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)** |

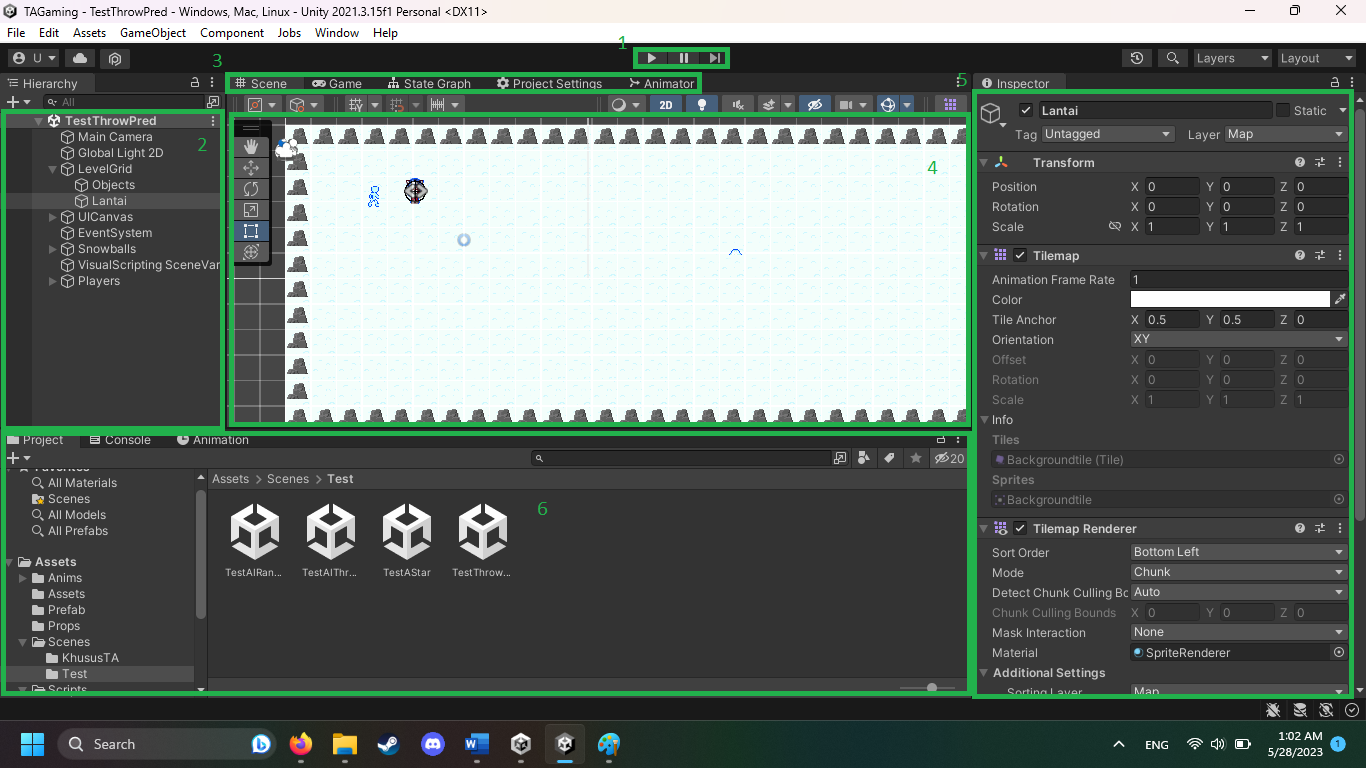
Gambar 3.1

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.

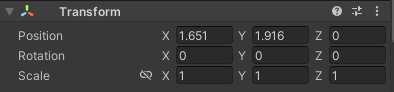
Dalam penjelasan diatas disinggung mengenai Scene dan Game Object. Game Object akan dijelaskan dalam subbab selanjutnya. Sementara itu, Scene adalah sebuah area dimana sebuah Player dapat melakukan apapun yang mereka kehendaki sesuai dengan desain Developer. Kebanyakan waktu, sebuah Scene dalam sebuah 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

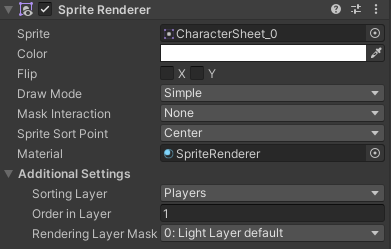


Gambar 3.3

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



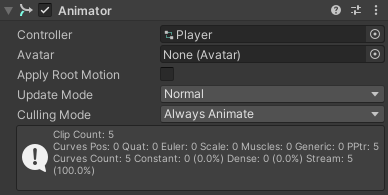
Gambar 3.4

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.

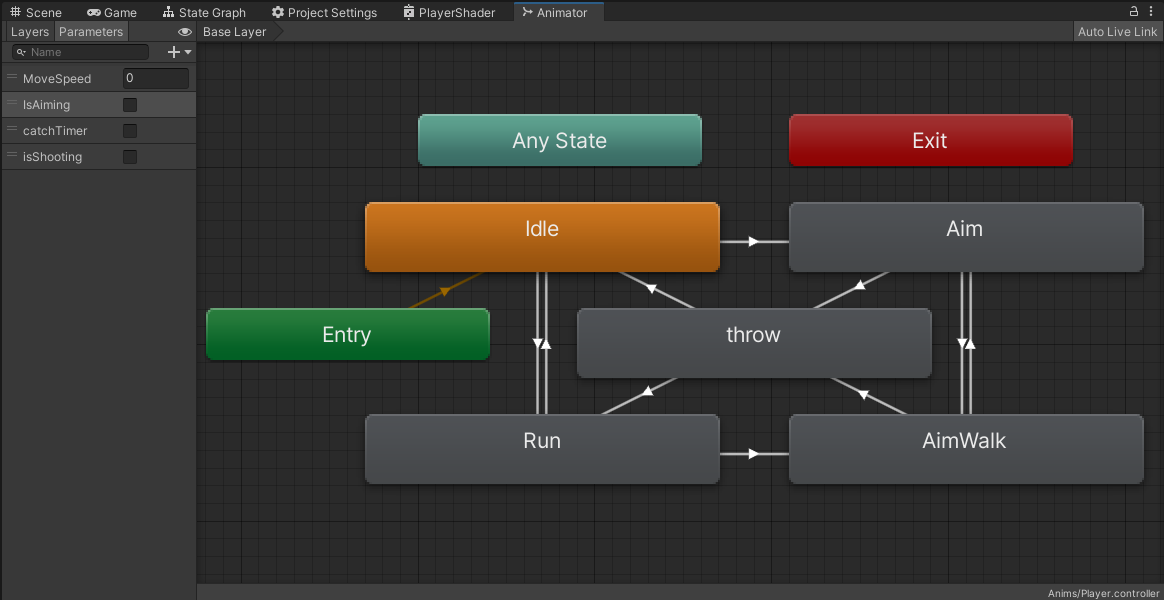
* 1. Animator & Animation



Gambar 3.5

Tampilan Animator

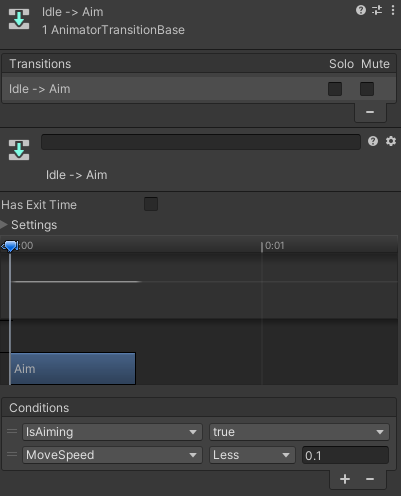
Animator adalah komponen yang memberi kehidupan ke gambar / model Gameobject, dimana komponen ini bertanggung jawab dalam mensimulasikan gerakan. Controller adalah komponen yang berisi bagaimana cara Gameobject bergerak. Avatar dan Apply Root Motion hanya berguna untuk Gameobject 3D jadi diabaikan dan Terakhir, Update Mode dan Culling Mode mengatur apabila animasi dijalankan setiap waktu atau tidak.



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. Dari kotak jungga animasi bisa berubah / berpindah ke animasi lain berdasarkan anak-anak panah yang tersedia.

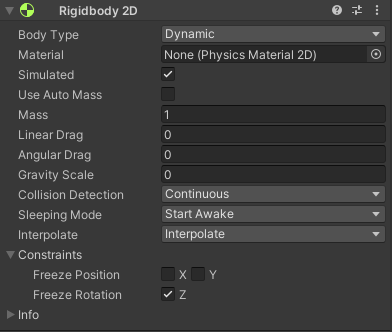


Gambar 3.7

Isi dari sebuah Transisi

Setiap animasi yang ada dalam Animator dapat dihubungkan ke satu sama lain menggunakan transisi yang akan tampil sebagai sebuah anak panah. Setiap transisi tersebut dapat diberi sebuah kondisi yang terdapat di bagian bawah gambar 3.7 dimana bila seluruh kondisi terpenuhi, maka perubahan animasi akan terjadi. Kondisi tersebut menggunakan variabel yang terdapat di bagian kiri layar Animator pada gambar 3.6, 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 dengan a ke b.

* 1. Rigidbody 2D



Gambar 3.8

Rigidbody 2D

Rigidbody mensimulasikan gerakan, gravitasi, dsb yang berhubungan dengan fisika Gameobject. Oleh karena itu Rigidbody berguna untuk memudahkan user dalam membuat gerakan untuk karakter-karakterny. Rigidbody juga berguna untuk mengatur kolisi antara 2 Gameobject sehingga Developer tidak perlu mengkode kolisi secara manual. 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 sering digunakan karena gravitasi, percepatan, perlambatan dsb. akan dikalkulasi terus dalam Dynamic.
* Simulated : Memilih bila fisika ingin disimulasikan dalam Gameobject, hanya memiliki pengaruh bila Gameobject memiliki Body Type Dynamic.
* Mass : Massa dari Gameobject untuk dihitung dalam fisika Gameobject.
* Linear & Angular Drag : Variabel yang menentukan hambatan udara untuk gerakan dan rotasi Gameobject dimana bila nilai semakin besar, maka Gameobject akan semakin cepat berhenti.
* Gravity Scale: Kekuatan gravitasi pada Gameobject ini.
* Collision Detection : Menentukan seberapa sering kolisi dideteksi dalam sebuah Gameobject.
* Sleeping mode : Menentukan keaktifan Rigidbody saat objek dibuat pertama kali.
* Interpolate : Menentukan bila gerakan dari Gameobject ingin diinterpolasi atau tidak.
* Freeze Position : Mencegah objek dari bergerak pada sumbu yang dipilih.
* Freeze Rotation : Mencegah objek untuk berrotasi.
  1. Collider 2D

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

Gambar 3.9

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

Collider 2D bekerja baik dengan Rigidbody 2D bila user ingin objek bisa menabrak/ditabrak objek lain, dimana Collider membuat sebuah bentuk yang akan menjadi “tubuh” objek yang bisa ditabrak sesuai dengan gambar 3.9b yang diindikasikan 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 Gameobject agar bentuk kolisi bisa lebih spesifik sesuai dengan kehendak para Developer.

Untuk variabel yang diperlukan, hanya 2 yang bersifat penting. 2 Variabel tersebut adalah Edit Collider yang bila diklik dapat mengubah kotak hijau di gambar 3.9b sesuai dengan kehendak Developer. Variabel kedua yang penting adalah Is Trigger, dimana bila dicentang, semua Gameobject meskipun bila memiliki Rigidbody 2D dan Collider 2D akan melewati Gameobject tersebut, dimana akan dilakukan fungsi onTriggerEnter daripada onCollisionEnter, fungsi ini sangat berfungsi bila Gameobject tidak ingin memiliki kolisi tapi bila disentuh bisa melakukan sesuatu.

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 bekerja sesuai dengan apa yang tertera dalam Script. Unity telah menyediakan API yang memudahkan Developer dalam membuat Script sebuah Gameobject. Tidak hanya itu, Script yang menuruni Class Monobehaviour dapat mengintergrasikan isi dari Script menuju Editor Unity, sepeti bila variabel bersifat publik maka variabel tersebut dapat diisi melalui Editor Unity, atau dengan memberi [SerializeField] sebagai pengganti public agar bisa dimasukkan melalui Editor Unity, tapi dianggap private oleh Script lain.

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

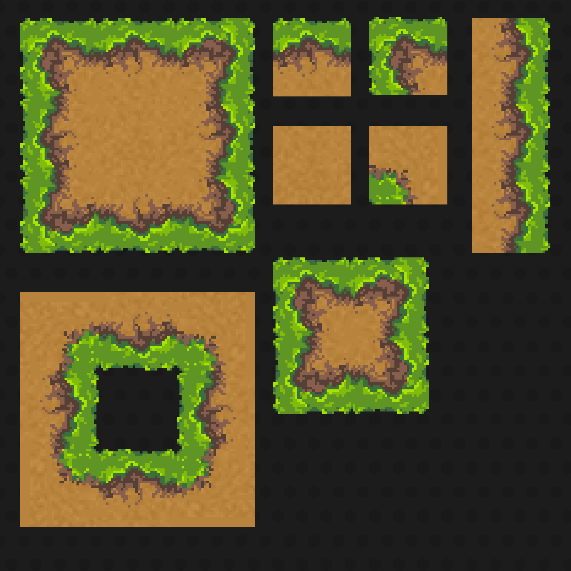
Gambar 3.10

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

Selain variabel, juga terdapat beberapa fitur lain, seperti GetComponent<Component>(). GetComponent dapat mengambil sebuah komponen yang dimiliki objek dan dapat mengubah property komponen sesuai kebutuhan. Dan terakhir adalah API-API penting yang dapat membantu dalam Development aplikasi Unity, yaitu :

* Awake : Fungsi yang dijalankan saat Script dibuat
* Start : Fungsi yang dijalankan saat frame pertama objek ada dalam Scene
* Update : Fungsi yang dijalankan setiap frame
* FixedUpdate : Dijalankan setiap kali fisika 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 Asynchronous dimana sebuah Coroutine bisa menggunakan delay.

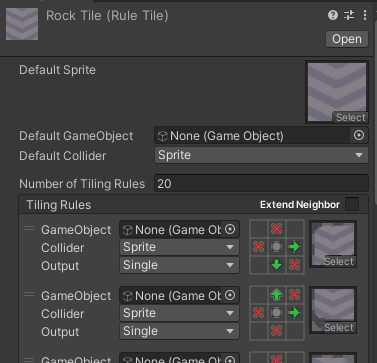
1. Tilemap & Rule Tile



Gambar 3.11

Contoh Tilemap

Tilemap adalah sebuah kumpulan tile yang dapat digunakan untuk membuat tampilan dari sebuah Map. Gambar 3.11 diatas merupakan sebuah Tilemap untuk tanah, dimana setiap Tile yang terdapat di dalam sebuah Tilemap digunakan untuk membuat sebuah Map yang berisi Tile-Tile tanah diatas. Kita dapat menggunakan Tile Palette untuk mengubah gambar Tilemap diatas menjadi sebuah palet lukis Tile siap dipakai.



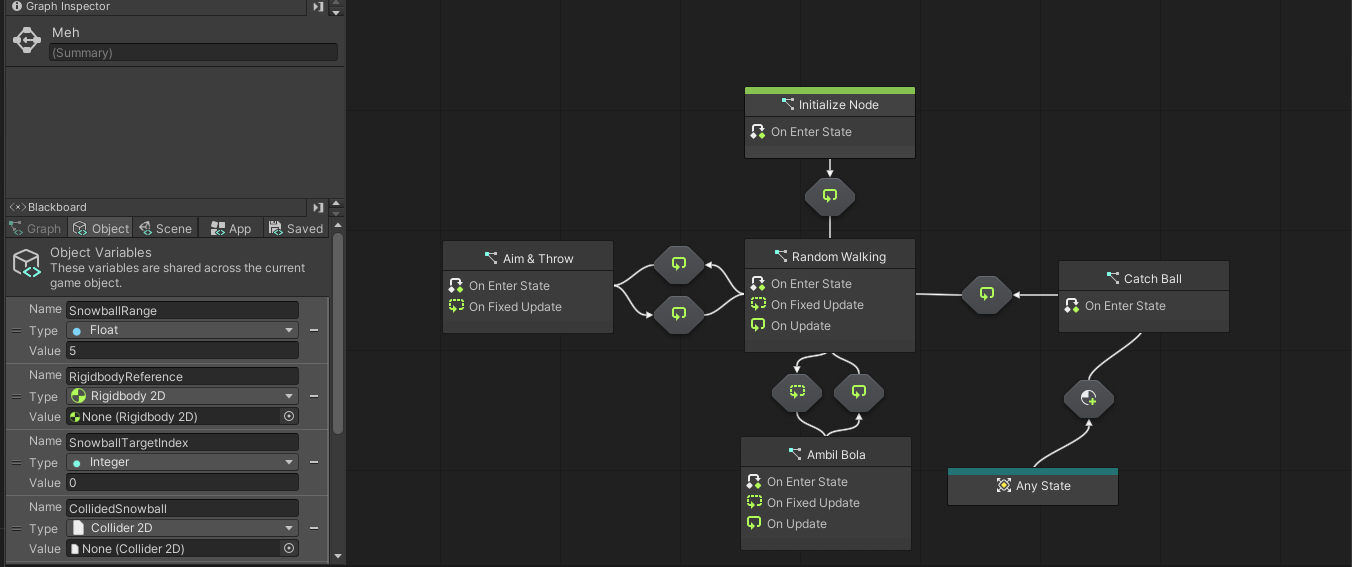
Gambar 3.12

Rule Tile

Tapi bila level yang dibuat bersifat dinamis atau dibuat menggunakan Script, maka akan sulit menempatkan Tile yang tepat sesuai dengan Tile di sekitarnya, sehingga terdapat Tilerule, sebuah Tile yang akan beradatapsi sesuai dengan penempatan Tile tersebut terhadap sekitarnya. Seperti pada contoh gambar 3.12 diatas, terdapat 20 tile rule, sehingga ada 20 tile yang bisa keluar dari 1 tile rule. Lalu untuk rule tersebut dilihat 8 tile di sekitar Ruletile seperti yang bisa dilihat di Tiling Rules pada gambar 3.12.

Terdapat 3 pilihan untuk menempatkan kondisi tile sekitar menggunakan Rule paling atas di dalam Tiling Rule sebagai contoh. Pertama adalah netral, dimana Rule Tile tidak melihat apabila ada Tile atau tidak di lokasi tersebut, disimbolkan dengan kotak kosong seperti pada kiri atas, kanan atas dan kiri bawah dari Rule. Pilihan kedua adalah terdapat sebuah Tile, dimana pilihan ini yang menunjukkan harus ada Tile di kotak yang ditentukan ditampilkan menggunakan sebuah anak panah seperti yang bisa dilihat dari Rule paling atas mengarah ke kanan dan bawah. Pilihan terakhir adalah tidak ada Tile, dimana di kotak yang ditentukan tidak boleh ada Tile ditampilkan dengan sebuah simbol X merah.

1. State Machine



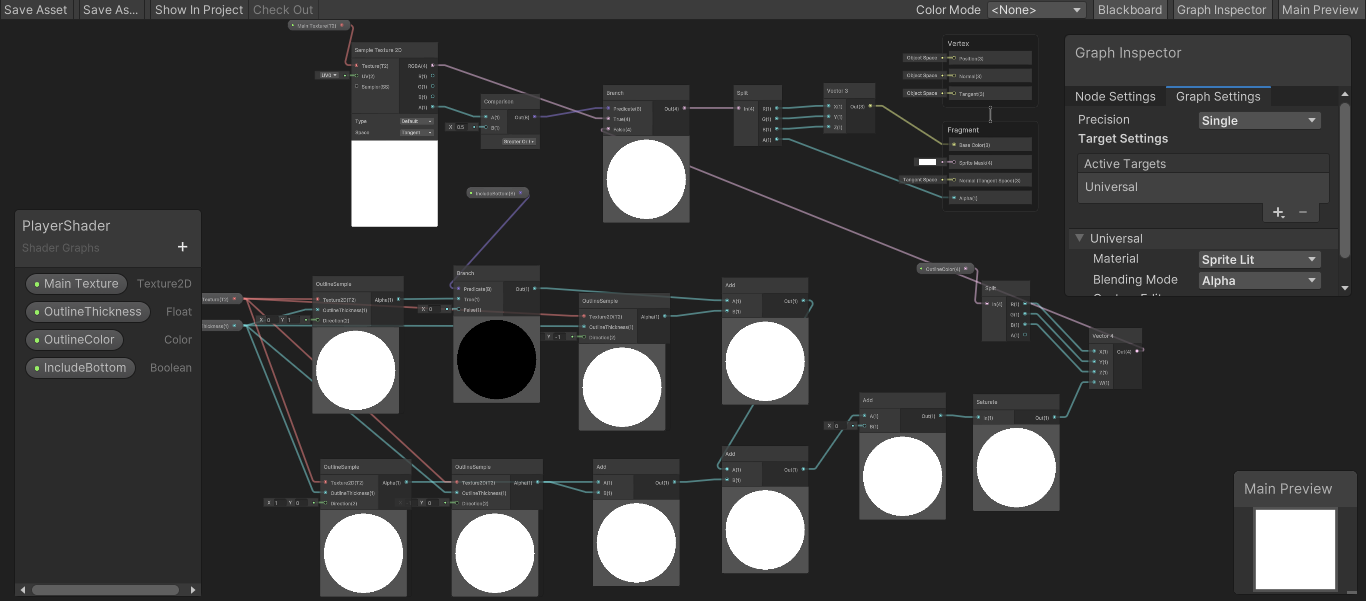
Gambar 3.13

State Machine

Bila tidak ingin membuat Script menggunakan C#, Unity juga menyediakan sebuah metode Programming visual yang berupa State Machine. Di gambar 3.13 diatas, di kiri bawah adalah variabel-variabel yang tersedia, sementara layar bagian 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 perubahan State terpenuhi, berbeda dengan Animator, pengecekan transisi State Machine dilakukan melalui fungsi yang disediakan Unity seperti Update atau OnTriggerEnter dimana animator secara otomatis mengecek setiap Frame.

1. Shader Graph

Shader Graph digunakan untuk membuat Shader sendiri, dimana Shader adalah sebuah fungsi komputer yang digunakan 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, yang dapat dipasang ke dalam komponen Sprite Renderer sebuah Gameobject. Setelah dipasang, Shader tersebut akan tampil pada Gameobject yang ditentukan, 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