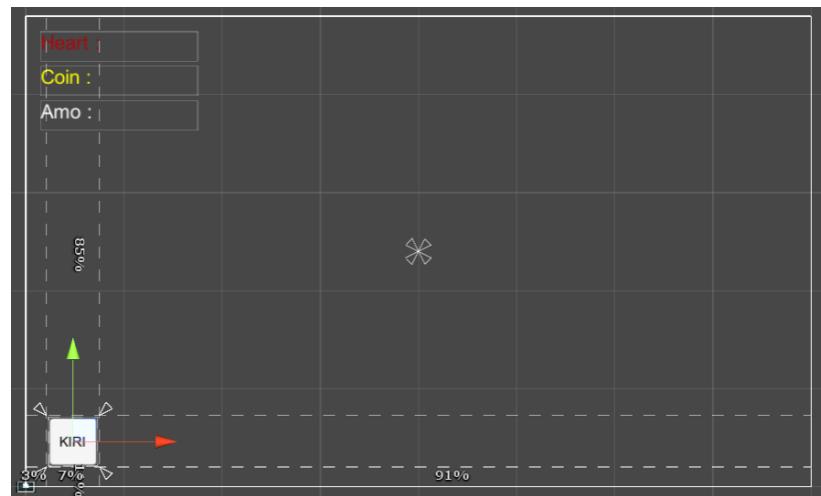
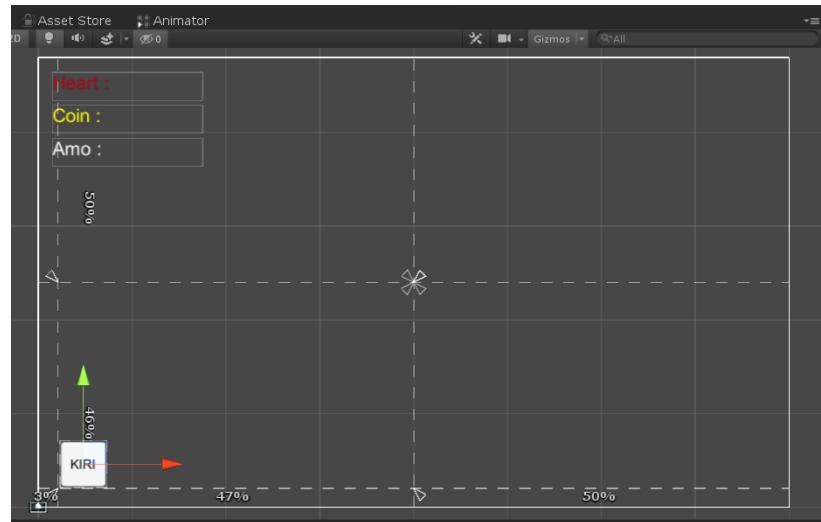
## b. Konfigurasi UI Responsif

- Pada dasarnya, ketika game di jalankan dengan resolusi yang sudah di atur (800x480),

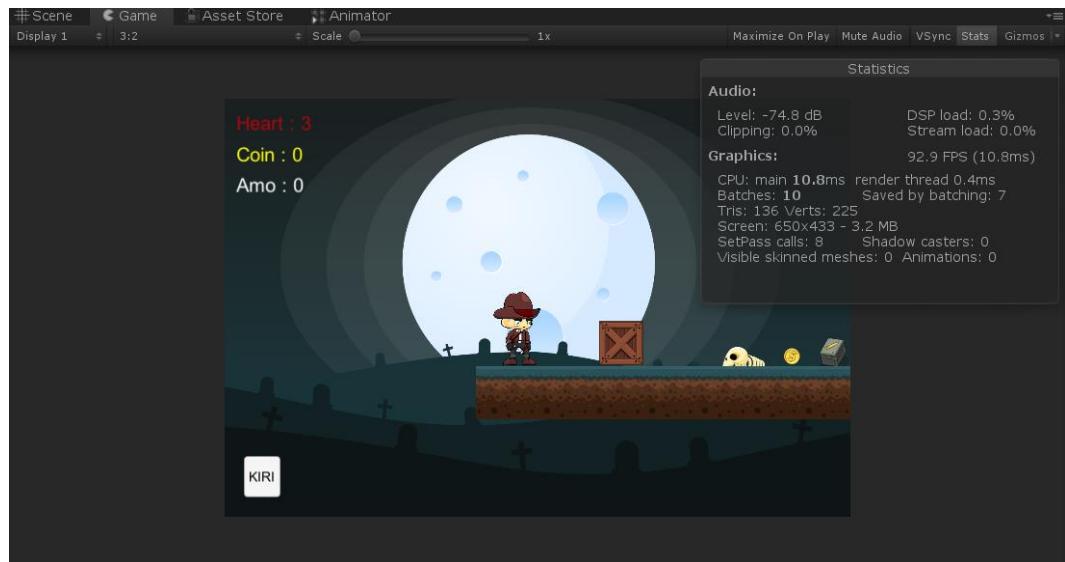
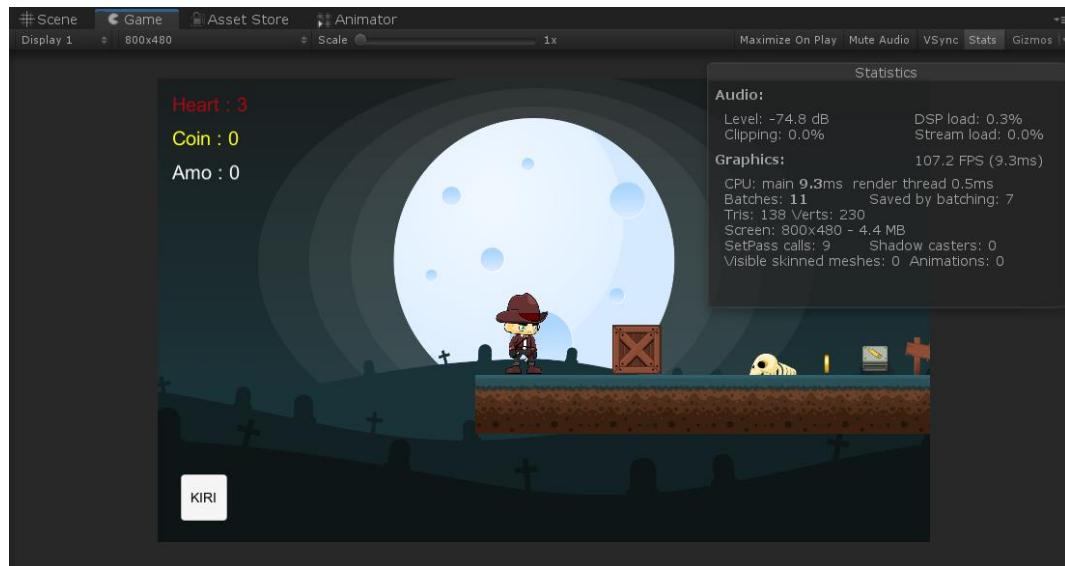


- Ketika dirubah menjadi maximize/minimize atau ukuran 3:2 dari ukurannya, seharusnya User Interface mengikuti dengan resolusi yang dilakukan. Oleh karena itu perlu pengaturan UI yang responsive agar tampilannya mengikuti resolusi yang diinginkan.
- Untuk itu, perlu setting Rec transform di semua objek UI, sehingga membuat UI menjadi responsive. Untuk pengaturannya ikuti langkah berikut :
  - Klik objek UI, lalu drag cursor objek transform ke bagian setiap sudut UI.

- Perhatikan untuk penempatan Rec Transform untuk Tombol Kiri dibawah ini

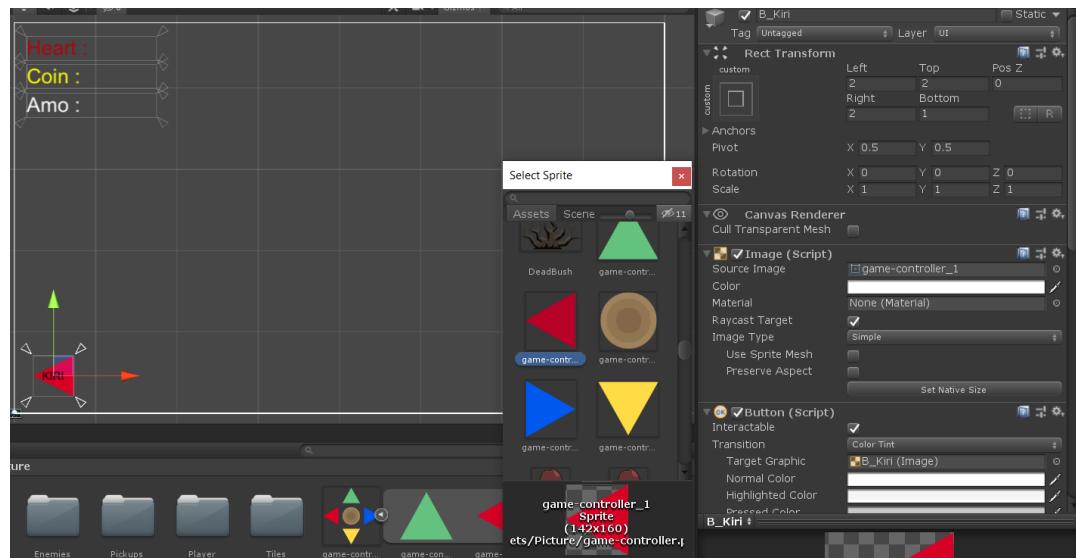


4. Ulangi, untuk semua komponen UI, Lakukan satu persatu.
5. Play game, Lakukan perubahan resolusi. Jika berhasil akan seperti di bawah ini :



### c. Mengubah Button Standar dengan Image

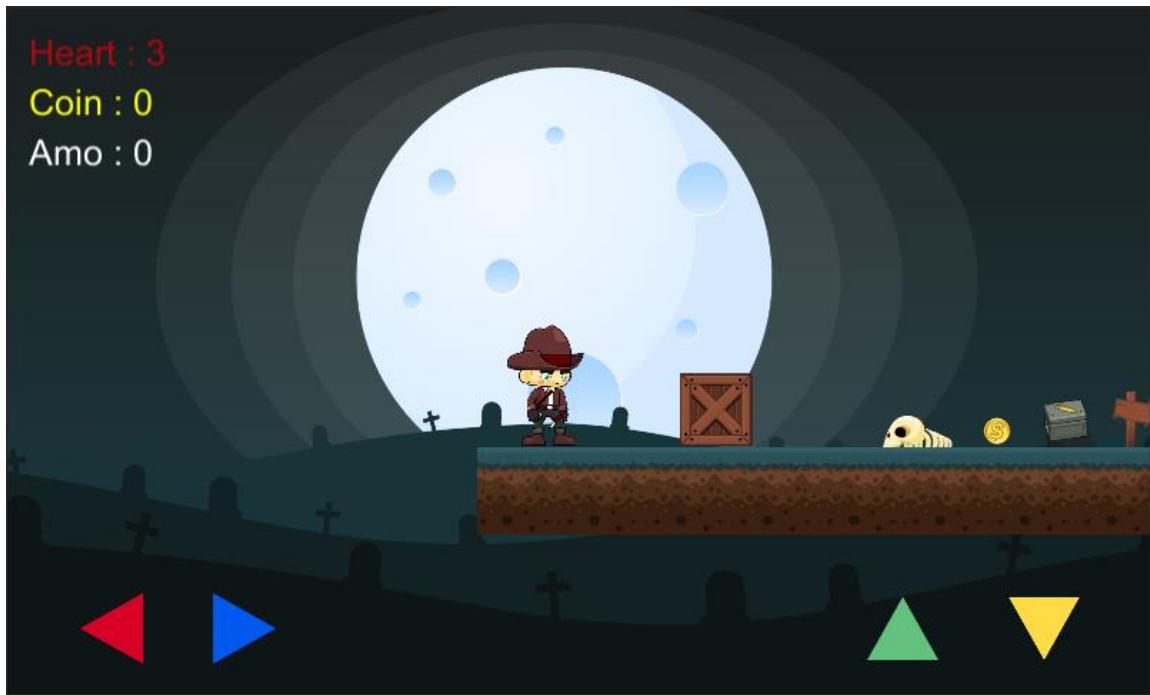
1. Pilih Button yang akan dirubah,
2. Perhatikan pada bagian inspector, pada atribut image script → klik lingkaran source image → pilih image yang akan dijadikan Button.



3. Setelah dipilih, jangan lupa untuk menceklist Preserve Aspect, agar image mengikuti ukuran button yang ditentukan.
4. Play Game, amati hasilnya

### 3. Latihan Project

Lengkapi pembuatan UI untuk Tombol Kanan dan Lompat, dan Setting agar Responsif, serta tambahkan image yang sesuai untuk setiap button.



## L. UI Win and Lose

### 1. Teori Dasar

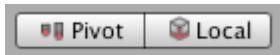
#### The Rect Tool

Every UI elemen direpresentasikan sebagai persegi panjang untuk keperluan tata letak. Kotak ini dapat dimanipulasi dalam **Tampilan Pemandangan**. Tampilan menggunakan **Rect Tool** di bilah alat. Rect Tool digunakan baik untuk fitur 2D Unity dan untuk UI, dan pada kenyataannya dapat digunakan bahkan untuk **objek** juga.



Rect Tool dapat digunakan untuk memindahkan, mengubah ukuran, dan memutar elemen UI. Setelah Anda memilih elemen UI, Anda dapat memindahkannya dengan mengklik di mana saja di dalam persegi panjang dan menyeret. Anda dapat mengubah ukurannya dengan mengklik tepi atau sudut dan menyeret. Elemen dapat diputar dengan mengarahkan kurSOR sedikit jauh dari sudut sampai kurSOR mouse terlihat seperti simbol rotasi. Anda kemudian dapat mengklik dan menyeret ke salah satu arah untuk memutar.

Sama seperti alat lainnya, Alat Persegi Panjang menggunakan mode dan ruang pivot saat ini, yang diatur dalam bilah alat. Saat bekerja dengan UI, biasanya ide yang baik untuk mengurnya menjadi **Pivot** dan **Lokal**.



#### Rect Transform

**Rect Transform** adalah baru **mengubah komponen** yang digunakan untuk semua elemen UI alih-alih komponen **Transform** biasa .



Rect Transforms memiliki posisi, rotasi, dan skala seperti Transforms biasa, tetapi juga memiliki lebar dan tinggi, yang digunakan untuk menentukan dimensi persegi panjang.

### Mengubah Ukuran Versus Penskalaan

Ketika Rect Tool digunakan untuk mengubah ukuran suatu objek, biasanya untuk **Sprite** dalam sistem 2D dan untuk objek 3D itu akan mengubah *skala* lokal objek. Namun, ketika itu digunakan pada objek dengan Rect Transform di atasnya, itu malah akan mengubah lebar dan tinggi, menjaga skala lokal tidak berubah. Pengubahan ukuran ini tidak akan memengaruhi ukuran font, batas pada gambar yang diiris, dan sebagainya.

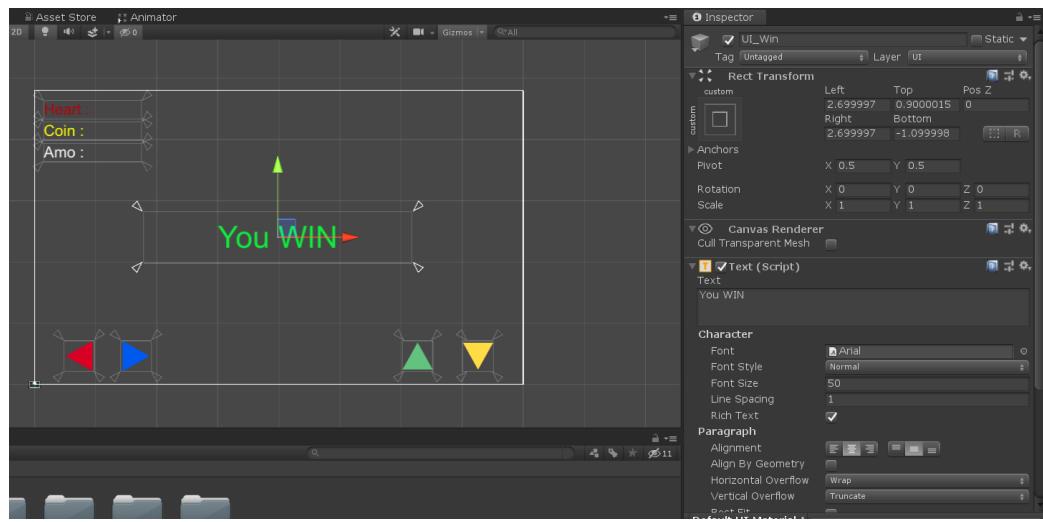
### Poros

Rotasi, ukuran, dan modifikasi skala terjadi di sekitar pivot sehingga posisi pivot memengaruhi hasil rotasi, pengubahan ukuran, atau penskalaan. Saat tombol Pivot bilah alat disetel ke mode Pivot, pivot dari Rect Transform dapat dipindahkan di **Scene**.

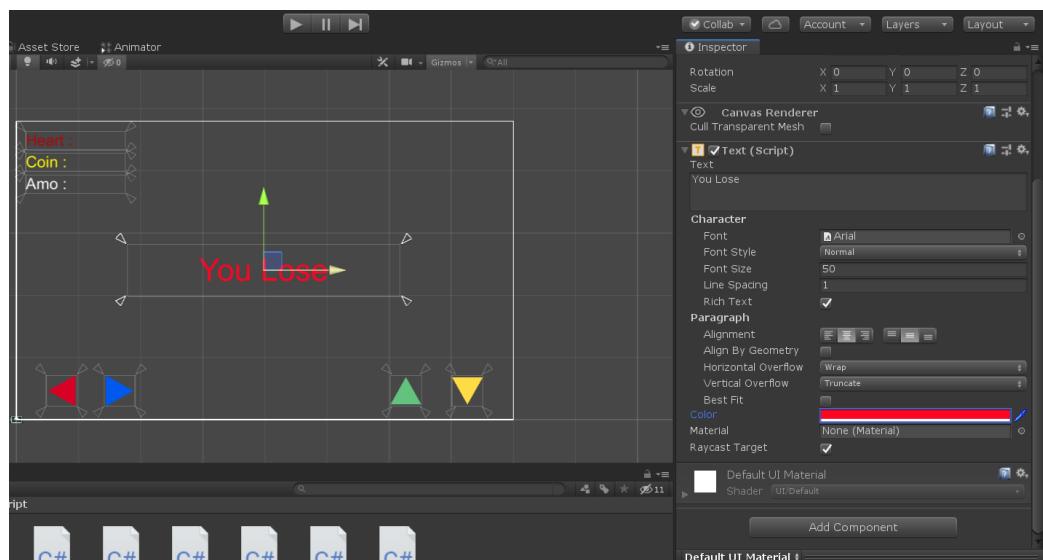
## 2. Praktikum

### a. Membuat Tampilan UI Win dan Lose

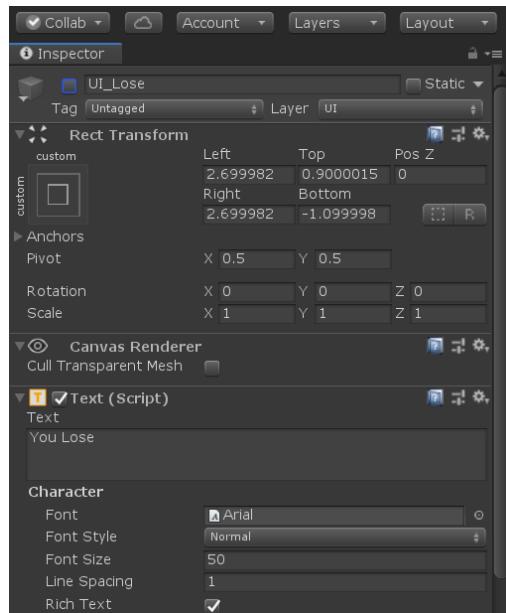
1. Buat game object baru untuk User Interface Kemenangan,
2. Pilih menu Game Object → UI → Text
3. Berinama Game Object UI tersebut : UI\_Win
4. Perhatikan pada bagian inspector, lakukan setting sesuai dengan kebutuhan, (Jenis Text, Ukuran text, warna, dll).
5. Setting juga untuk penempatan UI\_Heart dan Rec.Transformnya.



6. Ulangi langkah 2-5 untuk membuat UI kalah.



7. Setting UI\_Win dan UI\_Lose Deactive/Unckecklist



8. Buka script gerak.cs, tambahkan script logic untuk kemenangan :

- Buat variabel

```
public GameObject menang,kalah; //variabel untuk menang dan kalah
```

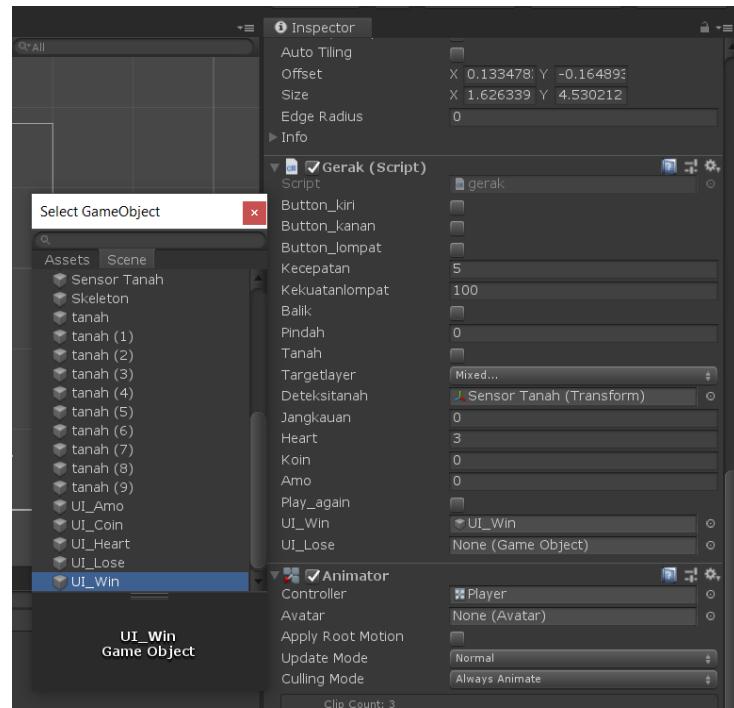
- Tambahkan kode untuk kondisi (Jika Sebelumnya

```
//Logik Heart
if(Heart<0)
{
    Destroy(gameObject);
}
```

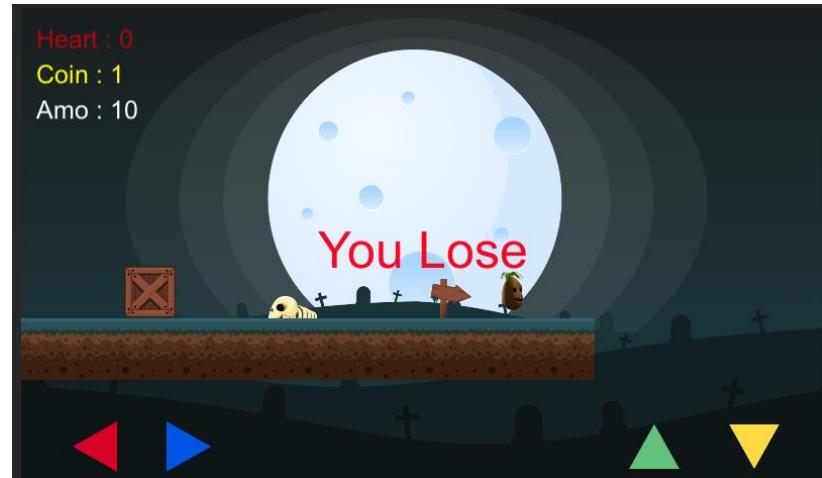
Menjadi

```
//Logik Heart
if(Heart <= 0)
{
    Destroy(gameObject);
    UI_Lose.SetActive(true); // UI Kalah aktif
}
else if (Koin >= 5)
{
    UI_Win.SetActive(true); //UI Menang Aktif
}
```

9. Setting Nilai UI\_Win dan UI\_Lose pada inspector player inputan UI\_Win dan UI\_Lose.



#### 10. Play Game, dan Amati Hasilnya.



### 3. Latihan Project

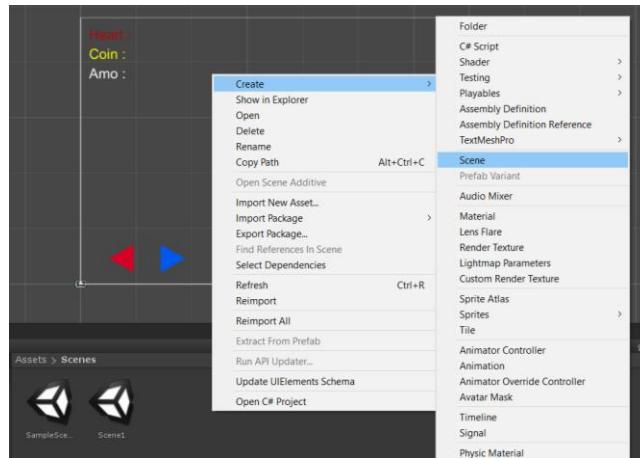
Buatlah Logic ketika Player jatuh dari Tanah/area Game akan memunculkan UI You Lose (UI\_Lose).

## M. Main Menu

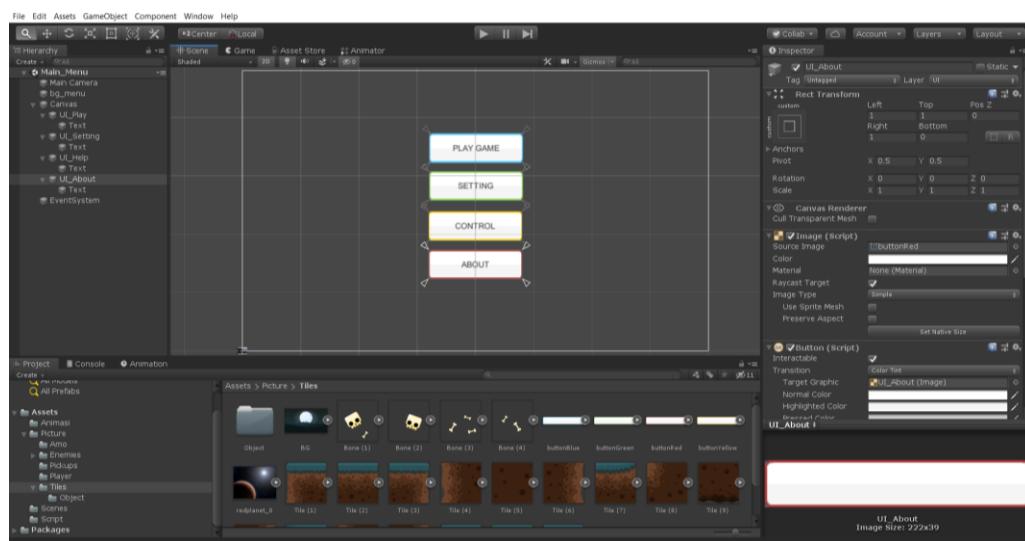
### 1. Praktikum

#### a. Membuat Scene Main Menu

- Buat Scene Baru dengan Nama Main\_Menu



- Buat game object baru untuk User Interface Button,
- Pilih menu Game Object → UI → Button
- Berinama Game Object UI tersebut : UI\_Play dan UI\_Setting, UI\_Control, UI\_About.
- Perhatikan pada bagian inspector, lakukan setting sesuai dengan kebutuhan, (Jenis Text, Ukuran text, warna, Objek dll).



6. Buatlah script baru dengan nama Goto.cs dengan script sebagai berikut :

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

public class Goto : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

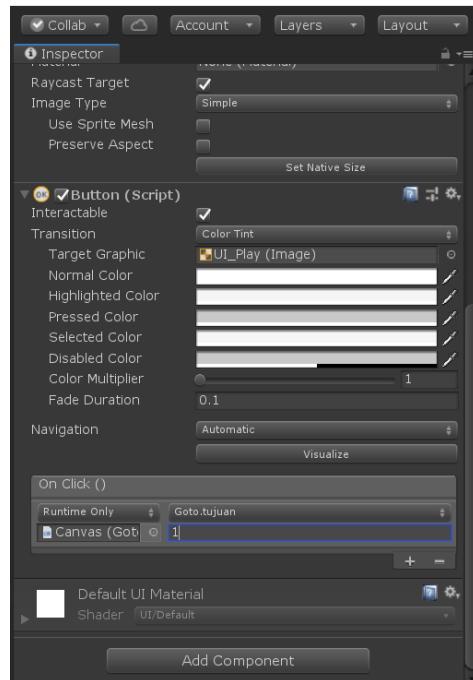
    }

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

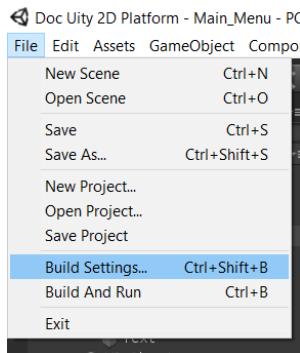
    }

    public void tujuan(int mainmenu)
    {
        Application.LoadLevel(mainmenu);
    }
}
```

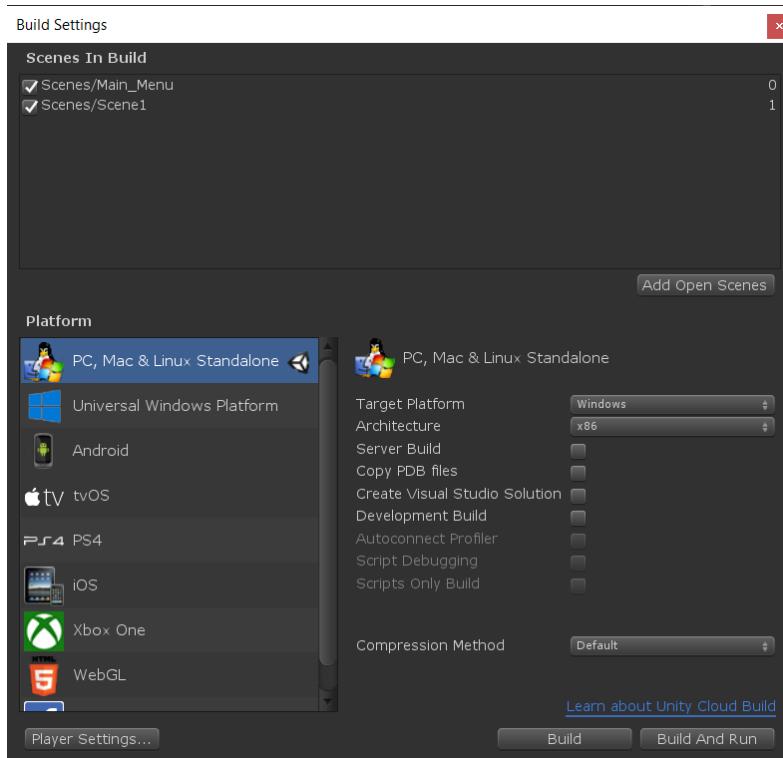
7. Masukan script *Goto.cs* ke game object Canvas  
 8. Selanjutnya, pilih game object UI\_Play, lalu pada bagian inspector tab button script, tambahkan list dengan konfigurasi sebagai berikut :



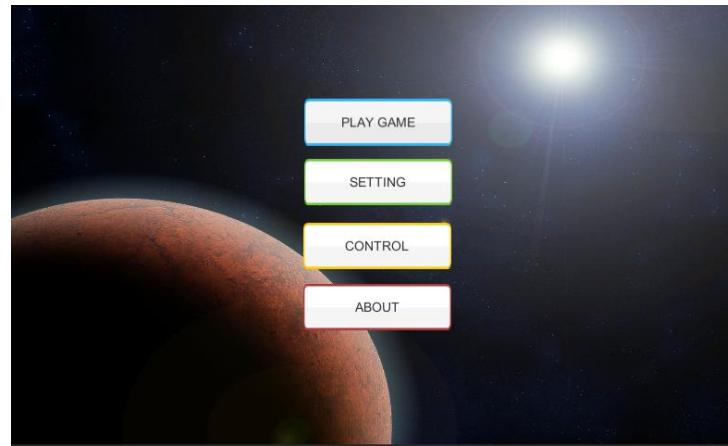
9. Setting Build Scene dengan cara masuk ke file → Build Setting



10. Akan muncul windows build setting, selanjutnya drag scene ke area windows tersebut.



11. Save, Play Game dan Amati Hasilnya



### b. Membuat Scene Back Menu

1. Buka kembali Scene1
2. Buat script baru dengan nama *Back.cs*
3. Masukan kode program berikut :

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

public class Back : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            Application.LoadLevel(0);
        }
    }
}
```

4. Masukkan script *Back.cs* ke dalam Game Object Main Camera di Scene1.
5. Simpan project tersebut, dan Play game, amati hasilnya.

## 2. Latihan Project

Buatlah Scene yang lain untuk Control, setting, about dan konfigurasikan perpindahannya.

## N. Build Game

### 1. Teori Dasar

#### Pengaturan Build

Anda dapat menggunakan jendela Build Settings untuk memilih platform target Anda, menyesuaikan pengaturan untuk build Anda, dan memulai proses build. Untuk mengakses jendela Build Pengaturan, buka **File > Build Setting**. Setelah Anda menentukan pengaturan build Anda, Anda dapat mengklik **Build** untuk membuat build Anda, atau klik **Build And Run** untuk membuat dan menjalankan build Anda di platform yang Anda tentukan.

#### Adegan dalam Build

Bagian jendela ini menunjukkan Adegan dari Proyek Anda yang akan dimasukkan dalam Build Anda. Jika Anda tidak dapat melihat Adegan apa pun di area ini, gunakan tombol *Add Open Scenes* untuk menambahkan Scene saat ini ke build, atau Anda dapat menyeret aset Scene ke jendela ini dari jendela Project Anda. Anda juga dapat menghapus centang Adegan dalam daftar ini untuk mengeluarkannya dari build tanpa menghapusnya dari daftar. Jika Anda tidak memerlukan adegan dalam pembuatan, Anda dapat menghapusnya dari daftar adegan dengan menekan tombol Delete di keyboard Anda.

Adegan yang Anda centang dan tambahkan ke daftar **Adegan dalam Build** termasuk dalam memBuild. Unity menggunakan daftar Adegan untuk mengontrol urutan Adegan dimuat. Untuk menyesuaikan urutan Adegan, seret ke atas atau ke bawah daftar.

#### Daftar platform

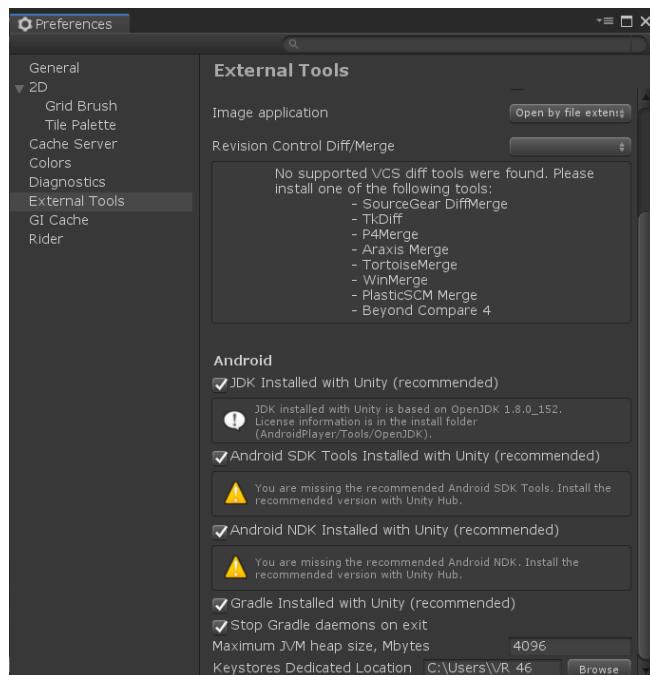
Area Platform di bawah area **Adegan dalam Build** mencantumkan semua platform yang tersedia untuk versi Unity Anda. Beberapa platform mungkin diklik untuk menunjukkan bahwa mereka bukan bagian dari versi Anda. Untuk mengontrol platform mana yang akan dibangun, pilih salah satu platform dalam daftar. Jika Anda mengubah platform target, Anda perlu menekan tombol **Switch Platform** untuk menerapkan perubahan Anda. Ini mungkin memerlukan beberapa, karena aset Anda mungkin perlu diimpor kembali dalam format yang sesuai dengan platform target Anda. Platform yang Anda pilih ditandai dengan ikon Persatuan di sebelah kanan nama platform.

Platform yang Anda pilih menunjukkan daftar opsi yang dapat Anda sesuaikan untuk build. Setiap platform mungkin memiliki opsi berbeda. Opsi-opsi ini tercantum di bawah ini. Opsi yang umum di banyak platform tercantum di bagian paling bawah dari bagian ini di bawah item Generik di detail bangunan.

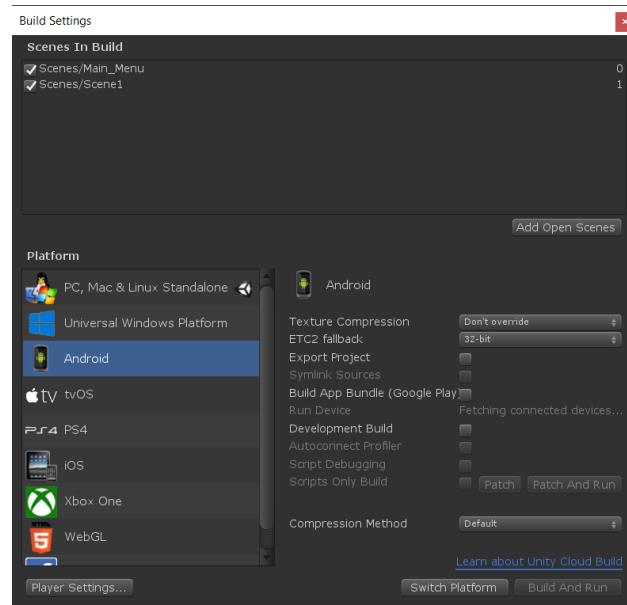
## 2. Praktikum

### a. Build Game Berbasis Mobile Android

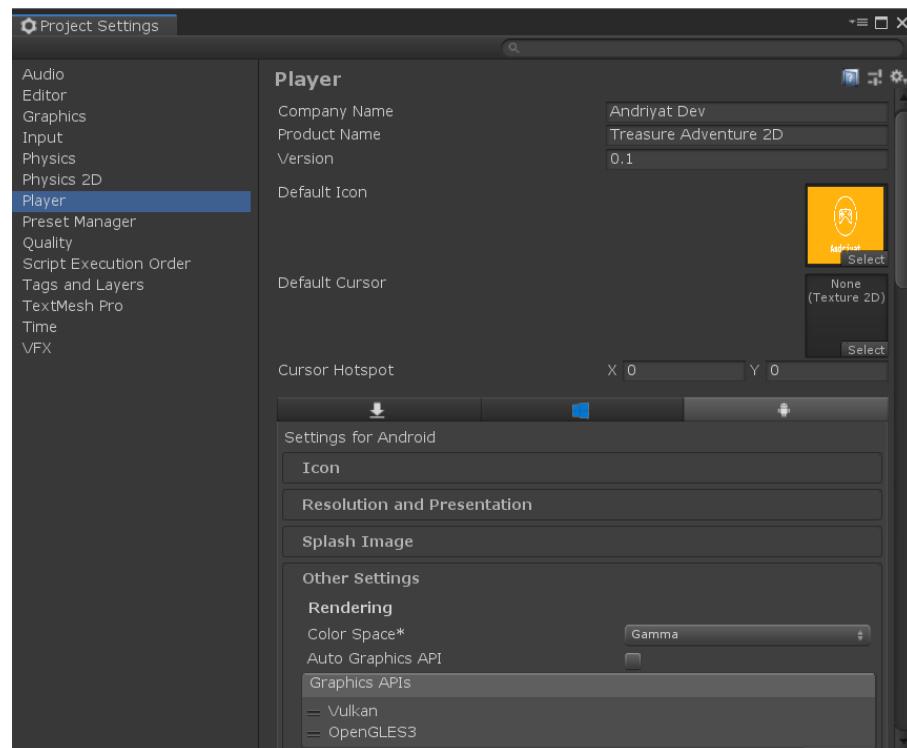
1. Untuk Build Game tersebut ke Berbasis Mobile Android ada beberapa software/library pendukung, diantaranya :
  - *JDK (Java Development Kit)*
  - *SDK (Software Development Kit)*
2. Selanjutnya masuk ke menu Edit → Preference
3. Tentukan penempatan lokasi JDK dan SDK, close apabila sudah.



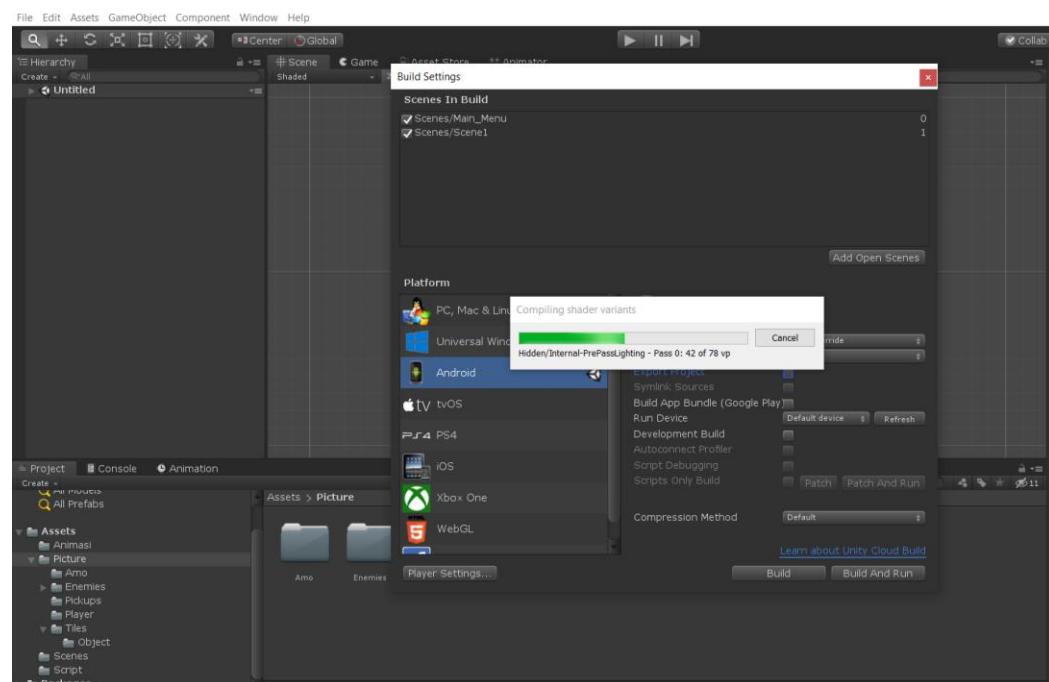
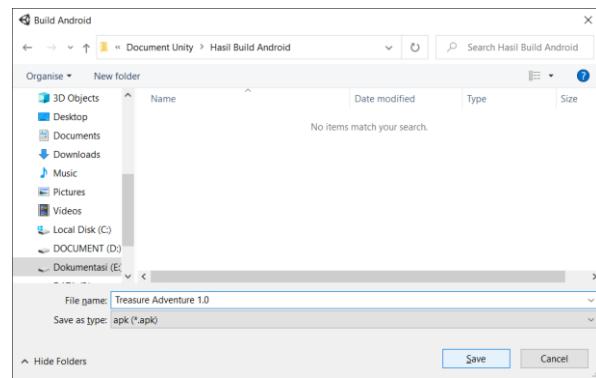
4. Selanjutnya Masuk ke Menu File→Build Setting
5. (Sebelumnya atur terlebih dahulu scene yang akan ditampilkan).
6. Pilih Platform Android, lalu klik Player Setting

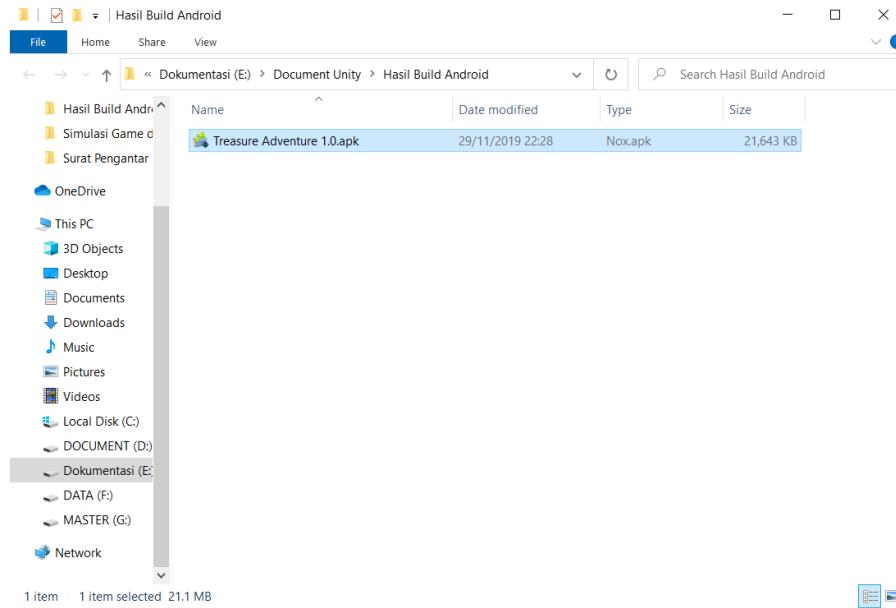


7. Lakukan Pengaturan sebagai berikut :



8. Jika Pengaturan sudah selesai, KLIK BUILD untuk Melakukan Build ke APK dan Tentukan Lokasi Penyimpanan File Apk.





### 3. Latihan Project

Lengkapi Project diatas sehingga menjadi sebuah game yang menarik untuk dimainkan.

## O. Glossary

### 1. Tab Unity

- **Tab Hierarchy**

Window yang berisi game object atau kumpulan game object yang di gunakan di dalam scene.

- **Tab Inspector**

Window yang menampilkan konteks atau keterangan dari object atau asset yang sedang dipilih. Window ini bisa menampilkan informasi property atau component dari sebuah game object ataupun asset.

- **Tab Project**

Window yang berisikan seluruh asset yang digunakan untuk membuat proyek game, bisa terdiri dari file, script, texture, 3D model, audio clip, asset dan lain-lain.

- **Tab Console**

Tempat penampilan pesan error dalam project, tapi paling sering terjadi pesan error saat pembuatan Script yang tidak dikenali oleh System atau kesalahan-kesalahan dalam pembuatan script

- **Tab Scene**

Tempat meletakan komponen seperti Camera, Terrain, Object, dll. Pada Tab ini dapat melakukan penempatan atau penggerjaan game secara keseluruhan dengan object yang bearada di Tab Assets.

- **Tab Game**

Window ini digunakan untuk melihat tampilan ketika permainan di jalankan. Di bagian atas, terdapat tombol play, pause, dan set frame by frame yang akan ditampilkan pada window Game

## 2. Tab Hiracy

## 3. Transform Tools



- **Hand tools**

Digunakan untuk menggerakan posisi sudut pandang di dalam scene.

- **Position transform tools**

Digunakan untuk mengubah komponen posisi sebuah Game Object terhadap sumbu x, y, dan z.

- **Rotation transform tools**

Digunakan untuk mengubah komponen rotasi pada sebuah Game Object terhadap sumbu x, y, dan z.

- **Scale transform tools**

Digunakan untuk mengubah komponen scale/ukuran dari sebuah Game Object terhadap sumbu x, y, dan z.

- **Rectangle transform tools**

Digunakan untuk mengubah komponen dari sudut pandang 2D. Biasanya tools ini akan sangat membantu jika ingin memodifikasi komponen UI dalam game.

## 4. Tab Scene

- **Scene**

Window yang digunakan untuk membangun game. Di dalamnya bisa melihat dan mengatur object di dalam sebuah scene.

### Toolbar

Toolbar berisikan tombol yang membantu untuk mengatur berbagai komponen di dalam permainan.

## 5. Other

- **Sprite**

Objek grafik 2D

- **Sprite Packer**

Sebuah fasilitas yang mengemas gambar dari beberapa **sprite** tekstur yang rapat dalam satu tekstur yang dikenal sebagai atlas. Unity menyediakan fasilitas **Sprite Packer** utilitas untuk mengotomatiskan proses menghasilkan atlas dari tekstur **sprite** individu.

- **Scene View**

Tampilan interaktif ke dunia yang Anda buat. Anda menggunakan Tampilan Scene untuk memilih dan memposisikan sceane, karakter, kamera, lighting, dan semua jenis Objek Game lainnya.

- **Skybox**

Material yang mewakili langit dalam pembuatan game di unity.

- **Sprite**

Objek grafik 2D

- **Sprite**

Objek grafik 2D

- **Transform**

Komponen property game object untuk menentukan posisi , Rotasi , dan Skala dari setiap objek dalam Scene. Setiap GameObject memiliki Transform.

- **MonoBehaviour** adalah *class* dasar dari setiap *script* Unity yang dibuat. Ketika menggunakan C# di unity, MonoBehaviour secara otomatis (*default*) akan tercipta. Sebagai kendali ke inspector objek game

## REFERENCE

<https://www.dicoding.com/blog/mengenal-komponen-pada-user-interface-unity/> (Diakses, 19 November 2019)

<https://docs.unity3d.com/Manual/index.html> (Diakses, 19 November 2019)

*Roedavan, Rickman. (2018). Unity Tutorial Game Engine (Revisi kedua). Penerbit Informatika. Bandung.*

<https://www.gameart2d.com/> (Diakses, 19 November 2019)

## File Project

File Project UNITY :

<https://drive.google.com/file/d/161-0Iq7ugHS8RIN1s65ZYySigEcNC-rf/view?usp=sharing>

Modul :

<https://staff.uniku.ac.id/rioandriyat/modul-unity-2d-platformers/>

Video Demo Hasil :

<https://youtu.be/55un61ByLBE>