BAB III

UNITY

Unity adalah sebuah game engine cross platform yang dibuat oleh Unity Technologies, sebuah perusahaan yang berpusat di San Fransisco. Game Engine Unity dapat membuat game untuk beberapa platform, diantaranya PC, Smartphone, Console dan perlengkapan VR. Unity bisa digunakan untuk membuat game 3D maupun 2D. Tidak hanya itu, Unity juga dipakai di indusrtri-industri lain selain game, meskipun Unity lebih dikenal untuk membuat beberapa game dari tim indie, seperti Cuphead, Fall Guys, Hollow Knight dan masih banyak lagi.

|  |  |
| --- | --- |
| **(a)** | **(b)** |

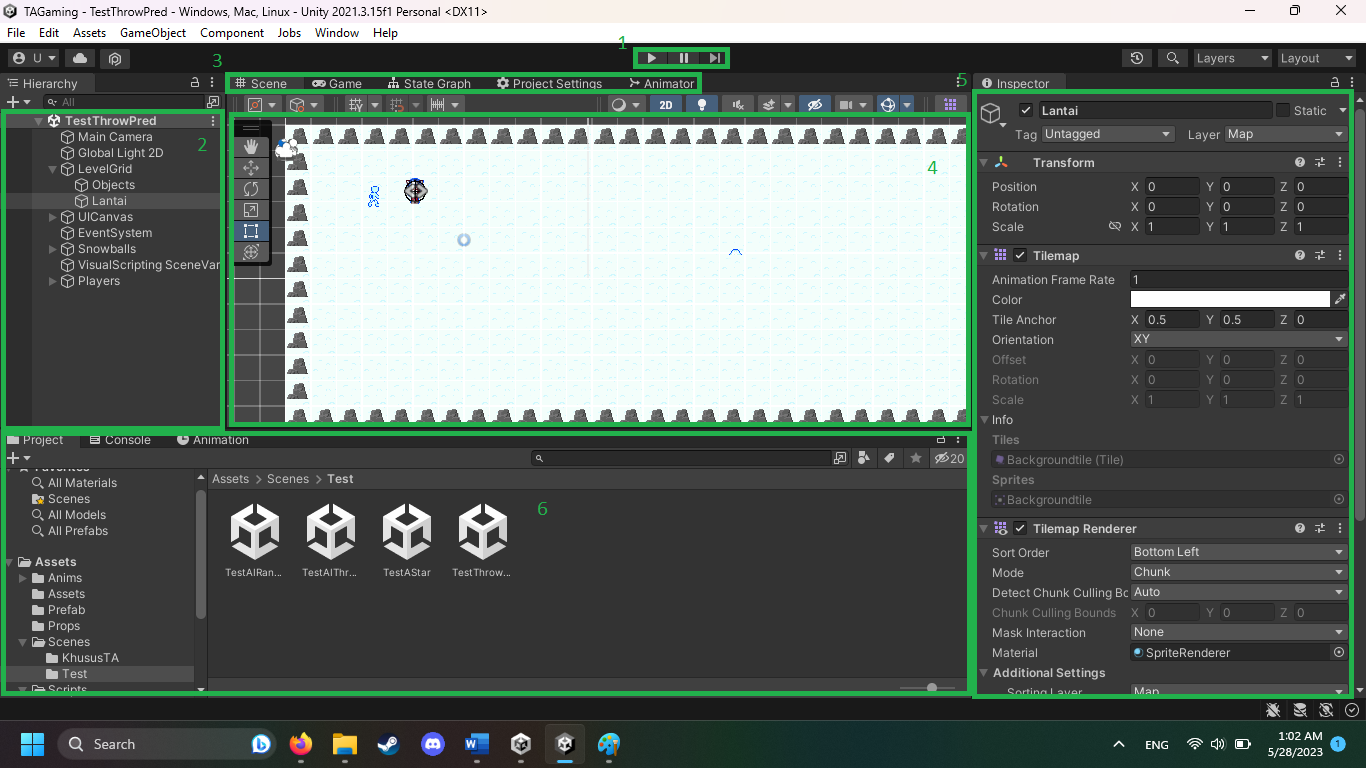
Gambar 3.1

Logo Unity (a) sebelum 2021 dan (b) dari 2021 hingga sekarang

Dalam bab ini akan dijelaskan konsep-konsep dan tahap-tahap yang diperlukan untuk membuat sebuah game, dalam kasus ini sebuah game 2D. Hanya akan dijelaskan subjek-subjek yang berhubungan dengan tugas akhir dikarenakan banyaknya pilihan yang disediakan oleh Unity. Subjek yang dijelaskan adalah sebagai berikut :

1. UI & Scene

Unity sebagai sebuah aplikasi memiliki sebuah UI yang cukup sederhana untuk tidak mengacaukan layar, tetapi juga cukup rumit untuk memberi informasi yang diperlukan oleh Game Developer. Secara default, Unity akan memiliki beberapa window seperti yang ditunjukkan gambar 3.2 dibawah ini. Setiap window dapat dipindah-pindah dan bisa membuat window baru bila diinginkan, dengan jumlah window secara bawaan sebanyak 4 window.



Gambar 3.2

UI Unity

Gambar 3.2 diatas adalah UI Unity secara default saat dibuka, tidak termasuk kotak-kotak yang digambar menggunakan Paint. Di UI default terdapat 6 bagian yang akan sering digunakan. Berikut adalah kegunaan dari setiap bagian dari UI diatas :

* 1 : Tombol play untuk menjalankan Scene yang sedang diedit, dengan tombol pause dan fast forward untuk mengatur laju kecepatan dari game. Berguna untuk membantu dalam melakukan debug.
* 2 : Tab Hierarki berisi semua Game Object yang ada di dalam Scene yang sekarang ini diedit
* 3 : Secara default berisi Tab Scene yang digunakan untuk melihat dan mengedit Scene kita dan Tab game yang digunakan untuk melihat tampilan di dalam game. Window lain bisa di geser ke sini dan akan Window tersebut akan muncul di bagian kanan kumpulan tab.
* 4 : Isi dari scene yang dipilih, tempat seluruh pembuatan Scene dibuat.
* 5 : Tab detail yang sesuai namanya berisi seluruh detail dari sebuah Game Object, akan dijelaskan di subbab selanjutnya.
* 6 : Tab Project yang berisi folder dan File-File asset dan script yang dipakai.

Seluruh window yang ada bisa di layar gambar 3.2 dapat digeser, diperbesar, diperkecil dan dihilangkan bila tidak perlu. Bila ingin mengakses sebuah window yang hilang, dapat menggunakan tab window di atas. Tetapi dengan begitu UI bawaan Unity sudah cukup bagus untuk keperluan dasar setiap Developer sehingga tidak perlu diganti terlalu banyak.

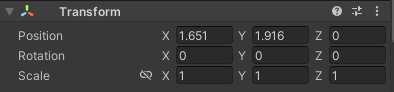
Dalam penjelasan diatas disinggung mengenai Scene dan Game Object. Game Object akan dijelaskan dalam subbab selanjutnya. Sementara itu, Scene adalah sebuah area dimana sebuah Player dapat melakukan apapun yang mereka kehendaki sesuai dengan desain Developer. Kebanyakan waktu, sebuah Scene dalam sebuah Game Development bisa dianggap sebagai sebuah Level, dimana setiap kali seorang player memasuki Level lainnya, akan diload Scene yang sesuai dengan Level yang barusan diselesaikan, termasuk Menu- Menu seperti Main Menu dan Loading Screen bisa dijadikan Scene yang berbeda.

1. Game Object & Prefab

Game Object sesuai dengan namanya adalah sebuah objek dalam sebuah Game, dimana game Object bisa berupa sebuah halangan ataupun karakter yang dikontrol oleh player. Yang membedakan sebuah objek rintangan dan player dalam unity adalah komponen-komponen yang dimiliki. Komponen-komponen pembeda tersebut dapat dilihat di bagian no 5 di gambar 3.2 diatas.

Berikut adalah komponen-komponen yang penting dalam pembuatan sebuah Game Object 2D Unity :

* 1. Transform

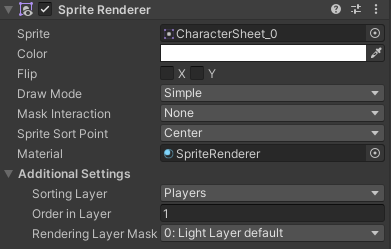


Gambar 3.3

Tampilan Transform

Transform digunakan untuk merealisasikan sebuah Game Object dalam sebuah lokasi 3D. Terdapat 3 variabel, yaitu Position yang menunjukkan posisi Game Object sekarang ini, Rotation menunjukkan rotasi Game Object sesuai dengan isi variable, seperti bila X Rotation berisi 30 berarti Game Object telah dirotasi sebanyak 30˚ terhadap sumbu X. Dan terakhir adalah Scale yang menunjukkan skala dari Game Object, jadi misal Y dari variabel Scale diisi 2, maka Game Object akan memanjang 2 kali lipat sesuai sumbu Y. Jadi Transform adalah komponen yang paling penting dan dasar dalam sebuah Game Object, khususnya bila Game Object dapat dilihat.

* 1. Sprite Renderer



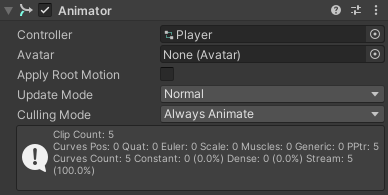
Gambar 3.4

Tampilan Sprite Renderer

Sprite Renderer adalah sebuah komponen yang berguna untuk menampilkan sebuah gameObject 2D menggunakan sebuah Sprite atau gambar. Tanpa Sprite Renderer, Game Object 2D tidak bisa dilihat Player tanpa menggunakan alternatif seperti membuat model 3D. Variabel yang perlu diketahui adalah Sprite berisi gambar yang merepresentasi Game Object, Color yang mengubah warna Sprite yang digunakan dan Flip untuk membalik Sprite tersebut secara X atau Y.

Lalu variabel-variabel selanjutnya yang bisa membantu adalah variabel-variabel di dalam Additional Settings, dimana setiap Sprite dicetak ke layar dalam urutan yang berbeda. Mengetahui ini. Additional setting digunakan untuk memanipulasi urutan pencetakan tersebut agar sprite tertentu tampil di depan/belakang sprite lainnya sesuai kebutuhan Developer. 3 Variabel dalam Additional Settings adalah Sorting Layer yang menentukan Layer di mana sprite ini akan dicetak, Order in Layer yang menentukan urutan pencetakan sprite dalam layer tersebut dan terakhir Rendering Layer Mask yang digunakan untuk memilih Mask yang digunakan untuk mencetak sprite.

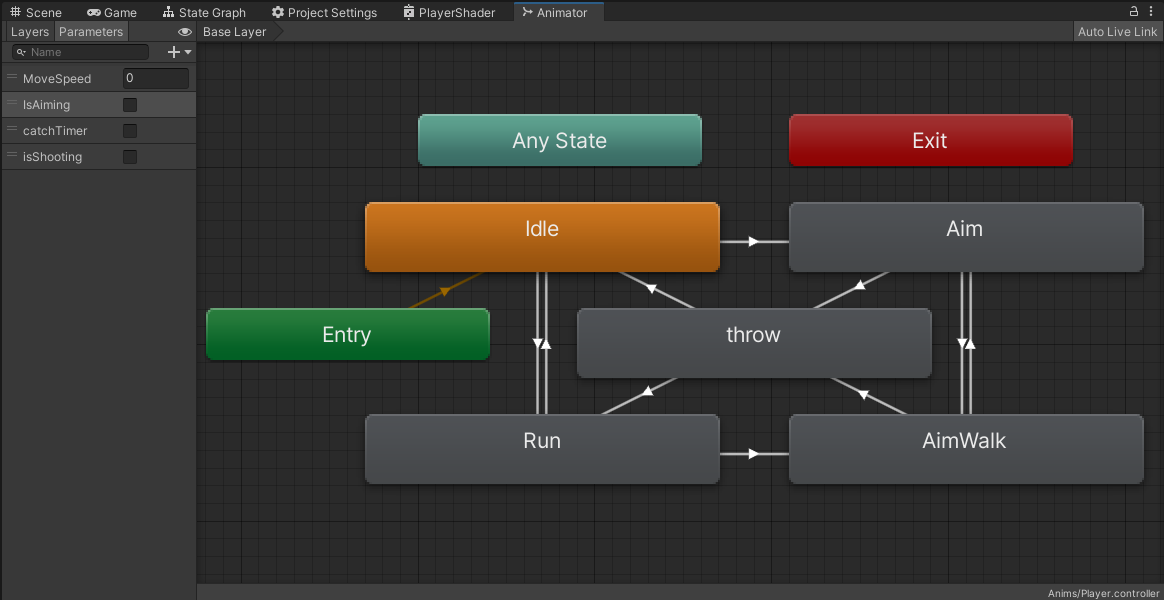
* 1. Animator & Animation



Gambar 3.5

Tampilan Animator

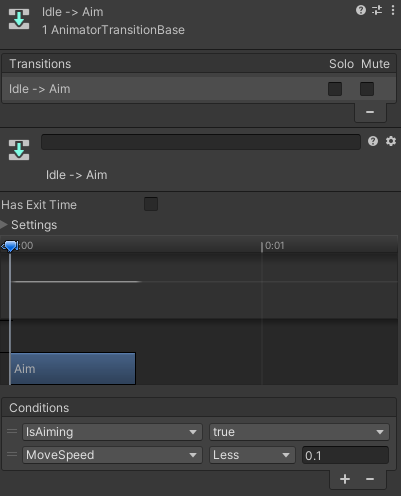
Animator adalah komponen yang memberi kehidupan ke gambar / model GameObject, dimana dia bertanggung jawab dalam mensimulasikan gerakan. Kita hanya perlu focus di Controller, karena Avatar dan Apply Root Motion hanya berlaku untuk karakter 3D, Update Mode bersifat situasional dan Culling mode hanya dipakai untuk mengoptimasi performa.



Gambar 3.6

Isi dari sebuah Animation Controller

Bila sebuah Animation Controller atau Animator dibuka, maka inilah layar yang akan tampil. Setiap kotak dalam gambar 3.6 adalah sebuah animasi, kecuali kotak berwarna hijau, biru dan merah, sementara kotak jingga merupakan animasi yang pertama dimainkan oleh GameObject dan dari situ animasi bisa berubah / berpindah ke animasi lain.

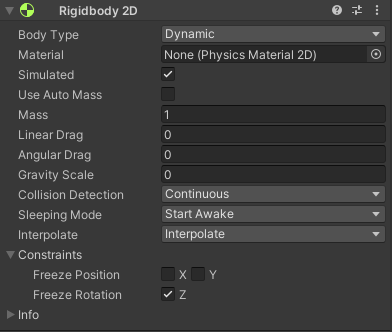


Gambar 3.7

Isi dari sebuah Transisi

Setiap animasi yang ada dalam Animator dapat dihubungkan ke satu sama lain menggunakan transisi, dan setiap transisi tersebut dapat diberi sebuah kondisi yang terdapat di bagian bawah gambar 3.7 dimana bila seluruh kondisi terpenuhi, maka transisi akan terjadi.Kondisi tersebut terdapat di bagian kiri dari layar Animator sesuai dengan gambar 3.7, dimana variabel hanya memiliki jenis data int, float, bool dan sebuah trigger. Transisi ini bersifat satu arah, sehingga terdapat keleluasaan untuk mengatur kondisi animasi b ke a agar bisa berbeda a ke b.

* 1. Rigidbody 2D



Gambar 3.8

Rigidbody 2D

Rigidbody mensimulasikan gerakan, gravitasi, dsb yang berhubungan dengan fisika untuk memudahkan user dalam membuat gerakan untuk karakter-karakternya bahkan dalam area 2 dimensi. Sekarang untuk variabel-variabel yang penting :

* Body Type : Memilih diantara 3 jenis tubuh : Dynamic, Kinematic dan Static. Ketiga tubuh ini memiliki reaksi yang berbeda terhadap kolisi dan simulasi fisika, dimana Dynamic adalah pilihan paling aman diantara mereka bertiga.
* Simulated : Memilih bila fisika ingin disimulasikan
* Mass : Massa dari Gameobject
* Linear & Angular Drag : Kekuatan dari udara atau Gameobject lain untuk melemahkan gerakan Linear & Rotasional sebuah Gameobject dimana semakin tinggu nilai Drag maka lebih cepat gerakan / rotasi sebuah objek berhenti setelah sebuah kolisi.
* Gravity Scale: Kekuatan gravitasi pada objek ini
* Collision Detection : Menentukan seberapa sering kolisi dideteksi
* Sleeping mode : Menentukan keaktifan Rigidbody saat objek dibuat
* Interpolate : Menentukan bila gerakan ingin diinterpolasi atau tidak.
* Freeze Position : Mengunci objek agar tidak bisa bergerak di sumbu yang dicentang
* Freeze Rotation : Mengunci objek dari berrotasi
  1. Collider 2D

|  |  |
| --- | --- |
| **(a)** | **(b)** |

Gambar 3.9

(a) Komponen Box Collider 2D dan (b) tampilan di dalam Gameobject

Collider 2D adalah teman baik Rigidbody 2D bila user ingin objek bisa menabrak / ditabrak dengan objek lain, dimana Collider membuat sebuah bentuk yang akan menjadi “tubuh” objek yang bisa ditabrak sesuai di gambar 3.9b yang ditandai dengan sebuah kotak berwarna hijau di tengah. Bentuk dari collider bisa bermacam macam, dari kotak, tabung, lingkaran dll. Tidak hanya itu, satu objek bisa memiliki lebih dari 1 Collider jadi 1 objek bisa memiliki beberapa collider agar bentuk kolisi bisa lebih akurat.

Lalu untuk variabel yang diperlukan, hanya diperlukan 2 yang penting. 2 Variabel tersebut adalah Edit Collider yang bila diklik akan membuat kotak hijau di gambar 3.9b bisa dirubah sesuai kemauan user. Variabel kedua yang penting adalah is Trigger, dimana bila dicentang, semua gameobject meskipun dengan Rigidbody 2D dan Collider 2D akan melewati objek, dimana akan dilakukan fungsi onTriggerEnter ketimbang onCollisionEnter, sangat berfungsi dalam situasi tertentu.

1. Script

Script adalah kode kita yang dalam unity dianggap sebagai sebuah komponen, dimana sebuah script dapat dipasang dalam sebuah gameobject, dan setelah dipasang maka objek akan berlaku sesuai dengan apa yang tertera dalam script. Kita tidak perlu membuat function khusus seperti main untuk dijalankan menggunakan script, Unity telah menyediakan beberapa intergrasi yang memudahkan dalam membuat obyek yang berbeda, salah satunya adalah variabel. Bila sebuah variabel dianggap publik maka variabel tersebut dapat diisi melalui unity, atau dengan memberi [SerializeField] sebagai pengganti public agar bisa dimasukkan melalui Unity, tapi tidak dianggap publik oleh script lain.

|  |  |
| --- | --- |
| **(a)** | **(b)** |

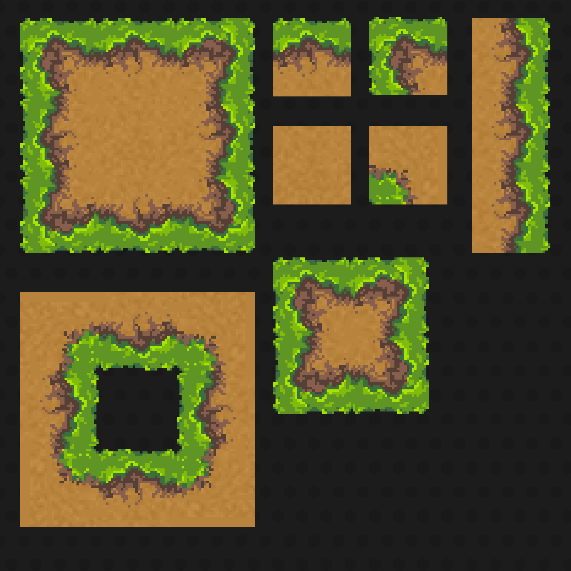
Gambar 3.10

(a) Komponen Script Bot Action dan (b) isi Variabel

Selain variabel, juga terdapat beberapa fitur lain, seperti GetComponent<ComponentDipilih>() yang dapat mengambil sebuah komponen yang dimiliki objek dan mengelola atau mengubah data yang diperlukan. Dan terakhir adalah beberapa fungsi khusus yang telah disimpan untuk kegunaan develompent Unity, yaitu :

* Awake : Dijalankan saat script dibuat
* Start : Dijalankan saat frame pertama objek ada
* Update : Dijalankan setiap frame
* FixedUpdate : Dijalankan setiap kali fisik game akan diupdate
* LateUpdate : Dijalankan setelah semua update selesai
* OnTriggerEnter : Dijalankan bila objek kolisi dengan objek trigger lain
* OnCollisionEnter : Dijalankan bila objek kolisi dengan objek lain
* Coroutine : Sebuah fungsi yang berjalan secara async dari update dkk dimana sebuah Coroutine bisa menggunakan delay.

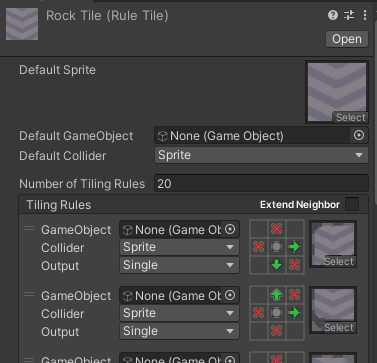
1. Tilemap & Rule Tile



Gambar 3.11

Contoh Tilemap

Tilemap adalah sebuah kumpulan tile yang dapat digunakan untuk membuat sebuah level. Gambar 3.11 diatas merupakan sebuah tilemap untuk tanah, dimana terdapat tile kotak di kiri atas yang bisa dipisah menjadi 3x3 agar setiap tile dapat ditaruh dalam game menggunakan sebuah grid yang telah disediakan unity. Kita dapat menggunakan Tile Palette untuk mengubah tilemap diatas menjadi siap dipakai seperti sebuah palet lukis.

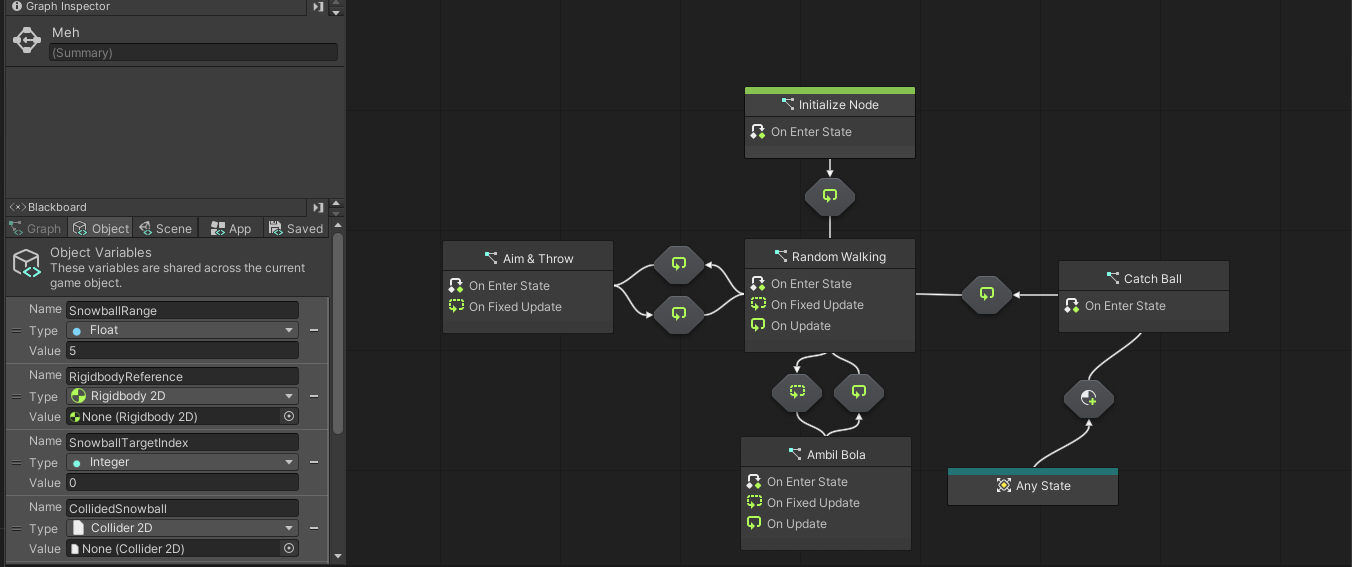


Gambar 3.12

Rule Tile

Tapi bila level yang dibuat dinamis, maka akan susah menempatkan tile yang tepat, sehingga terdapat tilerule, sebuah tile yang akan beradatapsi sesuai dengan penempatan tile tersebut. Misal di gambar 3.12 diatas, terdapat hingga 20 tile rule, sehingga ada 20 kemungkinan tile yang keluar dari 1 tile rule. Lalu untuk rule tersebut,

1. State Machine



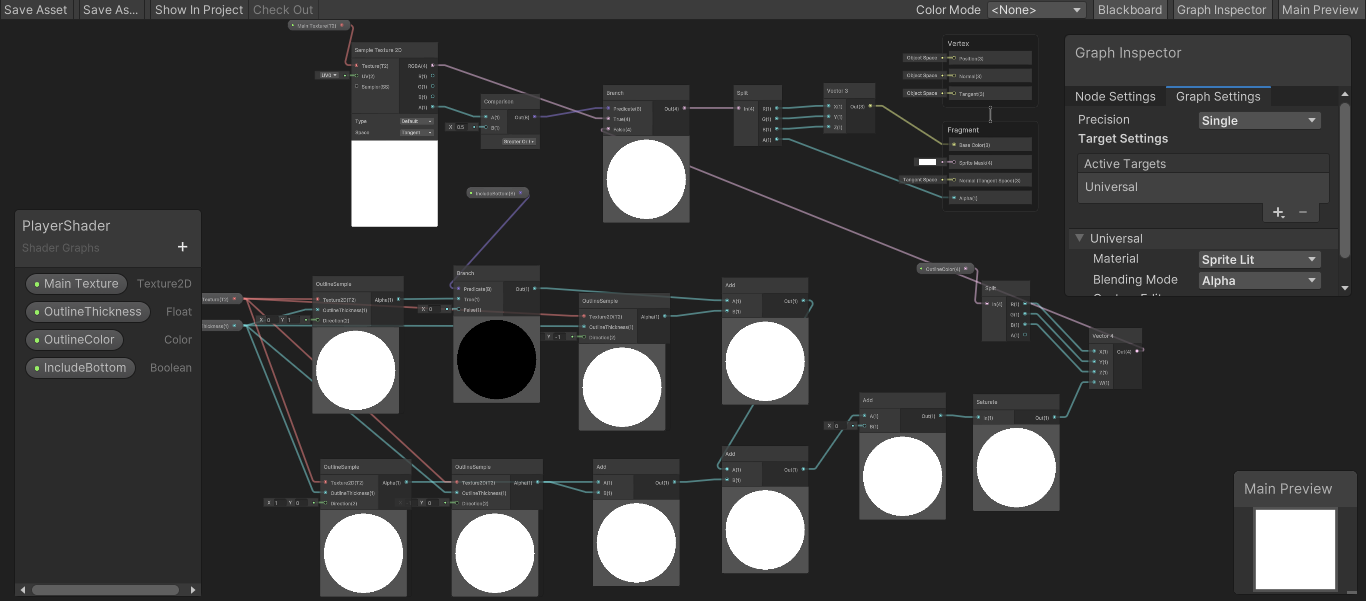
Gambar 3.13

State Machine

Bila tidak ingin membuat script secara C#, Unity menyediakan sebuah metode programming visual yang berupa state machine. Di gambar 3.13 diatas, di kiri bawah adalah variabel yang tersedia, sementara layar kanan merupakan state-state yang tersedia. State machine akan dimulai dari State yang ditandai dengan warna hijau, dan akan bergerak menuju state lain bila kondisi transisi terpenuhi, tidak seperti di animator, pengecekan transisi State Machine dilakukan melalui fungsi yang disediakan Unity seperti Update atau OnTriggerEnter.

1. Shader Graph

Shader Graph digunakan untuk membuat shader sendiri, dimana shader adalah sebuah fungsi komputer untuk menggambar sesuatu di layar. Sebuah Shader diprogram secara visual dengan visual scripting, dan setelah membuat sebuah shader graph, shader graph tersebut dapat dibuat menjadi sebuah material, dimana material tersebut dapat dipasang ke sebuah Sprite Renderer. Setelah dipasang, shader tersebut akan tampil, tapi bila ingin membuat shader dengan isi variabel yang berbeda, maka harus dibuat sebuah material baru karena bila material pertama diganti, perubahan ini akan terjadi pada seluruh gameobject yang memakai material yang sama.



Gambar 3.14

Shader Graph