PEMROGRAMAN GAME
JOB SHEET MINGGU KE-8

Kelompok

Maulinda Arum (3.34.21.0.13)

Nadya Rahmania Putri (3.34.21.0.18)

Rangga Prabaswara (3.34.21.0.21)

Yasmin Anindita Azzahra

(3.34.21.0.24)

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK ELEKTRO

POLITEKNIK NEGERI SEMARANG

2022/2023

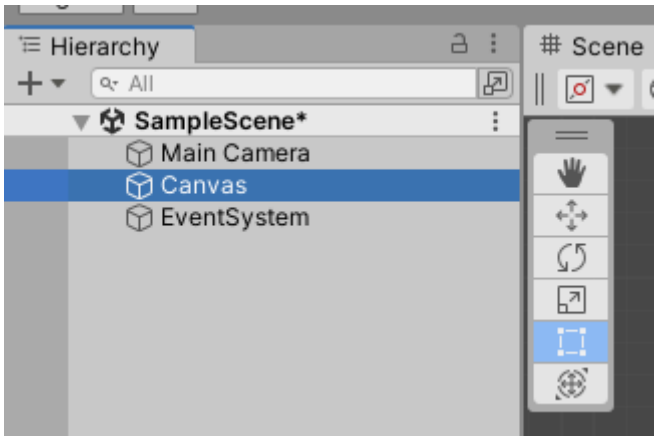
I. Tujuan Instruksional Khusus

Setelah melekukan praktikum ini Mahasiswa mampu memahami dan menerapkan konsep dasar game.

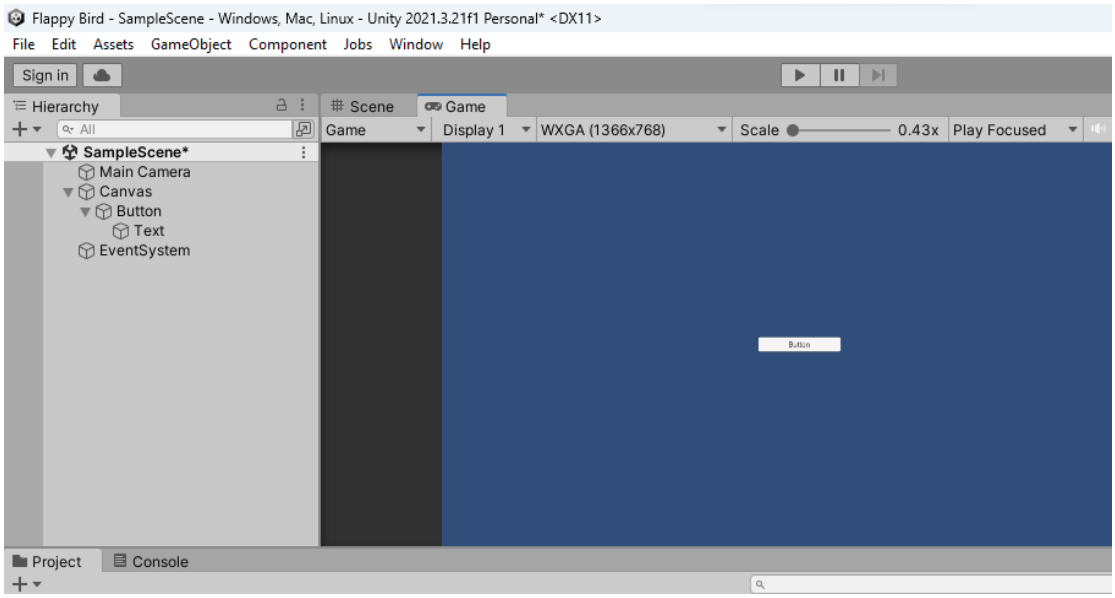
II. Dasar Teori

Canvas

Canvas adalah area dimana semua elemen UI harus ada di dalamnya. *Canvas* adalah *parent* sementara elemen UI adalah *child.* Kita dapat menambahkan Canvas dengan cara klik kanan pada **Hierarchy > UI > Canvas**.



Saat kita membuat *canvas* baru, maka *canvas* tersebut akan menjadi *parent* dari elemen UI.



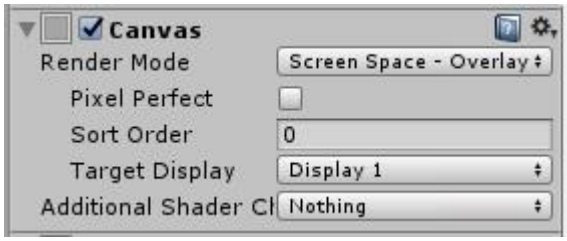
Jika kita membuat elemen UI baru seperti tombol (*button)* dengan cara klik kanan pada **Hierarchy > UI > Button**, otomatis terciptalah *canvas* baru. Elemen UI (*button*) akan otomatis menjadi *child* pada *canvas*.

Render Modes

Canvas memiliki pengaturan **Render Mode** yang dapat digunakan untuk ditampilkan pada layar.

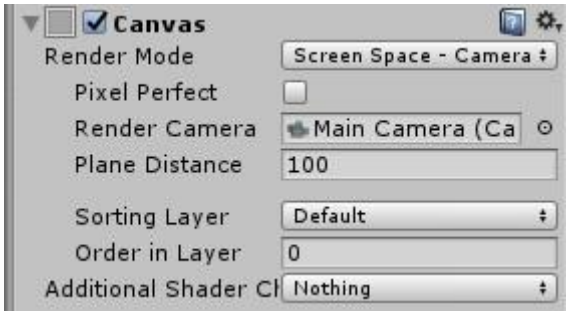
- **Screen Space - Overlay**

  Mode render ini menempatkan elemen UI yang ditampilkan pada layar. Jika kita mengubah ukuran layar atau resolusi, *canvas* akan secara otomatis menyesuaikan ukuran agar selaras dengan keduanya.



- **Screen Space - Camera**

  Screen Space - Camera juga mirip dengan Screen Space - Overlay, tetapi dalam mode render ini *canvas* ditempatkan pada jarak tertentu di depan kamera yang ditentukan. Kamera ini memberikan elemen UI yang berarti bahwa pengaturan Kamera akan mempengaruhi penampilan UI.



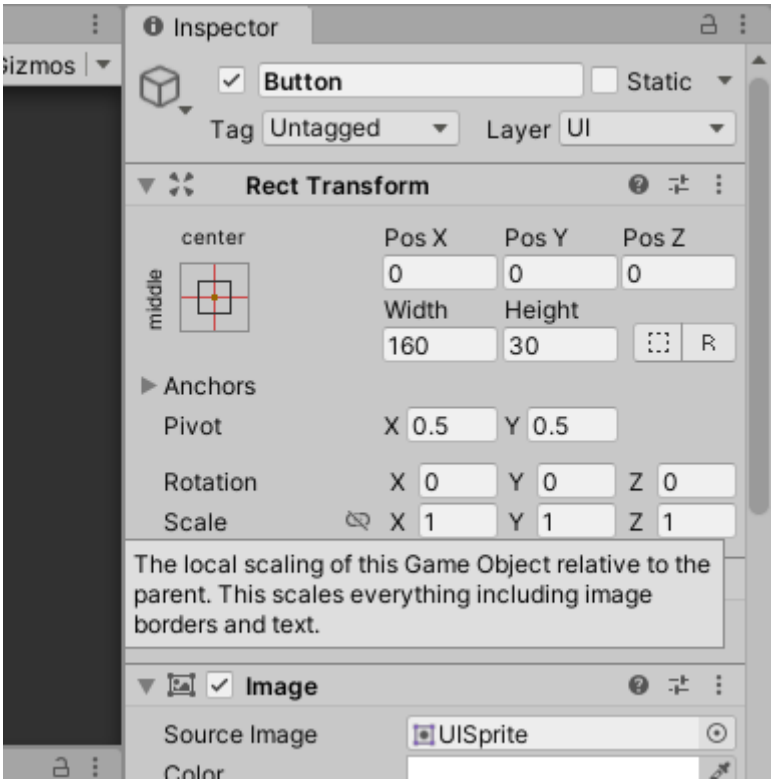
- **World Space**

  Dalam mode render in, *canvas* akan menjadi objek lain di dalam layar. Ukuran *canvas* dapat diatur secara manual menggunakan **Rect Transform** dan elemen UI akan ditampilkan di depan atau di belakang objek lain dalam *scene* berdasarkan penempatannya.

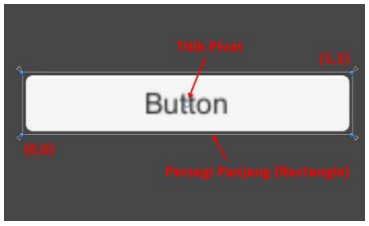
Berikut hasil tampilan dari **World Space**. Seolah-olah UI darah yang berada di atas objek **Sphere** menjadi bentuk tiga dimensi dan mengikuti setiap gerakan ke semua arah.

RectTransform

RectTransform adalah tata letak untuk mengatur bentuk objek 2D dari komponen *transform*. *Transform* merepresentasikan satu titik dan Rect Transform merepresentasikan persegi panjang di mana elemen UI dapat ditempatkan di dalamnya. Jika Rect Transform memiliki parent dan child, maka child berada di dalam parent. Child Rect Transform dapat menentukan posisi dan ukuran yang relatif terhadap persegi panjang parent Rect Transform.



Seperti gambar di bawah ini kita menggunakan **Button** sebagai contoh dari **Rect Transform**.

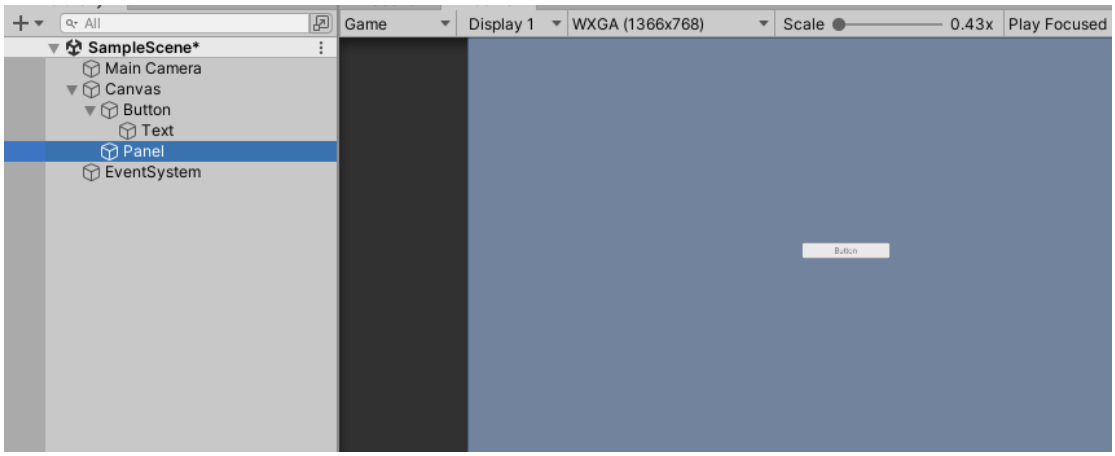


Berikut ini penjelasan fungsi masing-masing:

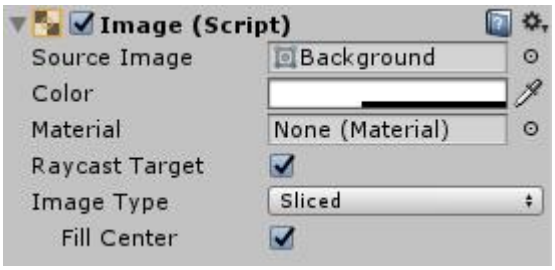
- **Pos (X, Y, Z)**: Posisi pivot point persegi panjang pada *anchor*.
- **Width/Height**: Lebar dan tinggi persegi panjang.
- **Anchor**: Titik *anchor* untuk sudut kiri bawah dan sudut kanan atas persegi panjang.
- **Min**: Titik *anchor* untuk sudut kiri bawah persegi panjang didefinisikan sebagai pecahan dari ukuran persegi panjang *parent*. Posisi 0,0 merepresentasikan sudut kiri bawah dari *parent*, sementara 1,1 berhubungan dengan sudut kanan atas dari *parent*.
- **Max**: Titik *anchor* untuk sudut kanan atas persegi panjang didefinisikan sebagai pecahan ukuran persegi panjang *parent*. 0,0 merepresentasikan sudut kiri bawah dari *parent*, sementara 1,1 berhubungan dengan sudut kanan atas dari *parent*.
- **Pivot**: Pivot point yang bentuknya bulat berwarna biru pada persegi panjang, yang didefinisikan sebagai pecahan ukuran persegi panjang itu sendiri. 0,0 sesuai dengan sudut kiri bawah sementara 1,1 sesuai dengan sudut kanan atas.
- **Rotation**: Sudut rotasi dari objek di sekitar titik pivot, mempunya sumbu X, Y, dan Z.
- **Scale**: Skala diterapkan pada objek dalam dimensi X, Y, dan Z.

Panel

Panel adalah adalah tempat untuk menyusun komponen GameObject secara grup. Kita dapat memanfaatkan Panel ini untuk membuat validasi, *grouping* komponen UI dan lainnya. Panel hampir serupa dengan Image, namun Panel secara *default* memiliki sumber gambar (*source image*) yang berupa Background. Kita dapat menambahkan Panel dengan cara klik kanan pada **Hierarchy > UI > Panel**.



Berikut tampilan komponen **Panel** atau **Image** pada tab **Inspector**.



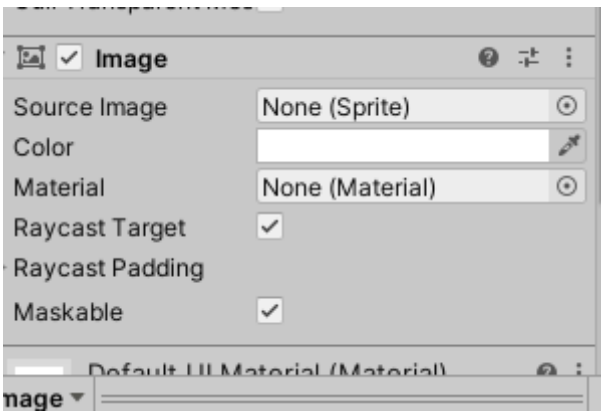
Berikut ini penjelasan fungsi masing-masing:

- **Source Image**: Tekstur yang mewakili gambar untuk ditampilkan (yang harus diimpor sebagai Sprite).
- **Color**: Warna untuk diterapkan pada gambar.
- **Material**: Bahan yang digunakan untuk rendering gambar.
- **Raycast Target**: Target untuk raycasting.
-

Image

Image adalah sebuah gambar yang tampil pada layar pengguna. Image dapat digunakan untuk hiasan, ikon, dan lainnya. Image dapat berasal dari script. Image di sini juga mirip dengan gambar mentah (raw image) yang bisa kita ubah posisi, ukuran, dan lainnya. Kita juga dapat mengubahnya menjadi gambar animasi yang bergerak.

Dengan tekstur, kita bisa mengubah baik image maupun raw image menjadi sprite (gambar UI 2D). Bedanya, jika image membutuhkan tekstur 2D, raw image apat menerima tekstur apapun. Kita dapat menambahkan Image dengan cara klik kanan pada **Hierarchy** > **UI** > **Image**. Berikut tampilan komponen **Image** pada tab **Inspector**.



Berikut ini penjelasan fungsi masing-masing:

- **Source Image**: Tekstur yang mewakili image untuk ditampilkan (yang harus diimpor sebagai Sprite).

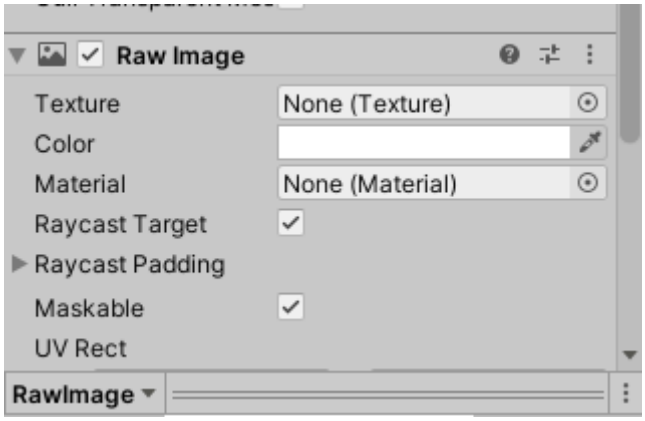
- **Color**: Warna untuk diterapkan pada image.
- **Material**: Bahan yang digunakan untuk rendering image.
- **Raycast Target**: Target untuk raycasting.

Berikut ini adalah contoh implementasi script untuk **Image** menggunakan Sprite yang dapat dilihat dan diubah pada Inspector sebagai bagian dari komponen gambar. Sprite juga dapat diubah menggunakan script. Berikut contoh kodenya.

```
.   using UnityEngine;
.   using UnityEngine.UI;
.
.   public class Images : MonoBehaviour {
.     Image m_Image;
.
.     // Atur pada Inspector
.     public Sprite m_Sprite;
.
.     void Start() {
.         // Ambil Gambar dari GameObject
.         m_Image = GetComponent<Image>();
.     }
.
.     void Update() {
.         // Tekan spasi untuk mengubah Sprite dari Gambar
.         if (Input.GetKey(KeyCode.Space)) {
.             m_Image.sprite = m_Sprite;
.         }
.     }
.   }
```

Raw Image

Raw Image adalah gambar non-interaktif untuk hiasan, ikon, dan lainnya. Sementara image adalah gambar yang biasa digunakan untuk Sprite, Background dan lainnya. **Raw Image** dan **Image** hampir sama penggunaannya. Kita dapat menambahkan Raw Image dengan cara klik kanan pada **Hierarchy > UI > Raw Image**. Berikut tampilan komponen **Raw Image** pada tab **Inspector**.



Berikut ini penjelasan fungsi masing-masing:

- **Texture**: Tekstur yang mewakili gambar untuk ditampilkan.
- **Color**: Warna untuk diterapkan pada gambar.
- **Material**: Bahan yang digunakan untuk rendering gambar.
- **Raycast Target**: Target untuk raycasting.
- **UV Rect**: Offset dan ukuran gambar dalam mengontrol persegi panjang. Kita memberikan UV Rect dalam koordinat yang dinormalkan (rentang 0,0 hingga 1,0). Kita merentangkan tepi-tepi gambar untuk mengisi ruang di sekitaran persegi panjang UV.

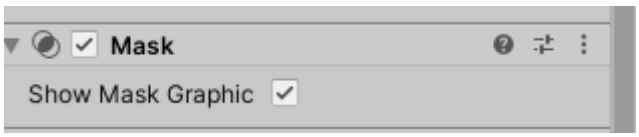
Berikut ini adalah contoh implementasi script untuk **Raw Image**, yang digunakan untuk mengubah atau mengembalikan tampilan Texture Raw Image. Raw Image dapat menampilkan Tekstur (texture) apapun, sedangkan komponen Image hanya dapat menampilkan Tekstur Sprite, berikut contoh kodenya.

```
.   using UnityEngine;
.   using UnityEngine.UI;
.
.   public class RawImages : MonoBehaviour {
.       RawImage m_RawImage;
.
.       // Pilih Tekstur pada Inspector untuk diubah menjadi gambar
.       public Texture m_Texture;
.
.       void Start() {
.           // Ambil komponen Raw Image dari GameObject
.           m_RawImage = GetComponent<RawImage>();
.
.           // Ubah Tekstur yang Anda tentukan di Inspector
.           m_RawImage.texture = m_Texture;
.       }
.   }
```

Mask

Mask bukanlah UI yang terlihat, melainkan cara memodifikasi tampilan elemen turunan dari child. Mask membatasi (yaitu, "**mask**") elemen child ke bentuk parent. Jadi, jika child lebih besar dari parent, maka hanya bagian child yang cocok dalam parent. Sebelum menambahkan Mask pada Inspector, kita tambahkan GameObject Image terlebih dahulu pad Hierarchy dengan cara klik kanan pada **Hierarchy > UI > Image**. Lalu kita dapat menambahkan Mask dengan cara pada bagian tab **Inspector > klik "Add Component" > ketik "Mask."**

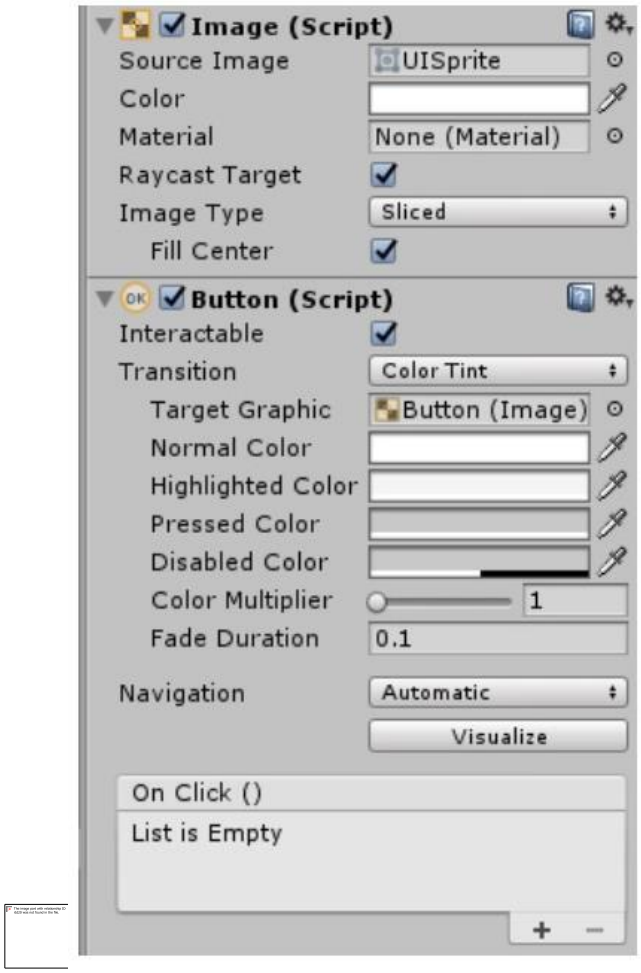
Berikut tampilan komponen **Mask** pada tab **Inspector**.



Kita dapat menyusun Mask yang telah dibuat dengan cara menambahkan dua GameObject yaitu **Image** dan **Panel**. Yang mana **Image** adalah gambar dan **Panel** adalah Background yang dijadikan **Mask**. Kita dapat merubah bentuk **Mask** yang diinginkan dengan cara seperti ini.

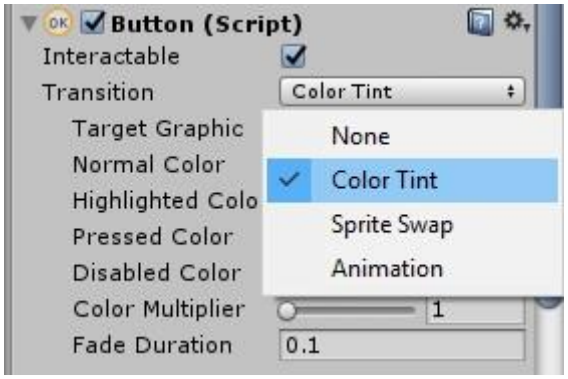
Button

Button adalah sebuah tombol yang merespon klik dari pengguna dan digunakan untuk memulai atau mengkonfirmasi suatu tindakan. Kita dapat menambahkan Button dengan cara klik kanan pada **Hierarchy > UI > Button**. Berikut tampilan komponen **Button** pada tab **Inspector**.



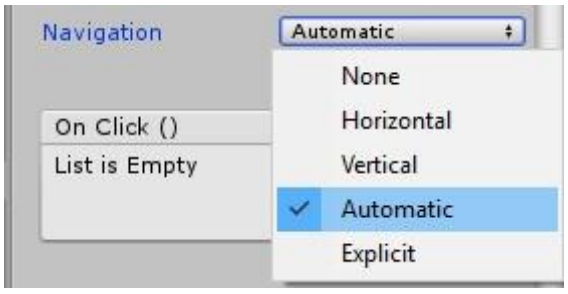
Berikut ini penjelasan fungsi masing-masing :

- **Source Image**: Sebuah gambar tombol yang dapat kita ubah bentuknya. Kita juga dapat merubah teks serta warna.
- **Interactable**: Ini menentukan apakah komponen akan menerima masukkan. Ketika diset nonaktif (tidak dicentang), maka ia tidak akan menerima masukkan, begitu pun sebaliknya.
- **Transition**: Di dalam transisi terdapat 4 komponen, **None**, **Color Tint**, **Sprite Swap** dan **Animation**.



- **None**: Tombol tidak memiliki efek sama sekali.
- **Color Tint**: Mengubah warna tombol tergantung pada keadaan saat tombol ditekan. Tujuannya untuk memilih warna dan durasi pudarnya warna (*faded*). Semakin tinggi angkanya, semakin lambat warna yang memudar.
- **Sprite Swap**: Menampilkan sprite (gambar UI) tergantung pada status tombol. Sprite juga dapat dikustomisasi.
- **Animation**: Membuat animasi tergantung pada keadaan saat tombol ditekan. Komponen animator harus ada pada transisi animasi. Untuk membuat pengontrol animasi, klik untuk membuat animasi . Lalu pastikan bahwa pengontrol animasi telah ditambahkan ke komponen animator tombol.
- **Transition**: Di dalam sub komponen transisi juga terdapat 4 pilihan transisi lainnya, tergantung pada kondisi apa yang dipilih. Keadaan yang berbeda adalah, *Normal* (normal), *Highlighted* (disorot), *Pressed* (ditekan) dan *Disabled* (dinonaktifkan).
- **Normal Color**: Kondisi tombol dalam keadaan normal (*default*).
- **Highlighted Color**: Kondisi saat kursor diarahkan pada tombol.
- **Pressed Color**: Kondisi saat tombol ditekan.

- **Navigation**:



- **None**: Tidak ada navigasi.
- **Horizontal**: Menavigasi secara horizontal.

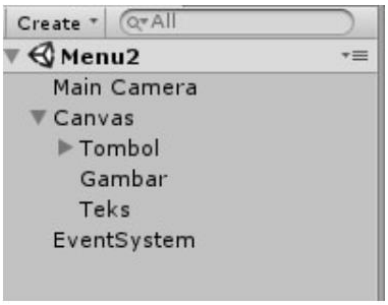
- **Vertical**: Menavigasi secara vertikal.
- **Automatic**: Navigasi otomatis.
- **Explicit**: Dalam mode ini Anda dapat secara eksplisit menentukan letak kontrol pada saat menavigasi ke tombol panah yang berbeda.
- **On Click ()**: Anda dapat memasukkan script yang berisi saat tombol ditekan atau menerima aksi (inputan).

Button akan berefek saat ditekan. Berikut ini adalah contoh implementasi *script* untuk botton tersebut.

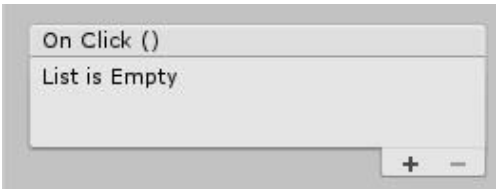
```
using UnityEngine;
using UnityEngine.UI;

public class Button : MonoBehaviour {
   void CobaKlik() {
       // Output ini akan tampil ketika Tombol diklik
       Debug.Log("Dicoding Indonesia");
   }

   void KlikDenganParameter (string message) {
       // Output ini akan tampil ketika Tombol diklik
       Debug.Log(message);
   }
}
```

Event System

Event System adalah sistem yang menerima *event* dari Event Trigger dan memanggil fungsi yang telah diinisialisasi untuk Event System. Event System akan otomatis ditambahkan saat kita memasukkan UI baru. Event Trigger dapat digunakan untuk menentukan fungsi yang Anda panggil untuk setiap peristiwa Event System.
Sebagai contoh, berikut ini ada beberapa objek yaitu gambar, teks dan teks akan dimasukkan ke Event System. Kita dapat menambahkan Event System, dengan cara klik kanan pada **Hierarchy > UI > Event System**. Berikut tampilan komponen yang harus dibuat dan disusun pada tab **Hierarchy**.



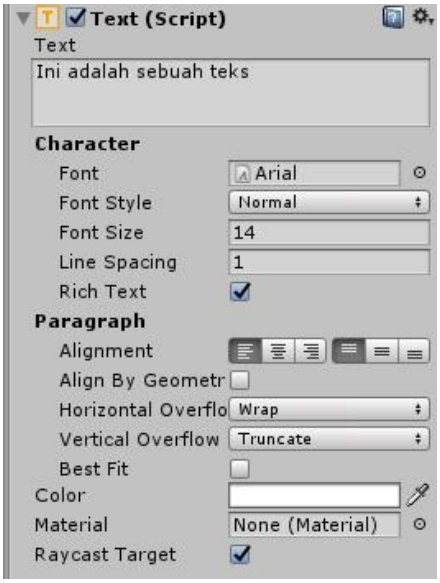
Tombol, gambar, dan teks tidak dimasukkan *script* pada list **On Click ()**.



Kita dapat memasukkan objek tombol ke dalam **Event System > First Selected.** Lalu kita dapat menekan tombol saat di Play Mode. Kita juga dapat menambahkan Event Trigger pada objek gambar. Kemudian kita **Add New Event Type > Pointer Enter** yang nantinya akan dimasukkan *script* untuk memindahkan *scene* dari menu 2 ke menu 1 . Caranya, arahkan pointer ke gambar.

Text

Text adalah sebuah potongan teks yang tampil pada layar pengguna. Teks memberikan keterangan atau label untuk GUI (Graphic User Interface) atau menampilkan instruksi. Kita juga dapat mengatur jenis, ukuran, posisi, dan warna teks. Kita dapat menambahkan teks dengan cara klik kanan pada **Hierarchy > UI > Text**. Berikut tampilan komponen **Text** pada tab **Inspector**.



Berikut ini penjelasan fungsi masing-masing:

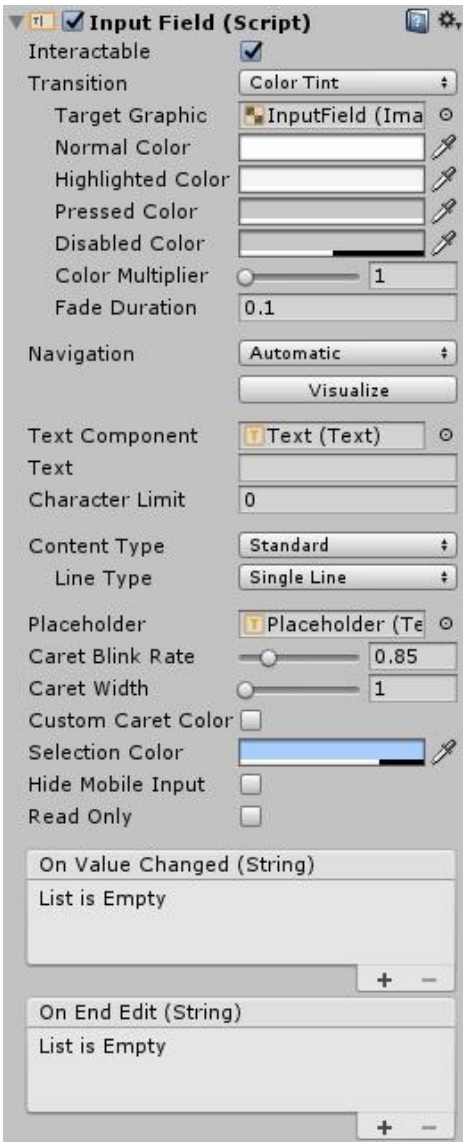
- **Text**: Tempat di mana kita menginputkan tulisan.
- **Font**: Jenis teks yang tampil.
- **Font Style**: Gaya yang diterapkan pada teks. Pilihannya antara Normal, Bold (tebal), Italic (miring) serta Bold-Italic (tebal dan miring).
- **Font Size**: Ukuran teks yang ditampilkan.
- **Line Spacing**: Pemisahan vertikal antara baris-baris teks.
- **Rich Text**: Elemen pengayaan teks.
- **Alignment**: Perataan teks secara horizontal dan vertikal.
- **Align By Geometry**: Penggunaan luasan geometri untuk mensejajarkan secara horizontal.
- **Horizontal Overflow**: Metode untuk menangani situasi di mana teks terlalu lebar untuk masuk ke dalam persegi panjang. Opsinya adalah Wrap and Overflow.
- **Vertical Overflow**: Metode untuk menangani situasi di mana teks yang terlalu tinggi untuk masuk ke dalam persegi panjang. Pilihannya adalah Truncate and Overflow.
- **Best Fit**: Unity mengabaikan properti ukuran dan cukup menyesuaikan teks ke persegi panjang.
- **Color**: Warna yang digunakan untuk me-render teks.
- **Material**: Bahan yang digunakan untuk me-render teks.
- **Raycast Target**: Target untuk raycasting.

Berikut ini adalah contoh implementasi script untuk teks. Ini adalah nilai string dari komponen Text. Gunanya untuk membaca atau mengedit pesan yang ditampilkan pada Teks. Contoh kodenya seperti di bawah ini

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

public class Texts : MonoBehaviour {
   public static int score;

   Text text;

   void Awake() {
       // Mengatur reference teks
       text = GetComponent <Text> ();

       // Mengembalikkan nilai awal skor
       score = 0;
   }

   void Update() {
        // Jumlah skor yang ditampilkan
        text.text = "Score: " + score;
   }
}
```

Input Field

Input Field adalah cara untuk membuat Kontrol Teks yang dapat diubah. Di dalamnya kita dapat menginput huruf, nomor, dan simbol. Kita dapat menambahkan Input Field dengan cara klik kanan pada **Hierarchy** > **UI** > **Input Field**. Berikut tampilan komponen **Input Field** pada tab **Inspector**.



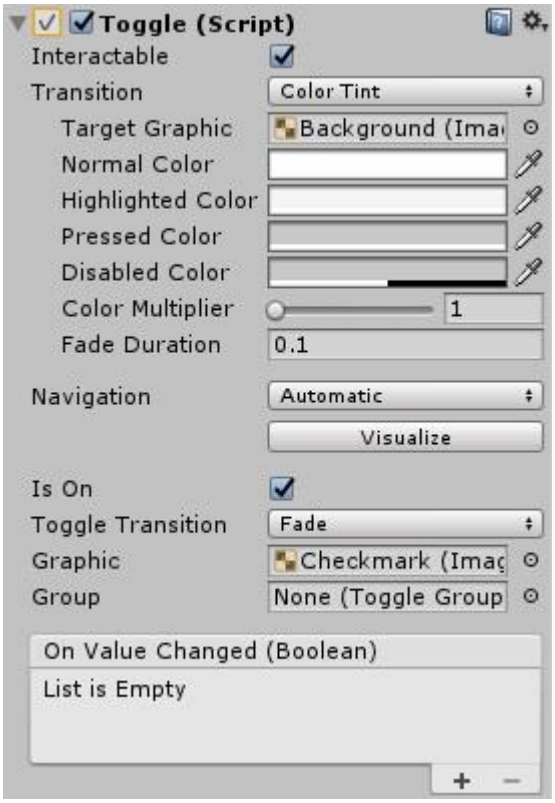
Berikut ini penjelasan fungsi masing-masing:

- **Interactable**: Ini menentukan apakah komponen akan menerima masukan. Ketika disetel nonaktif (tidak dicentang), maka tidak akan menerima masukkan, sebaliknya pun begitu.
- **Transition**: Digunakan untuk mengatur bagaimana transisi field input saat Normal, Highlighted, ditekan atau dinonaktifkan.

- **Navigation**: Properti yang menentukan urutan kontrol.
- **Text Component**: Referensi ke elemen teks yang digunakan sebagai isi dari Field Input.
- **Text**: Kolom teks gunanya nanti akan diisikan sebuah teks atau nilai.
- **Character Limit**: Nilai jumlah karakter maksimum yang dapat dimasukkan ke dalam kolom input.
- **Content Type**: Penentuan jenis karakter yang diterima oleh field input.
- **Line Type**: Menentukan bagaimana teks diformat di dalam bidang teks.
- **Placeholder**: Ini adalah grafik 'kosong' yang bersifat opsional untuk menunjukkan Bidang Masukan kosong teks.
- **Caret Blink Rate**: Menentukan tingkat kedipan untuk tanda yang ditempatkan pada garis. Gunanya untuk menunjukkan penyisipan teks.
- **Selection Color**: Warna pilihan latar belakang bagian teks.
- **Hide Mobile Input**: Sembunyikan bidang masukan asli yang melekat pada keyboard di layar pada perangkat seluler. Ini hanya berfungsi di perangkat iOS.
- **On Value Change ()**: UnityEvent yang dipanggil ketika konten teks dari Input Field berubah. Dapat mengirim konten teks sebagai argumen dinamis dengan tipe string.
- **On End Edit ()**: UnityEvent yang dipanggil ketika pengguna selesai mengedit konten teks baik dengan mengirimkan maupun mengklik suatu tempat yang menghilangkan fokus dari Input Field.

Toggle

Toggle adalah kotak centang yang memungkinkan pengguna mengaktif / non-aktifkan opsi. Kita dapat menambahkan Toggle dengan cara klik kanan pada **Hierarchy > UI > Toggle**. Berikut tampilan komponen **Toggle** pada tab **Inspector**.



Berikut ini penjelasan fungsi masing-masing:

- **Interactable**: menentukan apakah komponen akan menerima masukan. Ketika disetel non-aktif (tidak dicentang), maka tidak akan menerima masukkan, begitu sebaliknya.
- **Transition**: menentukan cara kontrol merespon tindakan pengguna secara visual.
- **Navigation**: menentukan urutan kontrol.
- **Is On**: mengaktif / non-aktifkan toggle.
- **Toggle Transition**: Cara toggle bereaksi secara grafis ketika nilainya diubah. Pilihannya adalah None (yaitu, tanda centang hanya muncul atau menghilang) dan Fade (yaitu, tanda centang memudar masuk atau keluar).
- **Graphic**: Gambar yang digunakan untuk tanda centang.
- **Group**: Toggle Group (jika ada) tempat Toggle ini berada.

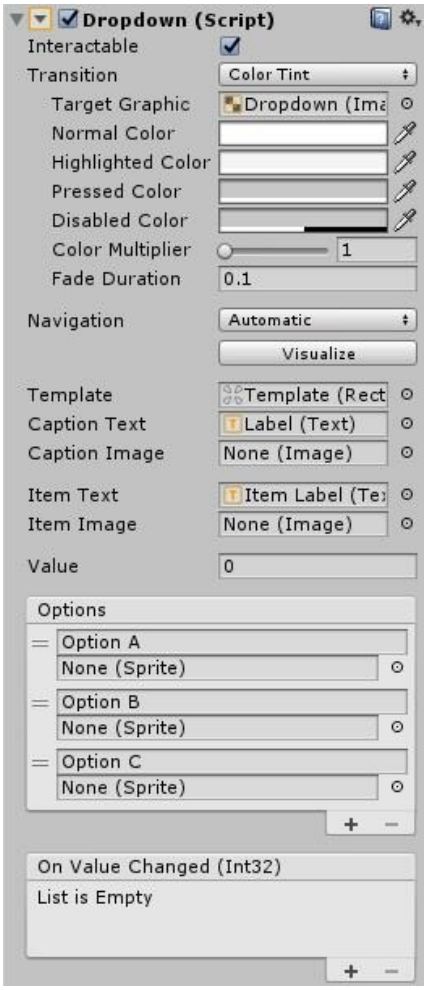
Berikut ini adalah contoh implementasi *script* untuk Toggle. S*cript* ini mengembalikan atau mengatur apakah Toggle aktif / tidak.

```
.   using UnityEngine;
```

```
.   using UnityEngine.UI;
.
.   public class Toogles : MonoBehaviour {
.       Toggle m_Toggle;
.       public Text m_Text;
.
.       void Start() {
.           // Ambil Toggle GameObject
.           m_Toggle = GetComponent<Toggle>();
.
.           // Inisialisasi Teks untuk mengecek apakah Toggle dalam keadaan positif (on) atau negatif
    (off)
.           m_Text.text = "Toggle dalam keadaan: " + m_Toggle.isOn;
.       }
.   }
```

Dropdown

Dropdown adalah komponen yang memungkinkan pengguna memilih satu opsi dari daftar. Setelah diklik, maka daftar opsi akan terbuka sehingga opsi baru dapat dipilih. Setelah memilih opsi baru, daftar ditutup lagi, dan kontrol menunjukkan opsi yang baru dipilih. Kita dapat menambahkan Dropdown dengan cara klik kanan pada **Hierarchy > UI > Dropdown**.



Berikut ini adalah penjelasan fungsi masing-masing:

- **Interactable**: menentukan apakah komponen akan menerima masukan. Ketika disetel nonaktif (tidak dicentang), maka tidak akan menerima masukkan, begitu sebaliknya.
- **Transition**: menentukan cara kontrol merespon tindakan pengguna secara visual.
- **Navigation**: menentukan urutan kontrol.
- **Template**: The Rect Transform dari template untuk daftar dropdown.
- **Caption Text**: Komponen Text untuk menahan teks dari opsi yang sedang dipilih.
- **Caption Image**: Komponen gambar untuk menyimpan gambar pilihan.
- **Item Text**: Komponen Text untuk memegang teks item.

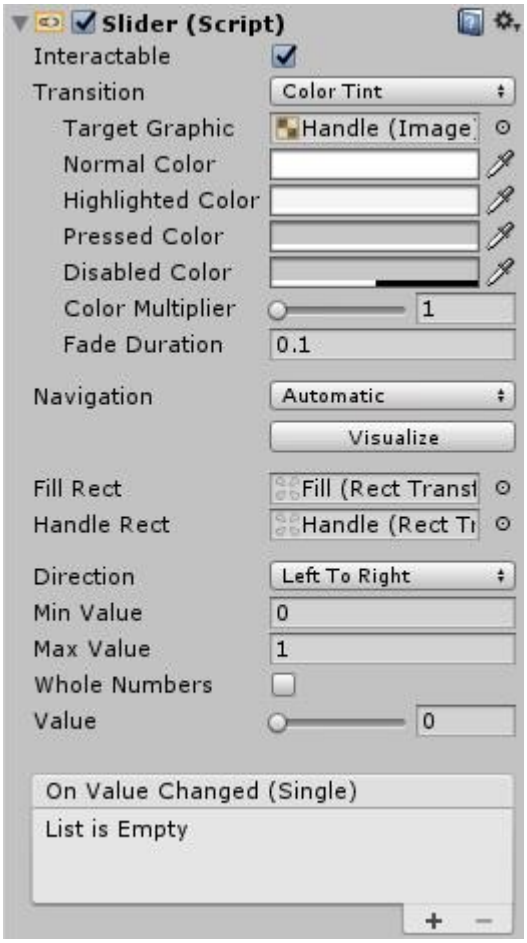
- **Item Image**: Komponen Gambar untuk menyimpan gambar item.
- **Value**: Indeks opsi yang saat ini dipilih. Nol (0) adalah opsi pertama, satu (1) yang kedua, dan seterusnya.
- **Options**: Daftar teks dan gambar dapat ditentukan untuk setiap opsi.

Berikut ini adalah contoh implementasi *script* untuk Dropdown, yakni tambahan beberapa opsi teks ke opsi Dropdown berdasarkan daftar string, berikut contoh *code* nya:

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

public class Dropdowns : MonoBehaviour {
    // Buat Daftar opsi Dropdown baru
    List<string> m_DropOptions = new List<string> {"Opsi 1", "Opsi 2", "Opsi 3"};

    // Ini adalah Dropdown
    Dropdown m_Dropdown;

    void Start() {
        // Ambil GameObject Dropdown yang dilampirkan pada script
        m_Dropdown = GetComponent<Dropdown>();

        // Tambahkan opsi yang dibuat pada Daftar
        m_Dropdown.AddOptions(m_DropOptions);
    }
}
```

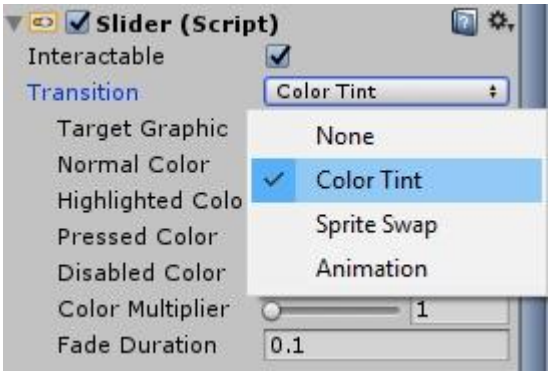
Slider

Slider adalah sebuah nilai numerik dari rentang yang telah ditentukan. Pengguna dapat men-drag mouse untuk memilih nilai numerik tersebut. Kita dapat menambahkan Slider, dengan cara klik kanan pada **Hierarchy** > **UI** > **Slider**. Berikut tampilan komponen **Slider** pada tab **Inspector**.

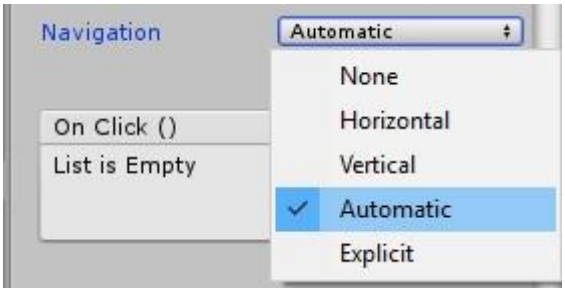


Berikut ini penjelasan fungsi masing-masing:

- **Interactable**: menentukan apakah komponen akan menerima masukan. Ketika disetel nonaktif (tidak dicentang), maka tidak akan menerima masukkan, begitu sebaliknya.
- **Transition**: Di dalam komponen transisi terdapat 4 pilihan transisi tergantung pada pilihan kondisi. Keadaan yang berbeda adalah, *Normal* (normal), *Highlighted* (disorot), *Pressed* (ditekan) dan *Disabled* (dinon-aktifkan). Di dalam transisi terdapat 4 komponen lainnya, yaitu.



- **None**: Tombol tidak memiliki efek keadaan sama sekali.
- **Color Tint**: Mengubah warna tombol tergantung pada keadaan saat tombol ditekan. Tujuannya untuk memillih warna sesuai selera serta mengatur durasi pudarnya warna (*faded*). Semakin tinggi angkanya, semakin lambat warna yang memudar.
- **Sprite Swap**: Memungkinkan sprite (gambar UI) untuk tampil tergantung pada status tombol, sprite juga dapat dikustomisasi.
- **Animation**: Memungkinkan animasi terjadi tergantung pada keadaan tombol. Komponen animator harus ada untuk digunakan pada transisi animasi. Untuk membuat pengontrol animasi, klik untuk membuat animasi (atau buat sendiri) dan pastikan bahwa pengontrol animasi telah ditambahkan ke komponen animator tombol.
- **Navigation**:



- **None**: Tidak ada navigasi.
- **Horizontal**: Menavigasi secara Horizontal.
- **Vertical**: Menavigasi secara Vertikal.
- **Automatic**: Navigasi otomatis.
- **Explicit**: Dalam mode ini kamu dapat secara eksplisit (jelas) menentukan dimana kontrol menavigasi ke tombol panah yang berbeda.
- **Fill Rect**: Grafik yang digunakan untuk mengisi area kontrol.
- **Handle Rect**: Grafik yang digunakan untuk bagian "*handle*" untuk menggeser.
- **Direction**: Yang mana nilai arah dari slider akan meningkat ketika *handle* diseret Pilihannya adalah **Kiri ke Kanan**, **Kanan ke Kiri**, **Bawah ke Atas** dan **Atas ke Bawah**.
- **Min Value**: Nilai penggeser saat *handle* berada di ujung bawah atau paling kiri (ditentukan oleh arah).
- **Max Value**: Nilai penggeser saat *handle* berada di ujung atas atau paling kanan (ditentukan oleh arah).

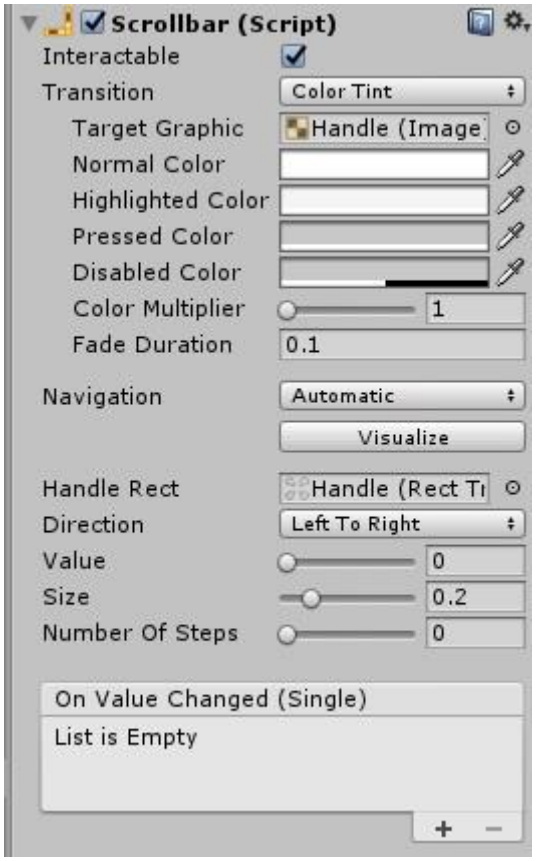
- **Whole Numbers**: Slider harus dibatasi ke nilai integer.
- **Value**: Nilai numerik dari slider. Jika nilai ditetapkan di **Inspector** akan digunakan sebagai nilai awal, tetapi akan berubah pada saat dijalankan (*play mode*).

Adapun contoh implementasi *script* untuk Slider yang memiliki nilai, berikut contoh *code* nya:

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

public class Sliders : MonoBehaviour {
    public Slider mainSlider;

    // Dipanggil ketika tombol submit diklik
    public void CobaSlider() {
        // Menampilkan nilai pada console
        Debug.Log(mainSlider.value);
    }
}
```

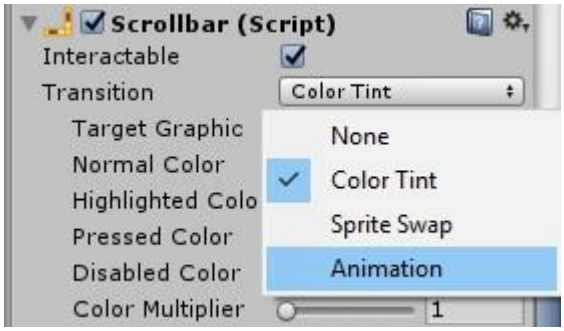
Scrollbar

Scrollbar adalah sebuah antarmuka yang memungkinkan pengguna untuk menggulir (*scroll*) gambar atau tampilan lain yang terlalu besar untuk dilihat. Contoh yang biasa digunakan pada Scrollbar adalah bar vertikal dan horizontal untuk melihat bagian dari gambar. Kita dapat menambahkan Scrollbar, dengan cara klik kanan pada **Hierarchy > UI > Scrollbar**. Berikut tampilan komponen **Scrollbar** pada tab **Inspector**.



Berikut ini penjelasan fungsi masing-masing:

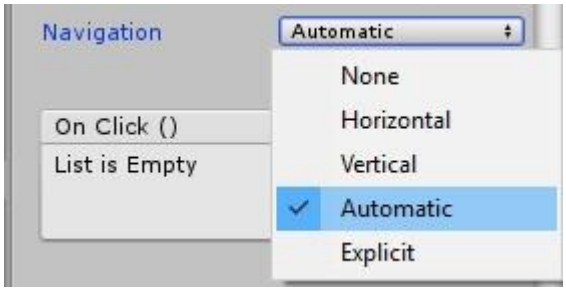
- **Interactable**: Ini menentukan apakah komponen akan menerima masukan. Ketika disetel nonaktif (tidak dicentang), maka tidak akan menerima masukkan, sebaliknya pun begitu.
- **Transition**: Didalam komponen transisi terdapat 4 pilihan transisi tergantung pada kondisi apa yang dipilih. Keadaan yang berbeda adalah: *Normal* (normal), *Highlighted* (disorot), *Pressed* (ditekan) dan *Disabled* (dinonaktifkan). Dan didalam transisi terdapat 4 komponen lainnya, yaitu.



- **None**: Tombol tidak memiliki efek keadaan sama sekali.
- **Color Tint**: Mengubah warna tombol tergantung pada keadaan saat tombol ditekan. Hal ini dimungkinkan untuk memilih warna untuk disesuaikan. Juga

dimungkinkan untuk mengatur durasi pudarnya warna (*faded*). Semakin tinggi angkanya, semakin lambat warna yang memudar.

- **Sprite Swap**: Memungkinkan sprite (gambar UI) untuk ditampilkan tergantung pada status tombol, sprite juga dapat dikustomisasi.
- **Animation**: Memungkinkan animasi terjadi tergantung pada keadaan tombol, komponen animator harus ada untuk digunakan pada transisi animasi. Untuk membuat pengontrol animasi, klik untuk membuat animasi (atau buat sendiri) dan pastikan bahwa pengontrol animasi telah ditambahkan ke komponen animator tombol.
- **Navigation**:



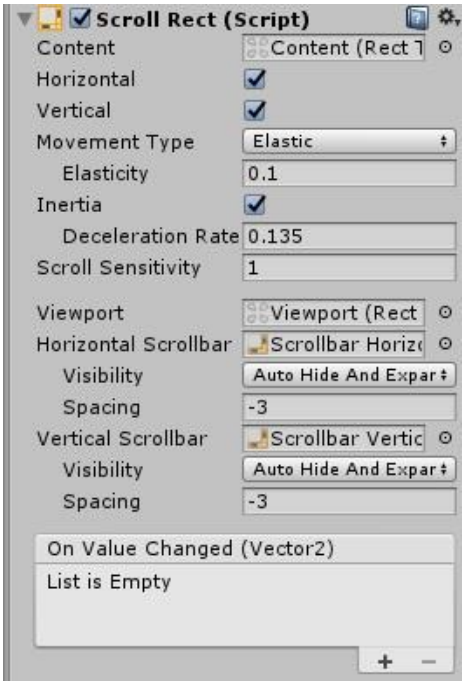
- **None**: Tidak ada navigasi.
- **Horizontal**: Menavigasi secara Horizontal.
- **Vertical**: Menavigasi secara Vertikal.
- **Automatic**: Navigasi otomatis.
- **Explicit**: Dalam mode ini kamu dapat secara eksplisit (jelas) menentukan dimana kontrol menavigasi ke tombol panah yang berbeda.
- **Fill Rect**: Grafik yang digunakan untuk mengisi area kontrol.
- **Handle Rect**: Grafik yang digunakan untuk bagian "*handle*" untuk menggeser.
- **Direction**: Yang mana nilai arah dari slider akan meningkat ketika *handle* diseret. Pilihannya adalah **Kiri ke Kanan**, **Kanan ke Kiri**, **Bawah ke Atas** dan **Atas ke Bawah**.
- **Value**: Nilai numerik dari slider. Jika nilai ditetapkan di **Inspector** akan digunakan sebagai nilai awal, tetapi akan berubah pada saat dijalankan (*play mode*).
- **Size**: Ukuran pecahan *handle* dalam Scrollbar, dalam rentang 0,0 hingga 1,0.
- **Number Of Steps**: Jumlah posisi *scroll* yang berbeda pada Scrollbar.

Adapun contoh implementasi *script* untuk Scrollbar, posisi *handle* vertikal sebagai nilai antara 0 dan 1, dengan 0 berada di bagian bawah dan 1 berada di bagian atas, berikut contoh *code* nya:

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

public class Scrollbars : MonoBehaviour {
    public ScrollRect newScrollRect;
    public Scrollbar newScrollBar;

    public void Start() {
        // Ubah posisi Handle vertikal
        newScrollRect.verticalNormalizedPosition = 0.5f;
    }
}
```

Scroll View

Scroll View adalah sebuah konten yang membutuhkan banyak ruang untuk ditampilkan di area yang kecil. Scroll View juga menyediakan fungsi untuk mengemas konten yang ada. Scroll View juga dapat dikombinasikan dengan satu atau dua Scrollbar yang dapat diseret untuk menggeser secara **Horizontal** atau **Vertikal**. Kita dapat menambahkan Scroll View, dengan cara klik kanan pada **Hierarchy** > **UI** > **Scroll View**. Berikut tampilan komponen **Scroll View** atau **Scroll Rect** pada tab **Inspector**.



Berikut ini penjelasan fungsi masing-masing:

- **Content**: Ini adalah *reference* ke RectTransform dari elemen UI yang akan digulir, misalnya gambar dengan skala yang besar.
- **Horizontal**: Menggeser secara horizontal.
- **Vertical**: Menggeser secara vertikal.
- **Movement Type**: Tidak terbatas, elastis atau dijepit. Gunakan **Elastic** atau **Clamped** untuk memaksa konten tetap berada di dalam batas-batas area ScrollRect.
- **Elasticity**: Ini adalah jumlah pantulan yang digunakan dalam mode elastisitas.
- **Scroll Sentivity**: Sensitivitas untuk *scroll* dan trackpad.
- **Viewport**: *Reference* ke *viewport* RectTransform yang merupakan *parent* dari konten RectTransform.
- **Horizontal Scrollbar**: *Reference* opsional untuk elemen scrollbar horizontal.
- **Visibility**: Scrollbar secara otomatis akan disembunyikan saat tidak diperlukan, dan secara opsional akan memperluas viewport juga.
- **Spacing**: Ruang antara *scroll* dan *viewport.*
- **Vertical Scrollbar**: *Reference* opsional untuk elemen scrollbar vertikal.

III. Alat dan Bahan

1. PC / Laptop

IV. Materi

A. Menyusun Tampilan Game

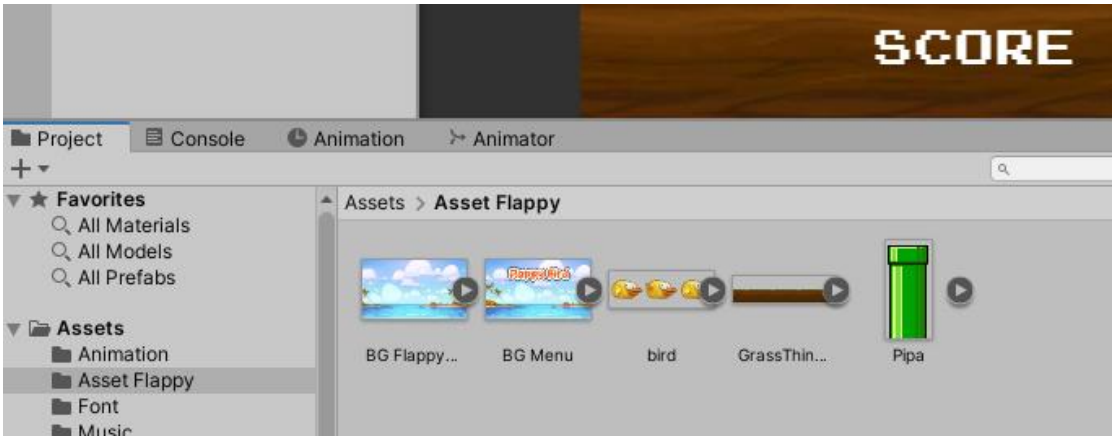
Pertama kali sebelum memulai menyusun tampilan game, kita harus membuat project baru di Unity. Langkah-langkahnya antara lain:

1. Buka **Unity.**
2. Klik **New** atau klik menu **File** > **New Project.**
3. Masukkan nama proyek yang diinginkan, yakni **“FlappyBird”**
4. Pilih atau centang pada **2D**.
5. Pada field **Location** pilih tempat di mana proyek tersebut akan disimpan.
6. Klik tombol “**Create project.**”

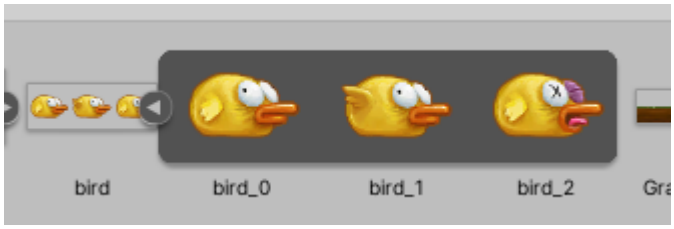
Pengaturan Sprite

1. Setelah sebelumnya Anda membuat project baru Unity, selanjutnya pada Panel Project buat folder baru dengan cara **Klik kanan di dalam folder Assets** > Pilih **Create** > Pilih **Folder**. Kemudian buat folder baru dengan nama **Sprites** . Sehingga folder yang sudah dibuat pada Panel Projects akan terlihat seperti gambar di bawah ini :

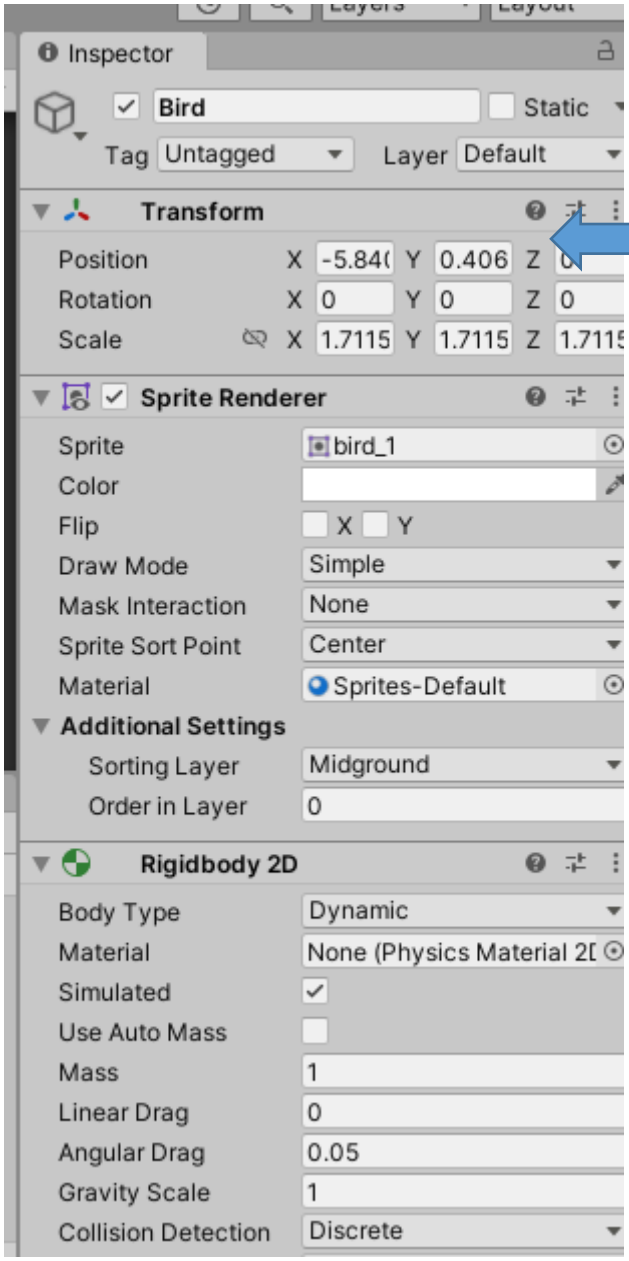
2. Selanjutnya masukkan Asset ke dalam folder yang sudah dibuat pada Panel Project di Unity.:



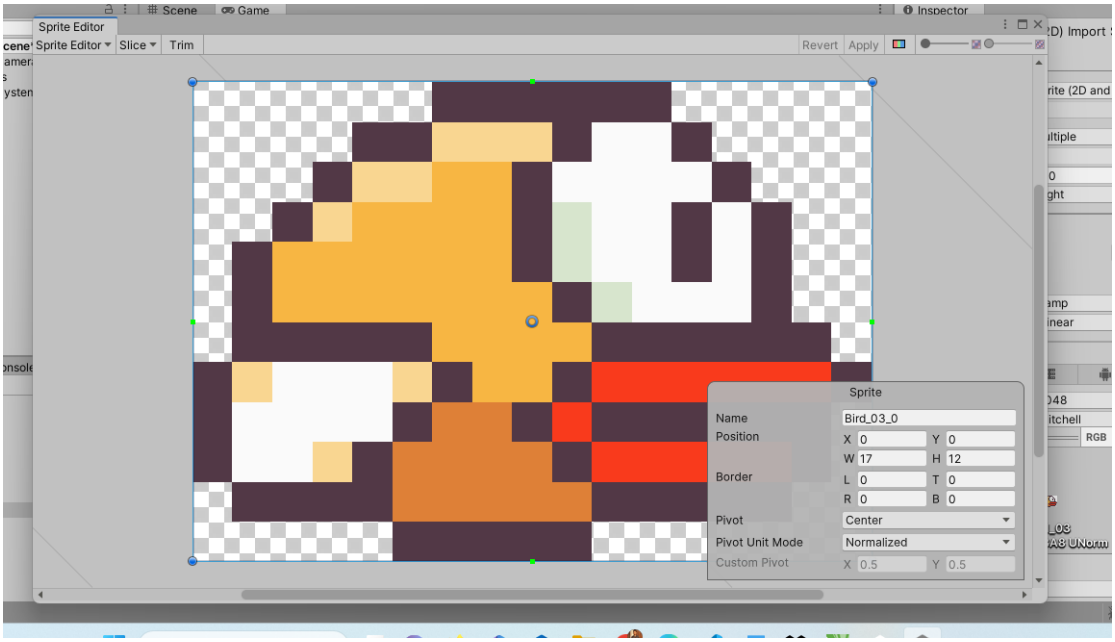
3. Dimulai dengan BirdHero,di dalam terdapat 3 sprites dalam 1 file. Seperti di bawah ini :



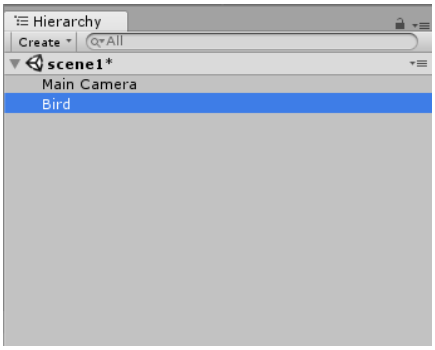
4. Klik BirdHero dan masuk ke panel Inspector, Ubah Sprite Mode menjadi Multiple lalu klik Apply dan masuk ke Sprite editorkita ubah mejadi



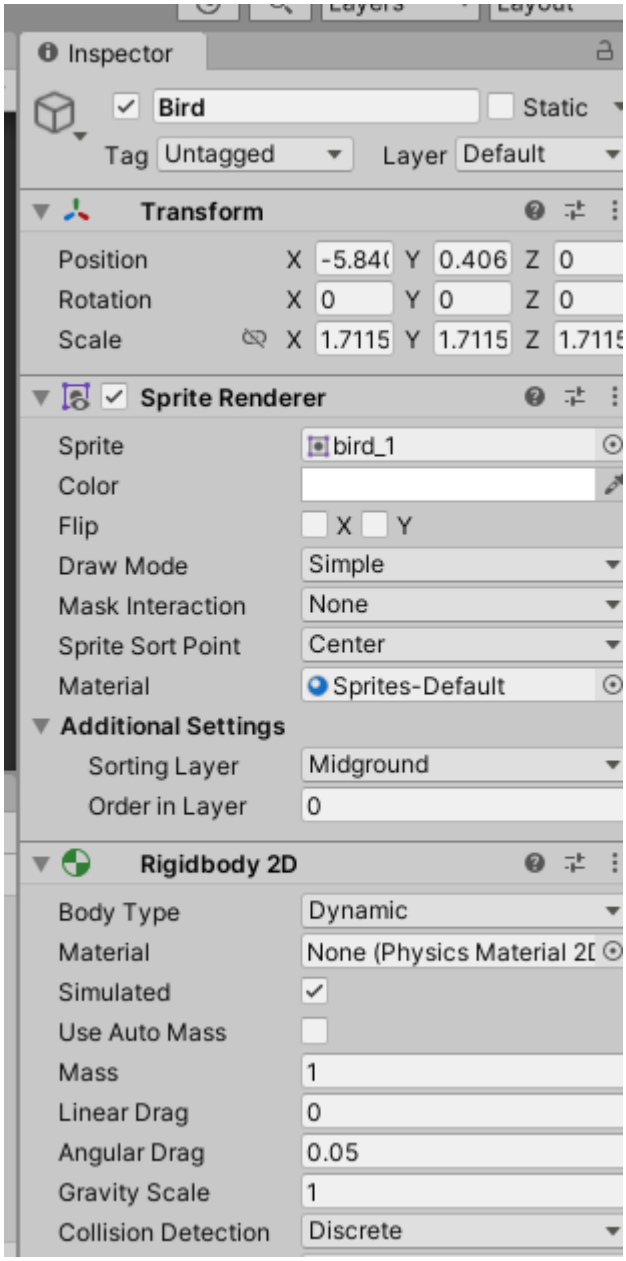
5. Lakukan slice pada Bird Hero



6. Masukkan BirdHero_0 kedalan Hierarki dan ubah namanya menjadi Bird



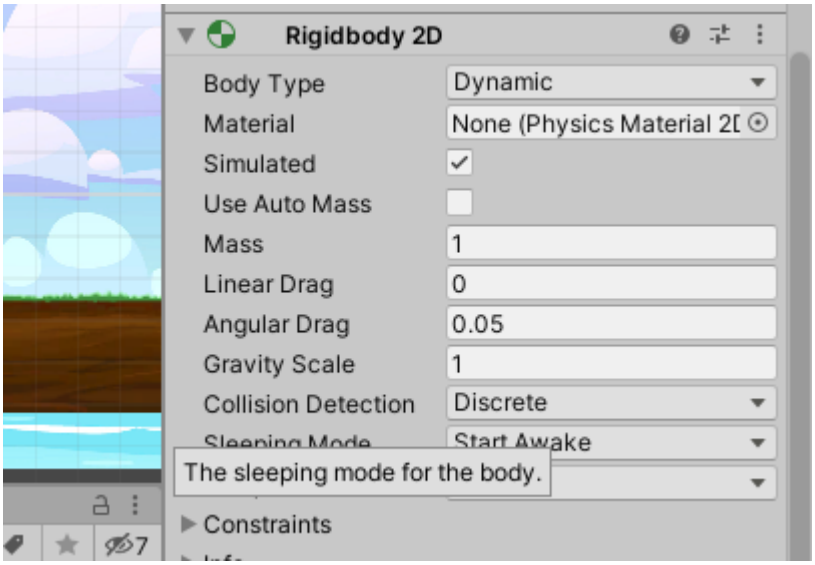
7. Tambahkan Rigidbody 2D (Add Component-->Physic 2D-->Rigidbody 2D) dan Tambahkan Polygon Collider 2D(Add Component-->Physic 2D-->Polygon Collider 2D). Sehingga menjadi :



8. Masukkan juga GrassThinSprite ke Hierarki lalu ubag namanya menjadi Ground dan atur posisinya berada di bawah, seperti gambar di bawah ini

9. Tambahkan Box Collider 2D (Add Component-->Physic 2D-->Box Collider2D), akan muncul box hijau pada sprites Ground, atur ukuran box tersebut sehingga fit dengan sprite Ground.

10. Tambahkan Box Rigidbody2D (Add Component-->Physic 2D-->Rigidbody 2D), lalu ubah Body Type menjadi Kinematic.



11. Lalu Masukkan file SkyTileSprite ke Herarki menjadi child dari Ground, Setelah itu kita atur SkyTileSprite menjadi background layer dari scene ini, Dengan Cara klik SkyTileSprite lalu pada panel Inspector --> Sorting Layer--> Add Sorting Layer

Selanjutnya ubah setting Sorting Layer SkyTileSprite menjadi Background, Ground menjadi Foreground, dan Bird menjadi Foreground.

Pengaturan Scripts

1. Buat Folder Scripts pada Asset

2. Buat Scripts baru (Klik kanan --> create-->C# Script) dengan nama Bird. Script ini akan digunakan untuk pergerakan dan control dari Bird.

Tambahkan 2 variable

```
public float upForce;
private bool isDead = false;
```

Tambahkan line code pada method Start() untuk memulai pergerakan Bird

```
rb2d = GetComponent<Rigidbody2D>();
```

Tambahkan line code berikut pada method Update() untuk memberikan mouse klik control untuk menggerakkan Bird

```
if (isDead == false)
```

```
{
     if (Input.GetMouseButtonDown(0))
    {
        rb2d.velocity = Vector2.zero;
        Rb2d.AddForce(new Vector2(0, upForce));
    }
 }
```

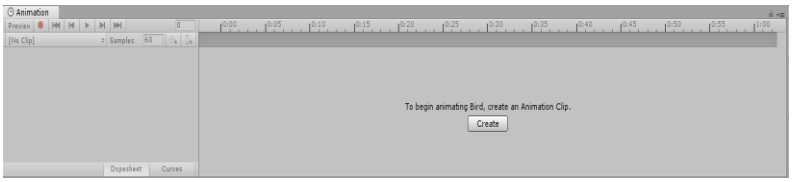
Tambahkan method OnCollisionEnter2D() untuk memberikan kondisi apabila Bird jatuh dia tidak bisa digerakkan lagi

```
void OnCollisionEnter2D(Collision2D other)
    {
     isDead = true;
    }
```

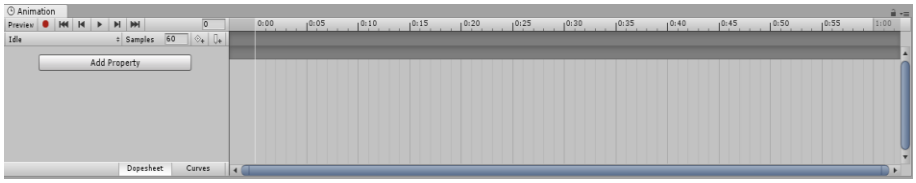
Pembuatan Animasi

Animasi digunakan mengubah state dari sebuah sprite, seperti misal orang berjalan, burung mengepakkan sayap dan lain-lain. Langkah pembuatan

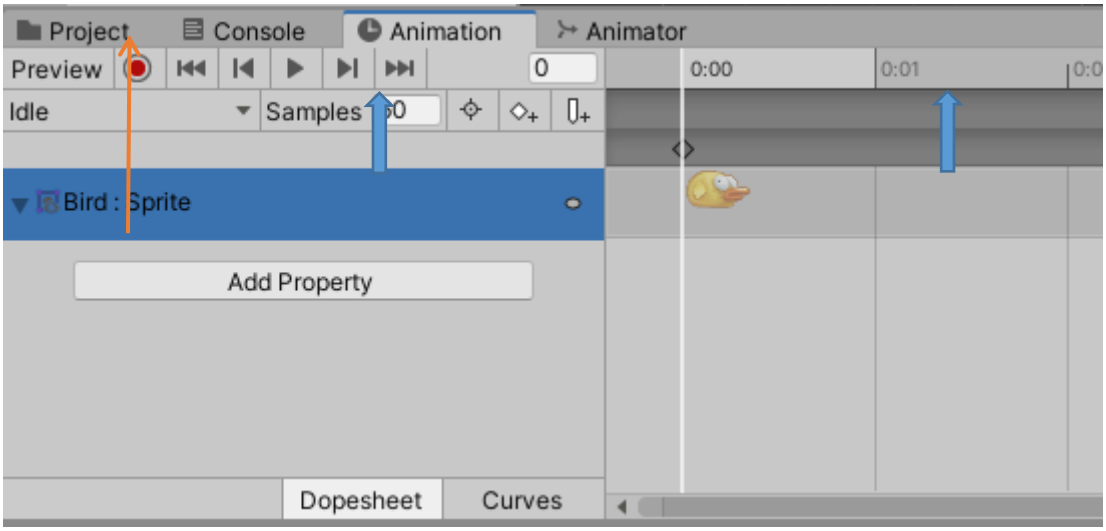
1. Buat Folder Animasi pada Asset
2. Klik Bird di game object/Hierrarki lalu klik menu Window --> Animation lalu akan muncul window seperti ini :



3. Lalu klik Create --> buatlah file .anim dengan nama “Idle.anim” pada folder Animasi yang sudah dibuat. Setiap file animasi yang kita buat akan mewakili gerakan/ state dari sprite yang kita buat, dalam kasus ini pada sprite Bird terdapat 3 state yaitu Idle saat dia diam, Flap pada saat Bird menaikkan sayap dan Dead pada saat dia mati, Setelah dibuat akan muncul window seperti ini

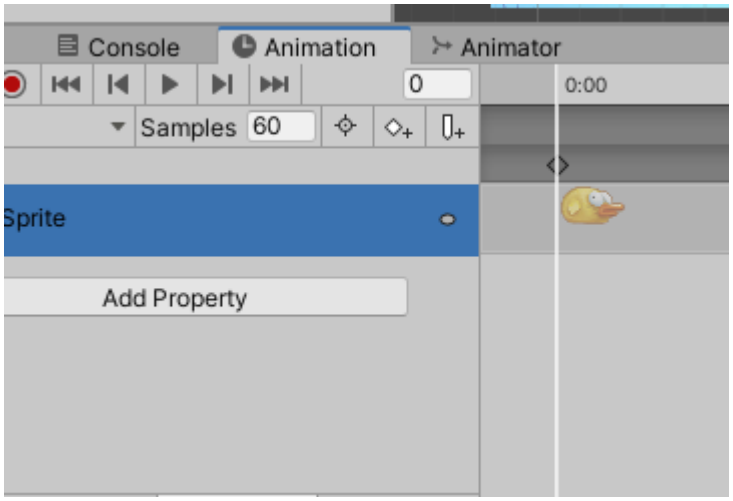


4. Selanjutnya Klik Add Property-->Sprite Renderer--> Klik tanda + pada Sprite Lalu akan muncul Kondisi awal dan kondisi akhir dari animasi Bird menjadi seperti ini:

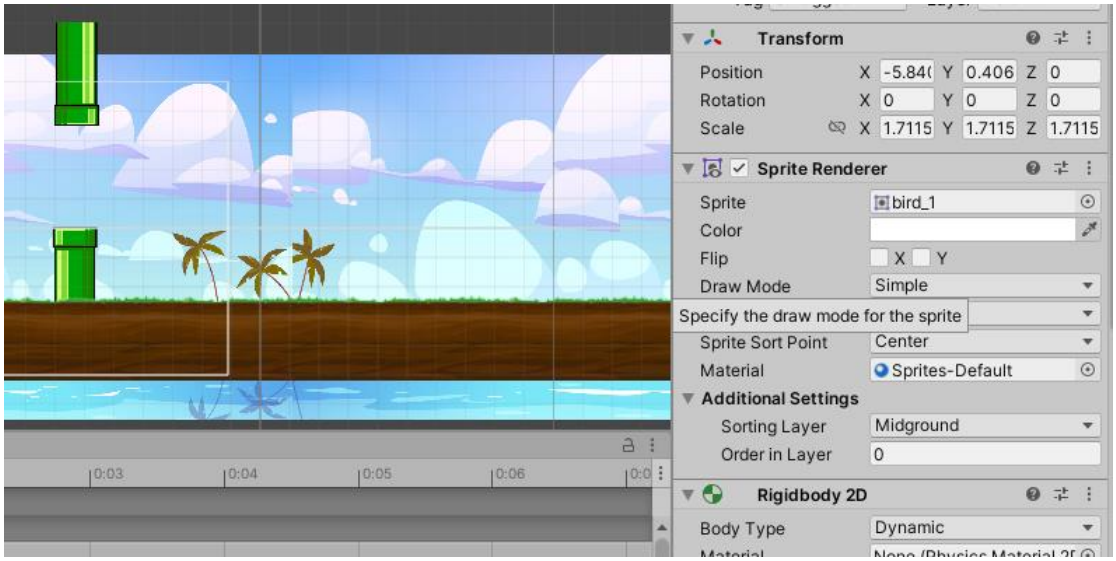


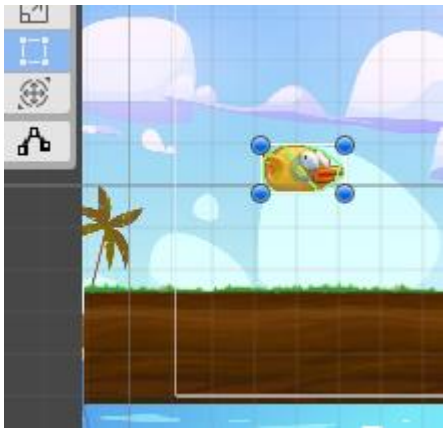
Catatan : Setiap merubah apapun dalam anim pastikan recornya menyala apabila tidak menyala klik

5. Selanjutnya kita hapus kondisi berhentinya menjadi :



6. Ulangi langkah 4 dan 5, Lalu pada Panel Inspector Pada Sprite Rendered Ubah BirdHero_0 dengan BirdHero_1
7. Untuk state Dead, Kita ulangi 6 dan 7 dengan nama file nya adalah Dead.anim dan mengganti BirdHero_1 pada Sprite Renderee dengan BirdHero 2, Sehingga akan menjadi seperti ini pada window game\

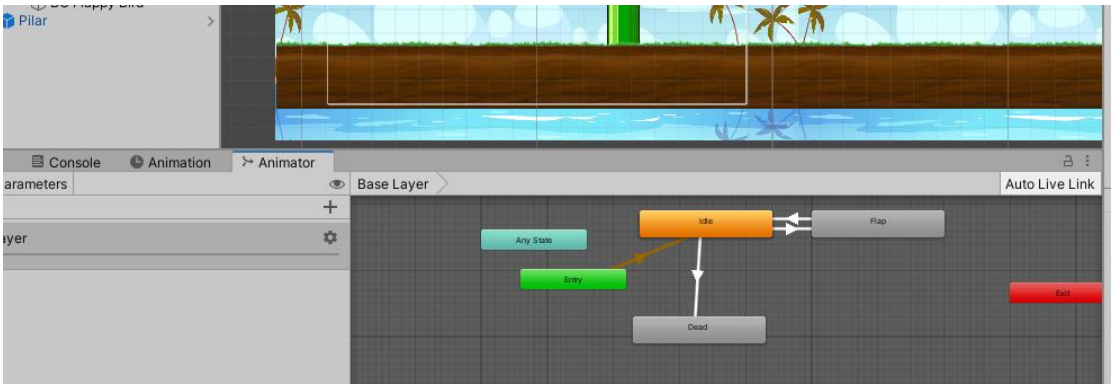




Pengaturan Animasi

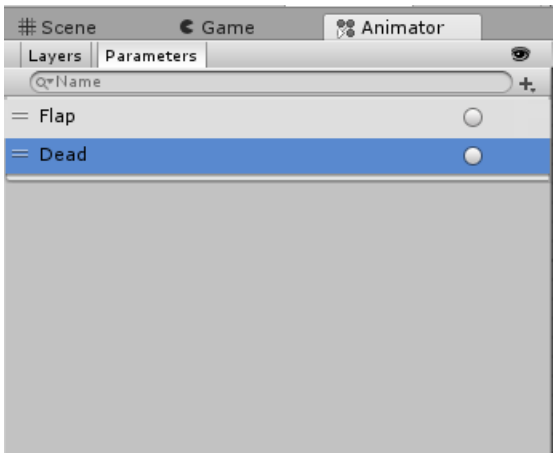
Pengaturan animasi ini dilakukan untuk menggabungkan animasi animasi yang sudah kita buat. Utuk Langkahnya sebagai berikut :

1. Window Animator, Seperti Berikut :

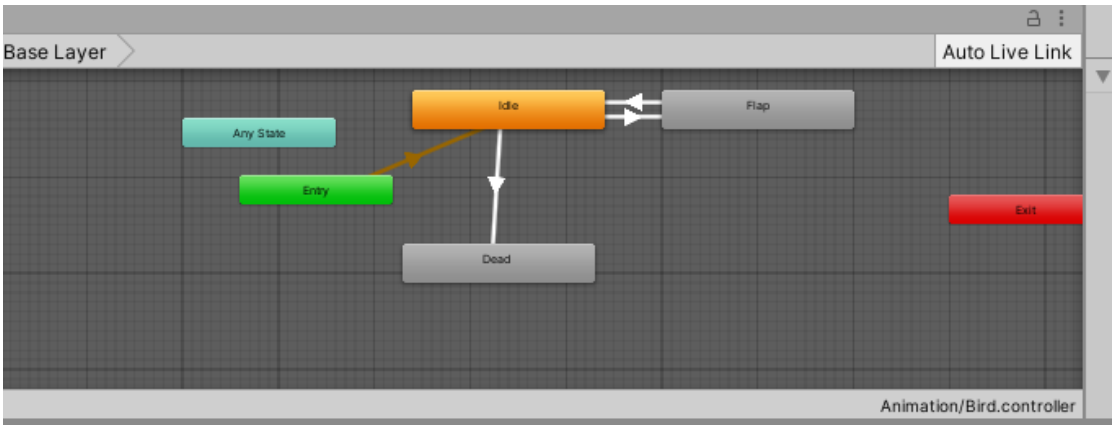


2. Balok - balok yang berada dalam window Animator panel BaseLayer merupakan state yang tersedia dalam scene kita. State state tersebut akan kita hubungkan sehingga pergerakan Bird sesuai dengan keinginan kita. Di dalam Window animator terdapat 2 panel yaitu panel Layers dan panel Parameters.

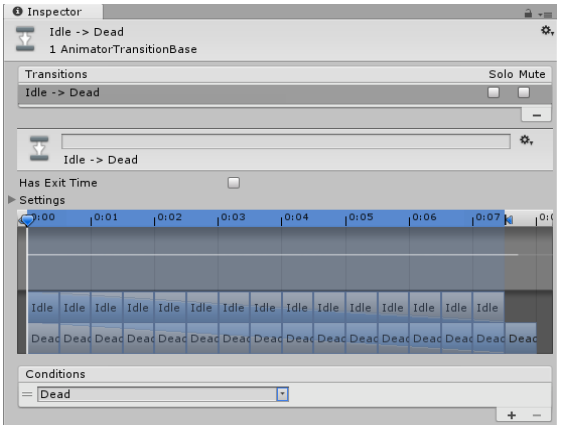
Klik Parameters --> Klik tombol Plus --> Triggered. Lalu ketikan Flap dan Died.sehingga menjadi :



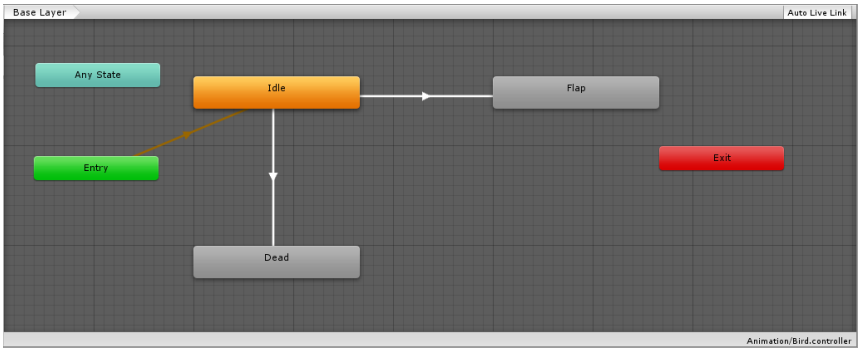
3. Selanjutnya kita akan menghubungkan state dari Bird nya. Yang pertama klik kanan pada balok Idle--> make Transaction lalu arahkan anak panah ke Balok Dead. Sehingga menjadi



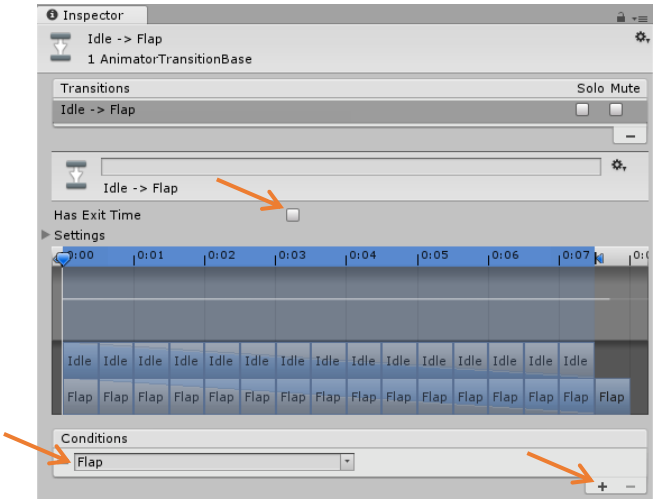
4. Selanjutnya Klik pada garis yang muncul sebelumnya akan muncul Inspector , Pada Panel Inspector Uncheck Has Exit Time dan Tambahkan Condition Dead.Menjadi seperti Berikut



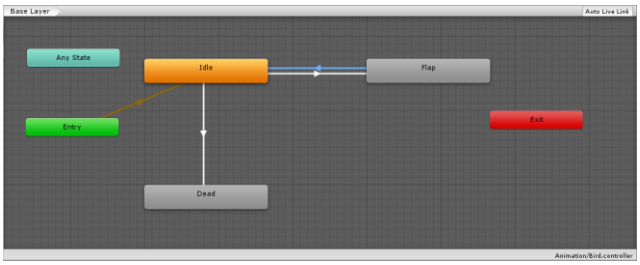
5. Selanjutnya klik kanan pada balok Idle--> make Transaction lalu arahkan anak panah ke Balok Flap. Sehingga menjadi



6. Selanjutnya Klik pada garis yang muncul sebelumnya akan muncul Inspector , Pada Panel Inspector Uncheck Has Exit Time dan Tambahkan Condition Flap.Menjadi seperti Berikut :



7. Selanjutnya klik kanan pada balok Flap--> make Transaction lalu arahkan anak panah ke Balok Idle. Dan untuk kasus yang ini kita tidak mengubah apapun pada panel Inspector, Sehingga menjadi



8. Selanjutnya kita kembali ke file Script Bird.cs, kita menambahkan variable :

```
private Animator anim;
```

Pada method Start() tambahkan=

```
anim = GetComponent<Animator>();
```

Pada method Update() tambahkan di dalam

```
if (Input.GetMouseButtonDown(0))
 {
```

```
            rb2d.velocity = Vector2.zero;
            rb2d.AddForce(new Vector2(0, upForce));
            anim.SetTrigger("Flap");
        }
```

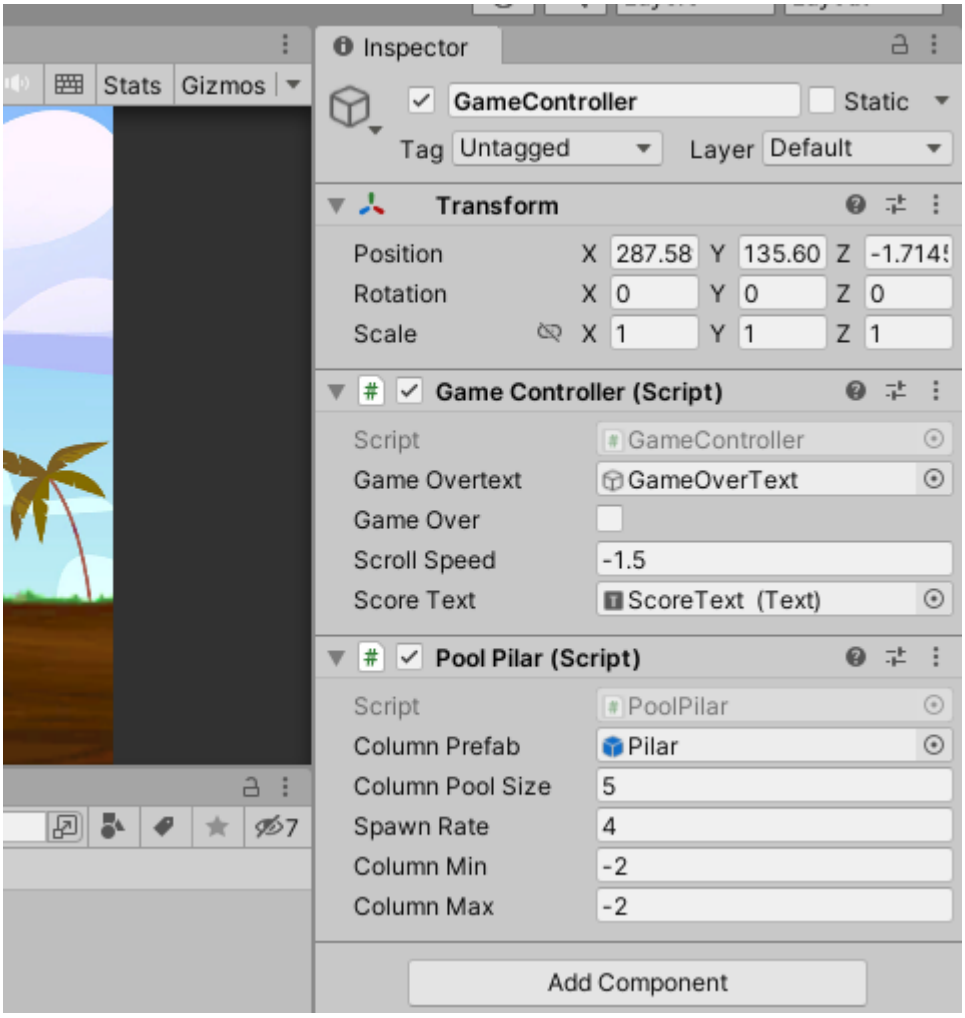
Pada method OnCollisionEnter2D tambahkan

```
anim.SetTrigger("Dead");
```

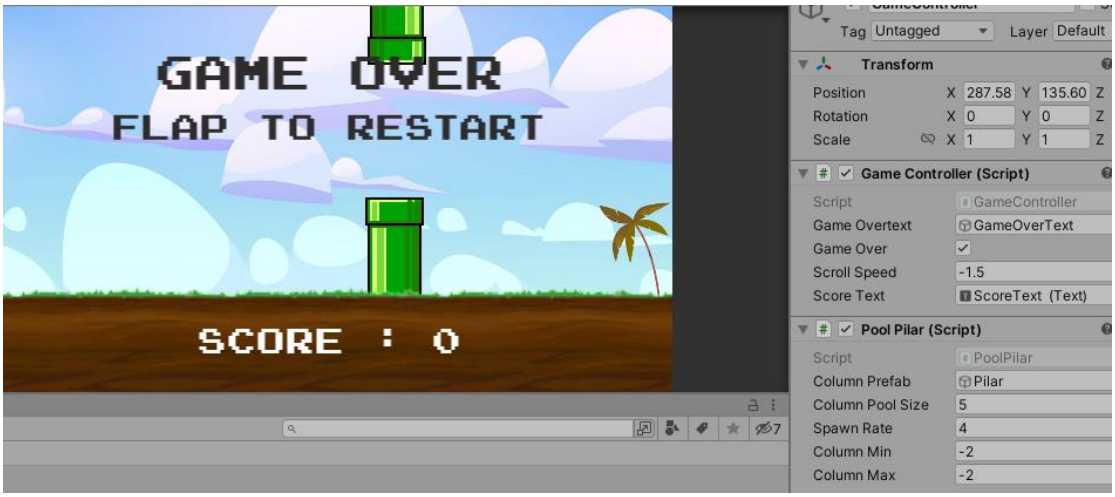
Pembuatan Tampilkan Score

1. Buat folder baru pada Asset bernama Scene. Masukkan Scene kalian ke folder tersebut
2. Buatlah 3 Text Pada Scene Kalian Text Pertama adalah ScoreText

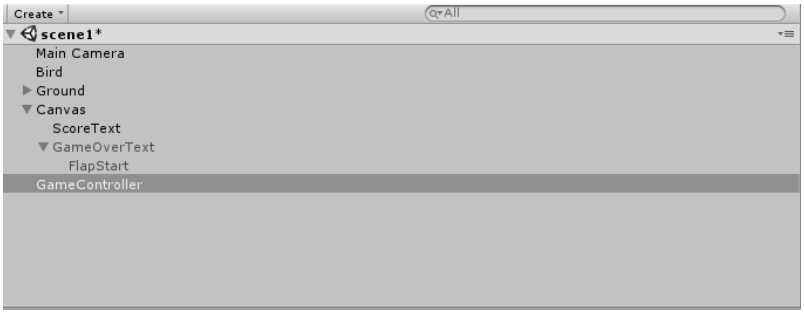
3. Text Kedua adalah GameOverText, untuk default nya uncheck check button pada panel Inspector



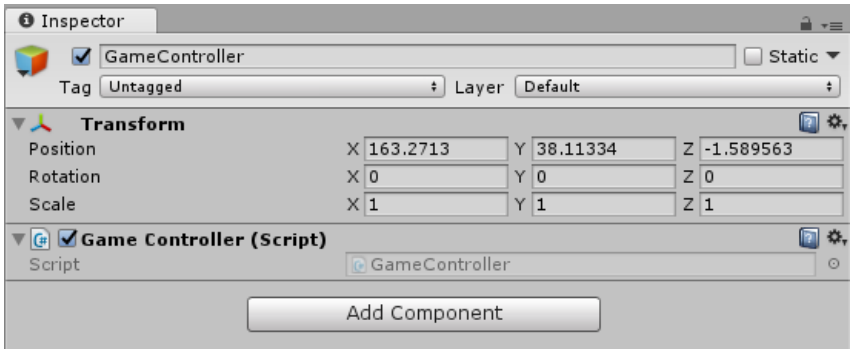
4. Text Ketiga adalah FlapStart, buat text ini menjadi Child dari text GameOverText, sehingga seluruh tamoulan textnya menjadi seperti ini



5. Setelah membuat tampilan score, selanjutnya kita akan menambahkan Game Object untuk mengatur jalannya game, dengan cara : GameObject--> Create Empty, lalu ubah namanya menjadi GameController sehingga menjadi



6. Selanjutnya tambahkan Script pada GameController dengan nama Gamme Controller juga dengan cara, Add Component--> New Scripts lalu masukkan namanya

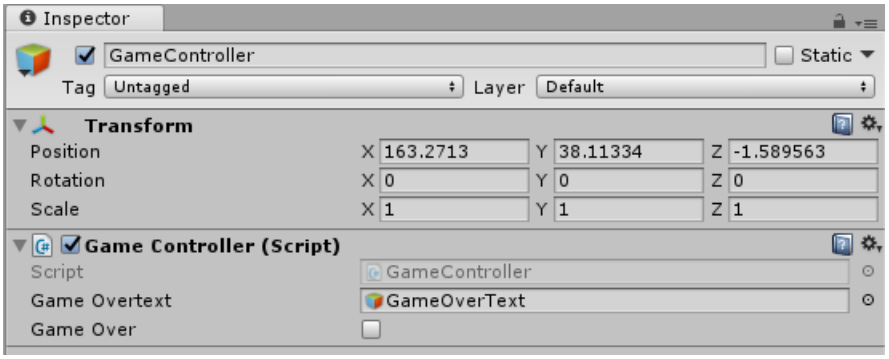


7. Lalu edit GameController.cs Sehingga menjadi seperti di bawah ini :

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



public class GameController : MonoBehaviour {
    public static GameController instance;
    public GameObject gameOvertext;
    public bool gameOver = false;


    void Awake()
    {
        if (instance == null)
            instance = this;
        else if (instance != this)
            Destroy(gameObject);
    }
    void Update()
    {
        if (gameOver && Input.GetMouseButtonDown(0))
        {
           SceneManager.LoadScene(SceneManager.GetActiveScene().
           buildIndex);
        }
    }

    public void BirdDied()
    {
```

```
            gameOvertext.SetActive(true);
            gameOver = true;
        }
    }
```

8. Selanjutnya tambahkan GameOverText ke dalam script GameController



9. Lalu tambahkan Code di bawah ini ke dalam method OnCollisionEnter2D pada script Bird.cs

```
GameController.instance.BirdDied();
```

Membuat Background Bergerak

1. Kita akan menambahkan 2 Script Code yaitu ScrollingObject.cs dan RepeatingBackground.cs, dan keduanya implementasikan ke Sprite Ground dengan cara drag and drop.

2. Dimana Scrolling Object.cs berisi

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

public class ScrollingObject : MonoBehaviour {

    private Rigidbody2D rb2d;
    void Start()
    {
        rb2d = GetComponent<Rigidbody2D>();
        rb2d.velocity = new Vector2
       (GameController.instance.scrollSpeed,0);
    }

    void Update()
    {
        if (GameController.instance.gameOver == true)
        {
            rb2d.velocity = Vector2.zero;
        }
    }
}
```

3. Dan dimana RepeatingBackground.cs berisi

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

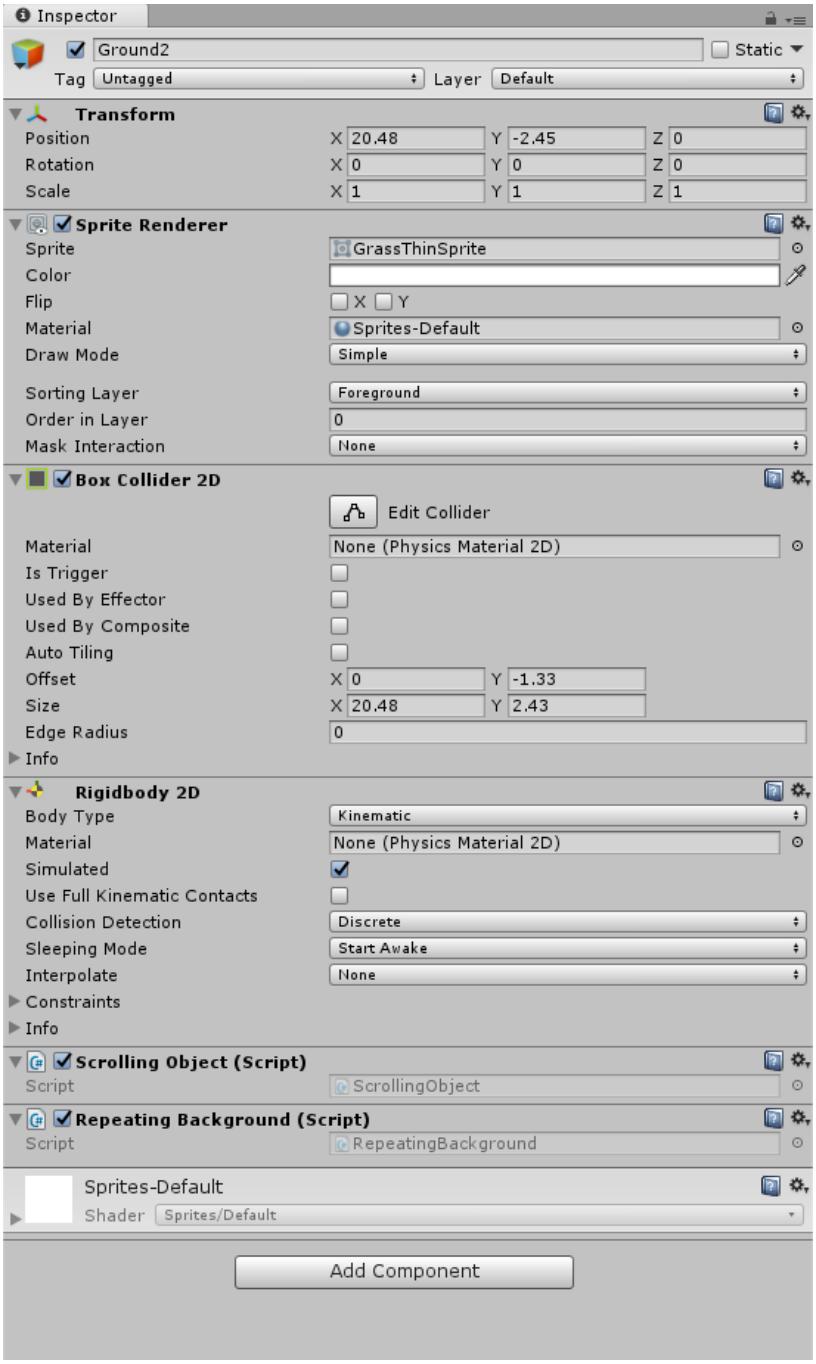
```
    public class RepeatingBackground : MonoBehaviour {

        private BoxCollider2D groundCollider;
        private float groundHorizontalLength;
        private void Awake()
        {
            groundCollider = GetComponent<BoxCollider2D>();
            groundHorizontalLength = groundCollider.size.x;
        }
        private void Update()
        {
            if (transform.position.x < -groundHorizontalLength)
            {
                 RepositionBackground();
            }
        }
        private void RepositionBackground()
        {
            Vector2 groundOffSet = new Vector2(groundHorizontalLength * 2f,
0);
            transform.position = (Vector2)transform.position +
groundOffSet;
        }
    }
```

4. Lalu Pada GameController.cs Tambahkan deklarasi variabel berikut :

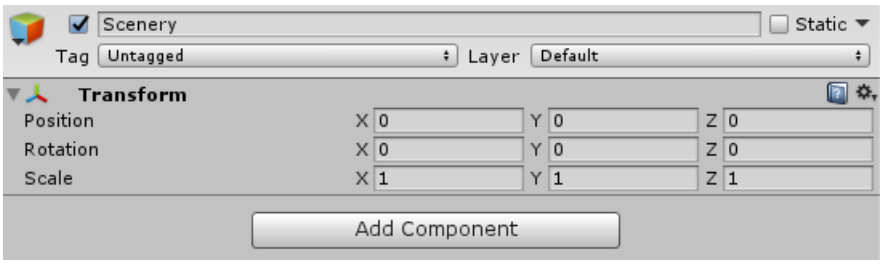
```
        public float scrollSpeed = -1.5f;
```

5. Agar backgound kita jauh setidaknya kita membutuhkan 2 kali background yang kita punya kita dapat menduplikat Ground yang kita punya dengan menamainya menjadi Ground2.

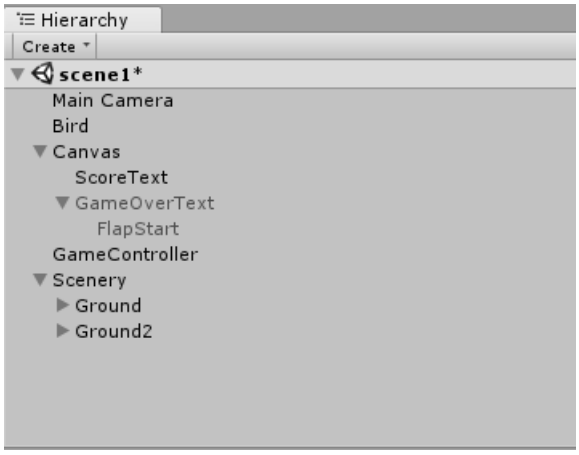
Kita akan mengatur posisi awalnya sesuai dengan panjang dari Ground kita pada Inspector.



6. Untuk mempermudah pengeditan kita buat GameObject dengan nama Scenery (GameObject-->Create Empty) dan set posisi awalnya menjadi seperti ini :

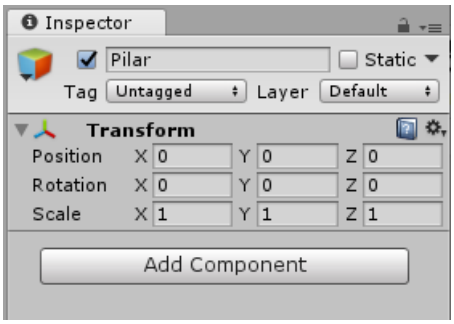


7. Setelah itu arahkan Sprite Ground 1 dan ground dua menjadi cild dari Scenery

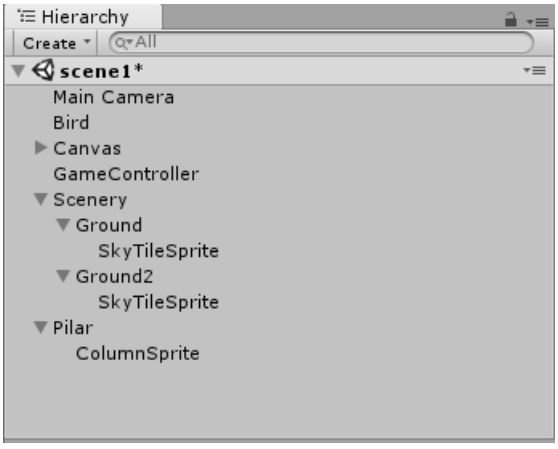


Membuat Rintangan

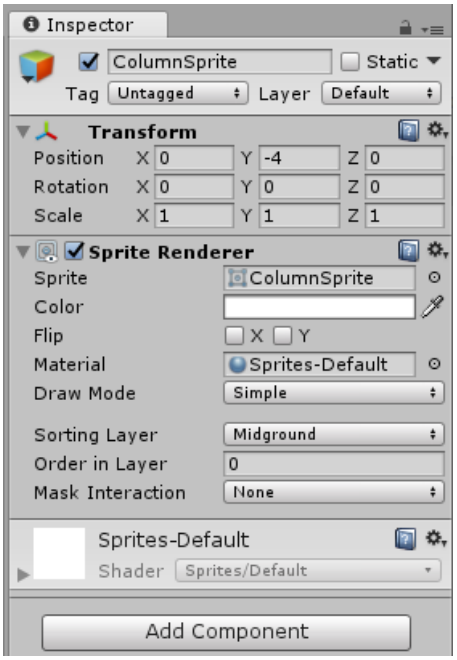
1. Untuk membuat rintangan kita akan membentuk suatu Game Object terlebih dahulu bernama Pilar dan pastikan posisi awal dari Pilar adalah x=0,y=0,z=0:



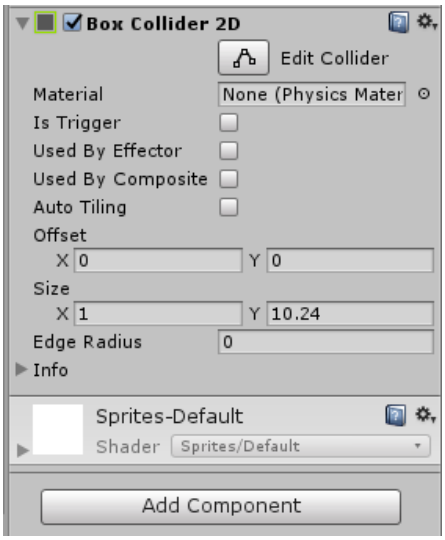
2. Selanjutnya kita akan memasukkan Sprite ColumnSprite ke dalam Scene dan menjadikannya child dari Pilar.



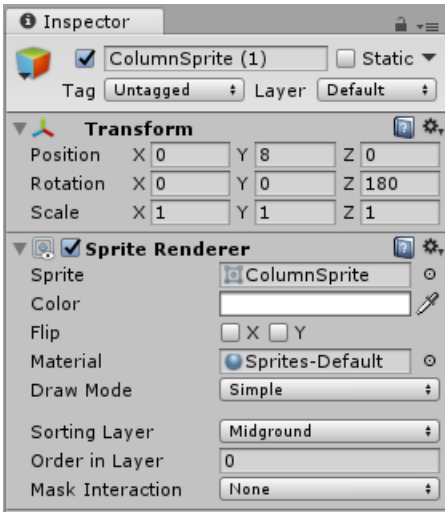
3. Lalu kita akan mengubah posisi dari ColumnSprite dan posisi layernya pada Inspector :



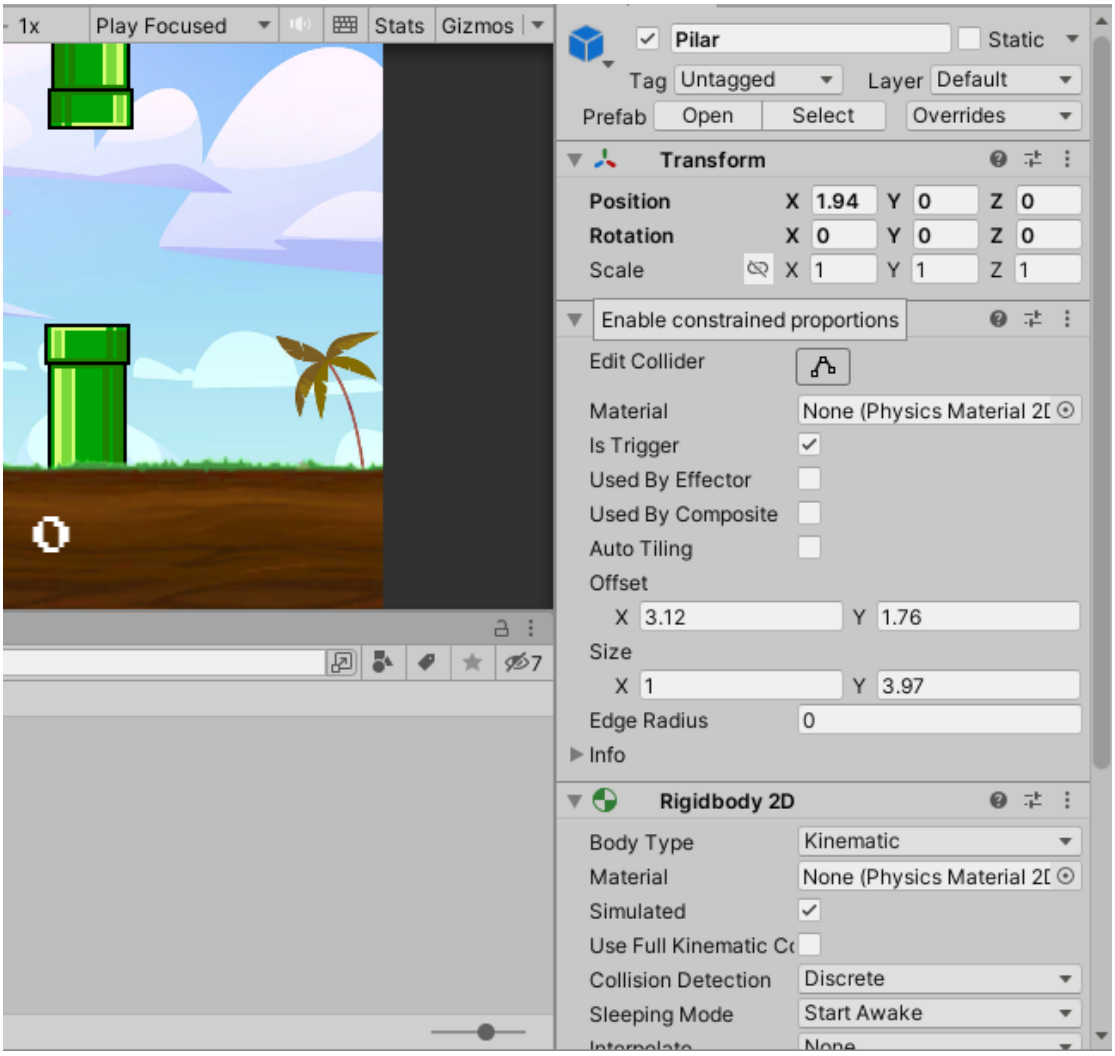
4. Selanjutnya tambahkan Box Collider 2D (Add Component --> Physics 2D --> Box Collider 2D) pada ColumnSprite dan ubah ukuran boxnya sehingga fit dengan gambar pilarnya.



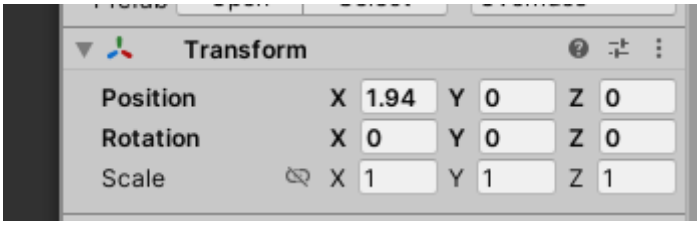
5. Setelah itu Duplicate ColumnSprite untuk membuat batas pilar atas dan aturlah posisinya seperti di bawah ini :



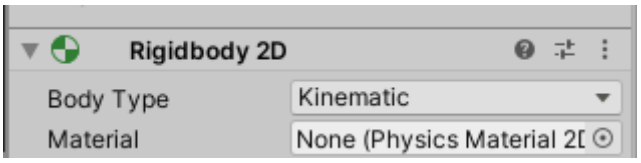
6. Selanjutnya masuk ke GameObject Pilar, kita tambahkan Box Collider 2D (Add Component --> Physics 2D --> Box Collider 2D), atur Box Collider berada di belakang celah antara 2 ColumnSprite. Seperti Berikut :



7. Dan kita ubah posisi dari Pilar menjadi seperti berikut :



8. Selanjutnya Tambahkan Rigidbody 2D (Add Component --> Physics 2D --> Rigidbody 2D). dan ubah Body Type menjadi Kinematic :



9. Selanjutnya masukkan script ScrollingObject.cs (Add Component--> scripts-->ScrollingObject) yang sudah kita buat sebelumnya ke Object Pilar

10. Setelah itu kita akan mengatur score apabila berhasil melewati pilar dengan menambahkan Script baru yaitu Column.cs dan menambahkan line code pada GameController.cs, untuk kita tambahkan :

Pada awal kita akan tambahkan library baru

```
using UnityEngine.UI;
```

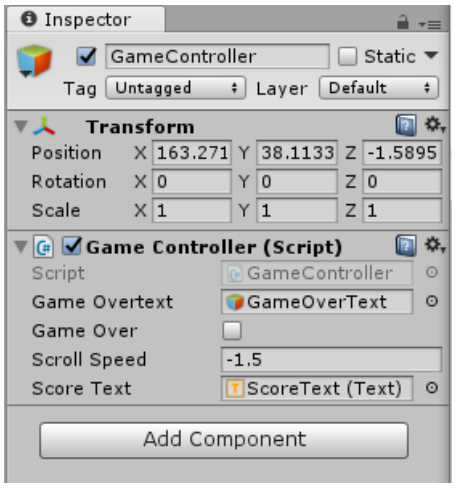
Selanjutnya tambahkan 2 variabel yaitu

```
public Text scoreText;
private int score = 0;
```

Selanjutnya tambahkan method BirdScore() untuk menghandle masalah penscoran yang berisi :

```
public void BirdScored()
{
    if (gameOver)
        return;
    score++;
    scoreText.text = "Score: " + score.ToString();
}
```

11. Dan masukkan score Text ke field Score Text pada object Game Controller:



12. Selanjutnya kita akan membuat Script baru bernama column.cs dan kita akan memasukkannya di Object Pilar. Dan isi dari Column. Cs adalah

```
public class Column : MonoBehaviour {

    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.GetComponent<Bird>() != null)
        {
            GameController.instance.BirdScored();
        }
    }
}
```

Script ini digunakan untuk mentrigger apabila Bird melewati celah, setelah itu masukkan script tsb dalam Pilar (Add Component--> scripts-->Column).

Memunculkan Pilar

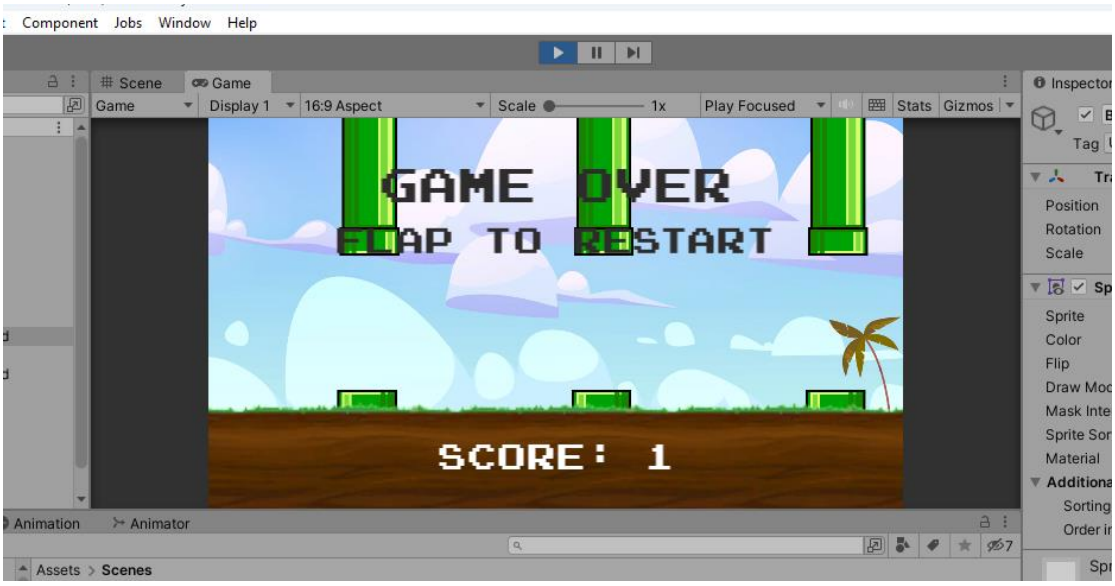
1. Kita akan membuat Script bernama PoolPilar.cs yang nantinya akan kita masukkan ke dalam Object GameController : Pool Pilar berisi :

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

public class PoolPilar : MonoBehaviour {

    public GameObject columnPrefab;//The column game object.
    public int columnPoolSize = 5;//How many columns to keep on standby.
    public float spawnRate = 3f;//How quickly columns spawn.
    public float columnMin = -1f; //Minimum y value of the column position.
    public float columnMax = 3.5f;//Maximum y value of the column position.

    private GameObject[] columns; //Collection of pooled columns.
```

```
    private int currentColumn = 0; //Index of the current column in the collection.

    private Vector2 objectPoolPosition = new Vector2(-15, -25);//A holding position
for our unused columns offscreen.
    private float spawnXPosition = 10f;

    private float timeSinceLastSpawned;


    void Start()
    {
        timeSinceLastSpawned = 0f;
        //Initialize the columns collection.
        columns = new GameObject[columnPoolSize];
        //Loop through the collection...
        for (int i = 0; i < columnPoolSize; i++)
        {
            //...and create the individual columns.
            columns[i] = (GameObject)Instantiate(columnPrefab,
            objectPoolPosition,Quaternion.identity);
        }
    }


    //This spawns columns as long as the game is not over.
    void Update()
    {
        timeSinceLastSpawned += Time.deltaTime;

        if (GameController.instance.gameOver == false && timeSinceLastSpawned
           >=spawnRate)         {
            timeSinceLastSpawned = 0f;

            //Set a random y position for the column
            float spawnYPosition = Random.Range(columnMin, columnMax);

            //...then set the current column to that position.
            columns[currentColumn].transform.position = new Vector2(spawnXPosition,
           spawnYPosition);

            //Increase the value of currentColumn. If the new size is too big, set it
back to zero
            currentColumn++;

            if (currentColumn >= columnPoolSize)
            {
                currentColumn = 0;
            }
        }
    }
}
```

Hasil Game


Scene
Game
Display 1
16:9 Aspect
Scale
1x
Play Focused
Stats
Gizmos
SCORE : 0
Inspector
BG Flappy Bird
Tag
Untagged
Transform
Position
Rotation
Scale
Sprite Renderer
Sprite
Color
Flip
Draw Mode
Mask Interaction
Sprite Sort Point
Material
Additional Settings
Sorting Layer
Animation
Animator


Component
Jobs
Window
Help
Scene
Game
Display 1
16:9 Aspect
Scale
1x
Play Focused
Stats
Gizmos
GAME OVER
FLAP TO RESTART
SCORE: 1
Inspector
Position
Rotation
Scale
Sprite
Color
Flip
Material
Animation
Animator
Assets
Scenes