**LAPORAN UJIAN AKHIR SEMESTER**

**PEMORGRAMAN GAME**

**BRICKS BALL SHOOTER**

Oleh:

Edward Hendryawan Michael 311610005

Reinaldo Sebastian Gunawan 311610015



Program Studi Teknik Informatika

Universitas Ma Chung

2019

**1.1 Judul dan Genre Game**

Game yang dibuat untuk memenuhi ujian akhir semester pemrograman *game* bernama *Bricks Ball Shooter* memiliki *genre* *arcade* dan *casual* sehingga cocok untuk dimainkan di waktu senggang tanpa terkendala waktu dan tempat.

**1.2 Platform Pembuatan**

Platform yang digunakan untuk membuat *game* ini adalah *Unity* versi 2018.3.6.f1. *Game* ini dibuat dengan kapabilitas *cross-platform*, sehingga dapat diinstall dan dimainkan di beberapa berangkat yang berbeda.

**1.3 Deskripsi Game**

Pemain harus menghancurkan balok( atau disebut *brick*) yang bermunculan dengan menembakkan bola sebagai alat untuk menghancurkan balok tersebut. Balok yang bermunculan memiliki sejumlah nyawa untuk dihancurkan dengan menembaknya sejumlah nyawa yang dimiliki. Setiap kali pemain selesai melakukan 1 tembakan maka nyawa balok baru yang bermunculan akan terus bertambah seiring dengan naiknya level. Pemain dapat menembakkan bola kepada objek *power up* untuk menambah jumlah bola yang dapat ditembakkan agar semakin cepat menghancurkan balok tersebut. *Game* akan berakhir jika balok sudah mencapai pada suatu batas tertentu sehingga pemain harus selalu waspada dengan balok-balok yang ada di dekatnya.



Gambar1.1 Tampilan *game (player* menembakkan bola berwarna merah dengan jumlah 1x, balok dengan nyawa yang dimiliki, bola putih adalah *power up*)

**1.5 Sumber Asset**

Untuk asset-asset yang digunakan kami ambil dari berbagai sumber dan kami pastikan merupakan *free license* agar menghindari masalah hak cipta. Untuk tampilan GUI yang digunakan kami menggunakan asset bertema ruang angkasa yang diambil dari situs craftpix.net (<https://craftpix.net/freebies/free-space-shooter-game-gui/?utm_source=itch&utm_medium=public&utm_campaign=gui>).



Gambar1.2 Beberapa asset GUI yang digunakan dari situs craftpix.net

Untuk asset balok yang digunakan diambil dari situs OpenGameArt.Org (<https://opengameart.org/content/sj-frame>). Dari 2 balok yang ada pada 1 gambar, hanya digunakan salah satu untuk membuat balok. Dengan bantuan dari *Unity* dapat dilakukan pemotongan gambar menjadi 2 *sprite* gambar yang berbeda secara otomatis. Pada *gameplay* warna-warna balok yang dihasilkan dapat berbeda-beda dikarenakan pengaruh dari *script*. Untuk asset bola dan *power up* digunakan asset *basic* lingkaran dari *Unity* dan diubah warnanya sesuai yang dibutuhkan.



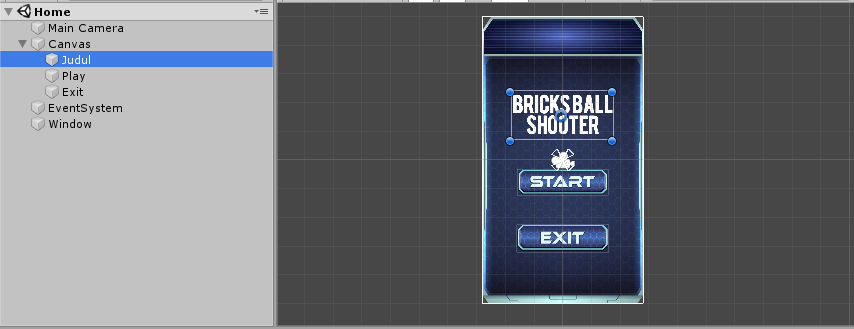
Gambar1.3 Asset balok yang digunakan(kiri) dari situs OpenGameArt.Org

Lagu yang digunakan sebagai BGM(*Background Music*) diambil dari *youtube* yang dipublikasikan oleh Sound R. Dengan judul *Brezee* (<https://www.youtube.com/watch?v=5GfW9eIGcpY&t=19s>). Lalu untuk *sound effect* yang digunakan ketika bola mengenai balok diambil dari situs freesound.org dengan judul Pop 19.wav yang dipublikasikan oleh debsound (<https://freesound.org/people/debsound/sounds/320549/>).

**1.6 Proses Pembuatan**

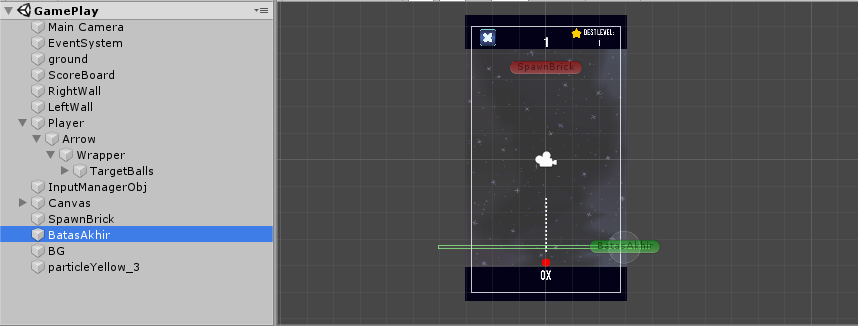
Dimulai dengan menentukan tema dan *game* seperti apa yang ingin dibuat yang akhirnya memunculkan ide untuk membuat *game* berdasarkan *brick breaker*. Kemudian memutuskan menggunakan *Unity* sebagai *platform* pembuatan *game* karena kemampuan *cross-platform* dan memiliki banyak fitur yang dapat membantu proses pembuatan *game*. Lalu memulai mencari *aset* yang dibutuhkan, proses pencarian asset dilakukan beberapa kali demi menemukan kesesuaian dan keseimbangan tampilan.

Proses selanjutnya adalah memulai dengan membangun tampilan dasar dari tiap *scene* dalam *game*. Ukuran layar yang digunakan untuk *game* ini adalah 720×1280. Dimulai dari membangun tampilan *home* dari *game* seperti pada gambar di bawah ini.



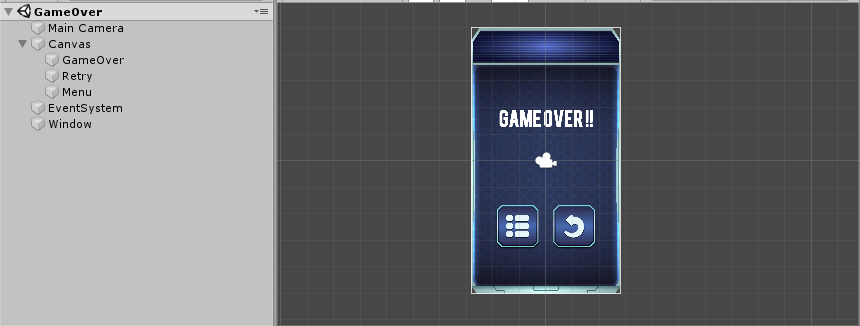
Gambar1.4 Struktur tampilan *home* pada *game*

Kemudian dilanjutkan dengan *scene* utama yaitu *scene gameplay*. Dalam *scene* ini paling banyak membutuhkan *script* agar permainan dapat dilakukan dengan benar dan berjalan dengan baik. Untuk struktur tampilannya dapat dilihat pada Gambar1.5. Tidak semua yang ada pada tampilan struktur akan tampak ketika *game* dijalankan, seperti pada titik *SpawnBrick* dan BatasAkhir. Meskipun tidak tampak pada layar ketika dijalankan namun tetap memiliki peranan penting, penjelasasn lebih jelasnya ketika pembahasan *source code*.



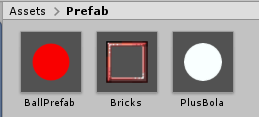
Gambar1.5 Struktur tampilan *gameplay* pada *game*

Dan *scene* terakhir yang disusun adalah *scene* ketika *game over*. Scene *game over* dimunculkan ketika balok-balok mencapai pada *game object* BatasAkhir. Strukturnya dapat dilihat pada gambar berikut ini.



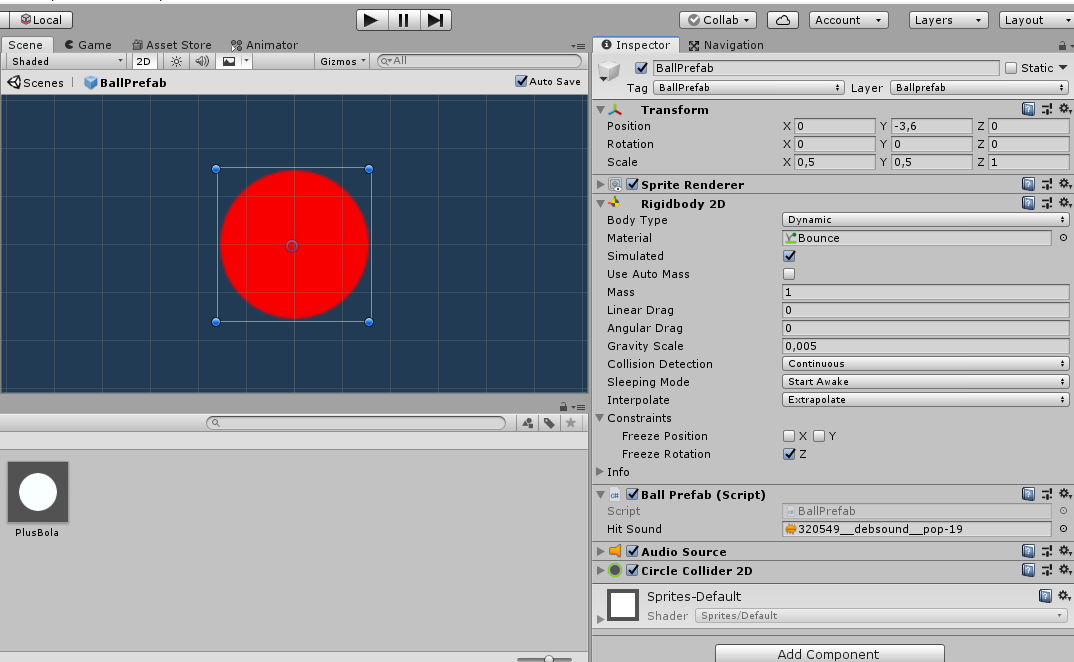
Gambar1.6 Struktur tampilan *game over* pada *game*

Setelah membuat struktur tampilan kemudian akan dibuat objek-objek yang akan digunakan yaitu bola dan balok. Semua objek yang digunakan akan disimpan menjadi sebuah *prefabs* yang nantinya akan berguna untuk pembentukan objek-objek yang diperlukan. Setiap *prefabs* memiliki strukturnya masing-masing sesuai dengan kebutuhannya.



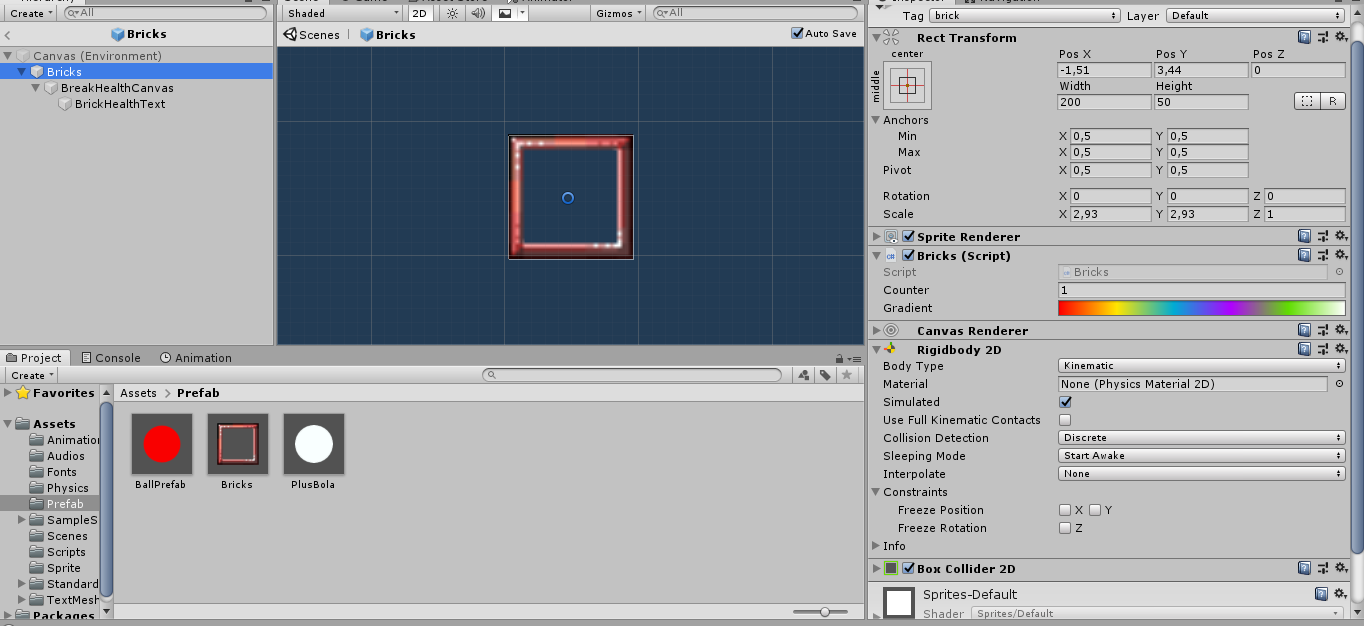
Gambar1.7 *Prefab* yang diperlukan dalam *game*

Untuk masing-masing struktur *prefab* akan dibahas, dimulai dari *BallPrefab* yang digunakan sebagai bola yang ditembakkan oleh pemain. Didalam *BallPrefab* tersusun dari *RigidBody2D*, *BallPrefab(Script), Audio Source,* dan *CircleCollider2D*. *RigidBody2D* berisikan tentang berbagai hukum fisika seperti berat, gravitasi, kecepatan, bagaimana bola memantul, dll.  *BallPrefab(Script)* adalah *script* untuk mengatur berbagai tindakan *game object BallPrefab* ini. *Audio Source* untuk mengatur suara. Kemudian ada *CircleCollider2D* yang berguna untuk mendeteksi adanya sebuah benturan atau tabrakan dengan objek lainnya. Khusus pada *game object BallPrefab* ini diatur *layer* pada *layer* 8 dengan nama layer *BallPrefab.* Label yang digunakan pada objek ini adalah *BallPrefab*.



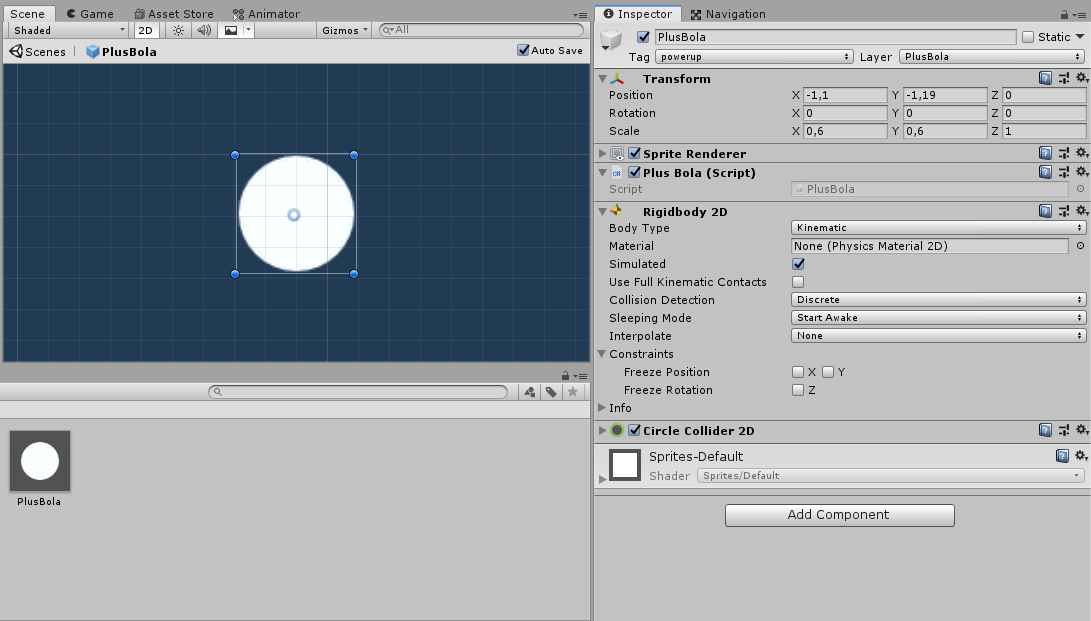
Gambar1.8 Struktur *prefab BallPrefab*

Kemudian untuk *BricksPrefab* strukturnya berisikan *RigidBody2D, Bricks(Script),* dan *BoxCollider2D*  yang berfungsi sama seperti *Circle Collider2D* hanya bentuknya yang membedakan.



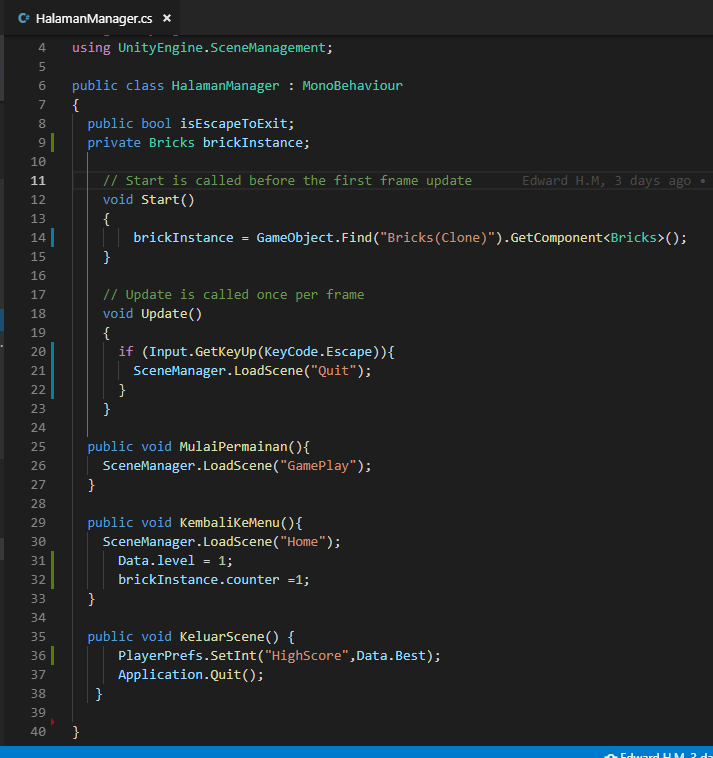
Gambar1.9 Struktur *prefab BrickPrefab*

Kemudian untuk *PlusBola* strukturnya berisikan *RigidBody2D, PlusBola(Script),* dan *Circle Collider2D*. Label objek ini adalah *powerup*. Masing-masing objek memiliki keunikannya dan fungsi sendiri dalam game ini.



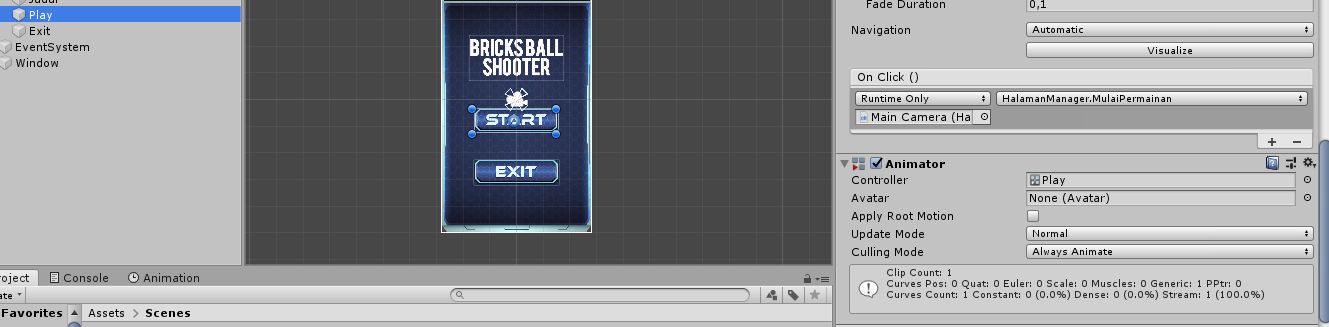
Gambar1.10 Struktur *prefab PlusBola*

Pembahasan selanjutnya adalah *source code* yang dibuat. *Source code* yang dibahas pertama adalah HalamanManager.cs yang berisikan funtion-function untuk mengatur perpindahan *scene*. HalamanManager.cs ini diletakkan pada setiap objek *Main Camera* yang ada pada *scene,* berikut struktur dari HalamanManager.cs.



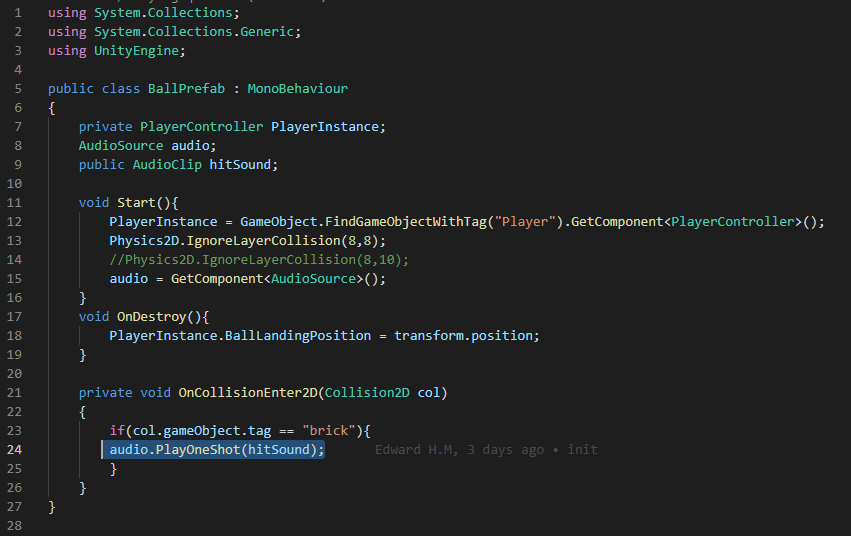
Gambar1.11 struktur dari HalamanManager.cs

Beisikan banyak function yang berguna untuk mengatur perpindahan *scene,* beberapa *function* ini dapat dijalankan melalui objek *button* ketika di klik dengan cara sepeti Gambar1.12. *Button* tersebut ketika di klik akan menjalankan *function* MulaiPermainan() yang menjalankan tugas untuk memanggil(*load*) *scene gameplay*. Application.Quit() berfungsi untuk menutup aplikasi ketika keluar. *PlayerPrefs* dapat digunakan untuk menyimpan atau memasang suatu nilai. Pada *Start*() berisikan *brickInstance* yang akan mencari objek dengan nama Brick(Clone) dan mengambil komponen dari objek tersebut.

 Gambar1.11 Penggunaan salah satu function di HalamanManager.cs

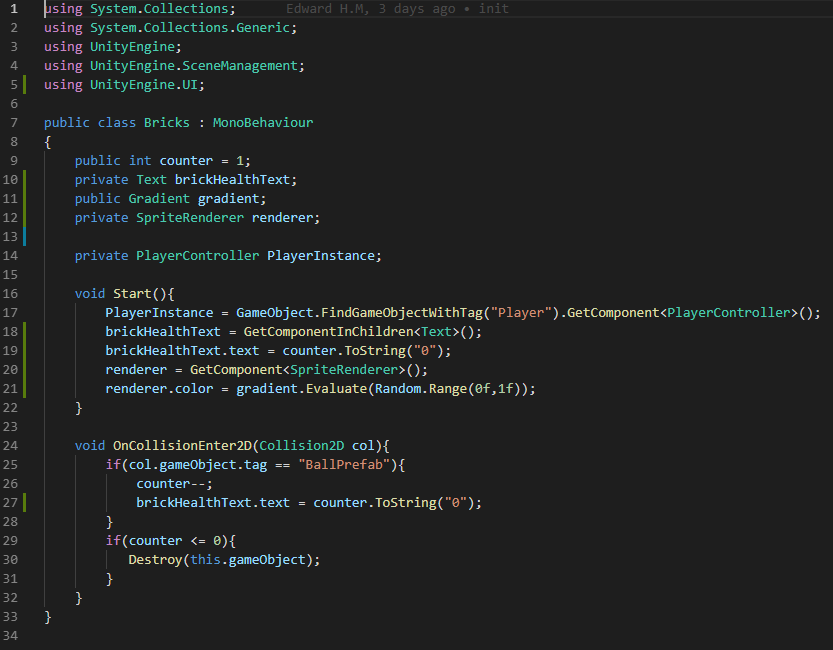
Selanjutnya ada BallPrefab*.*cs yang berisikan pengaturan dari *game object* *BallPrefab.* Pada Gambar 1.12 dapat dilihat pada *Start*() akan mencari *game object* dengan label *Player* dan diambil komponennya. Lalu *game object* *BallPrefab* diatur agar mengacuhkan tabrakan antara layer 8 dengan layer 8 tujuannya agar ketika bola ditembakkan dengan jumlah banyak, maka bola tersebut tidak bertabrakan satu bola dengan yang lainnya. Kemudian juga mengambil komponen dari sumber audio yang diinputkan. Ada *function OnDestroy*() yang berguna untuk mendapatkan nilai posisi bola mendarat dan menghancurkan objek *BallPrefab(Clone)*. Sebagai informasi *prefabs* ketika dipanggil kedalam *scene* maka akan menjadi *game object* dengan akhiran (*Clone*).

Terdapat *function* yang mengatur ketika bertubrukan dengan objek berlabel brick maka akan dimainkan *sound* yang menjadi inputannya.



Gambar1.12 Struktur BallPrefab*.*cs

Berikutnya ada Bricks*.*cs yang berisikan pengaturan dari *game object* *Bricks.* Pada Gambar 1.13 dapat dilihat pada *Start*() akan mencari *game object* dengan label *Player* dan diambil komponennya. Ada *brickHealthText* yang mencari komponen *text* dalam *canvas* yang menjadi *children* dari *prefabs* ini. Kemudian brickHealthText,text mengatur text yang dimunculkan sesuai dengan *counter*(nyawa *bricks*). Untuk membuat warna *random* digunakan *render.color* yang mengambil nilai acak dari rentang 0f hingga 1f, untuk warna dari rentang tersebut diatur oleh *developer* sesuai kebutuhan seperti pada Gambar1.9. Ketika bertabrakan dengan objek berlabel *BallPrefab* maka counternya akan berkurang dan tulisan pada *bricks* akan ikut menyesuaikan. Ketika *counter* mencapai 0 maka objek *Bricks* tersebutakan dihancurkan.



Gambar1.13 Struktur Bricks*.*cs

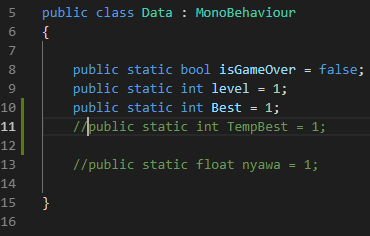
.

Berikutnya ada PlusBola*.*cs yang berisikan pengaturan dari *game object* *Bricks.* Pada Gambar 1.14 dapat dilihat pada *Start*() akan mencari *game object* dengan label *Player* dan diambil komponennya. Ketika objek dengan *script* ini tertrigger maka maka jumlah bola ketika akan ditembakkan akan bertambah 1, kemudian objek ini akan dihancurkan. Bola tidak akan terpental ketika mengenai objek dengan *script* ini melainkan akan menembusnya karena CircleCollider2Dnya dimatikan.



Gambar1.14 Struktur PlusBola*.*cs

Terdapat *script* yang hanya digunakan untuk menampung beberapa variabel nilai, *script* tersebut adalahData.cs. Struktur Data.cs dapat dilihat pada Gambar1.15.



Gambar1.15 Struktur Data*.*cs

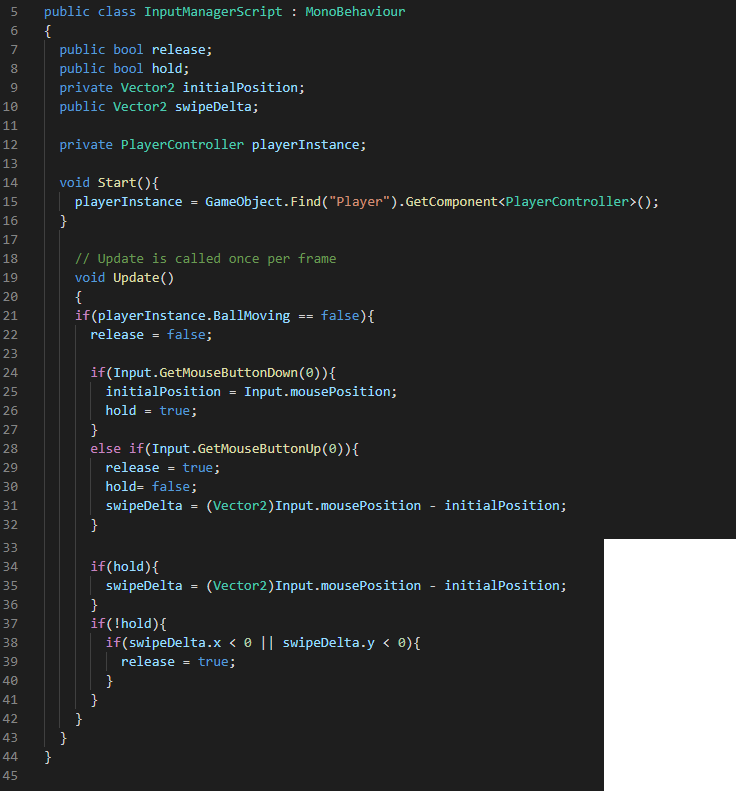
Untuk memunculkan *scene* *GameOver* maka pada object BatasAkhir diberikan *script* BatasAkhir.cs. Pertama ketika *Start*() mencari objek bernama *Bricks(Clone)* dan mengambil komponennya. Ketika ada objek yang berlabel *brick* mentrigger objek BatasAkhir maka akan memanggil *scene GameOver* dan mereset kembali level serta awal mulai counter pada balok menjadi 1 kembali.



Gambar1.16 Struktur BatasAkhir*.*cs

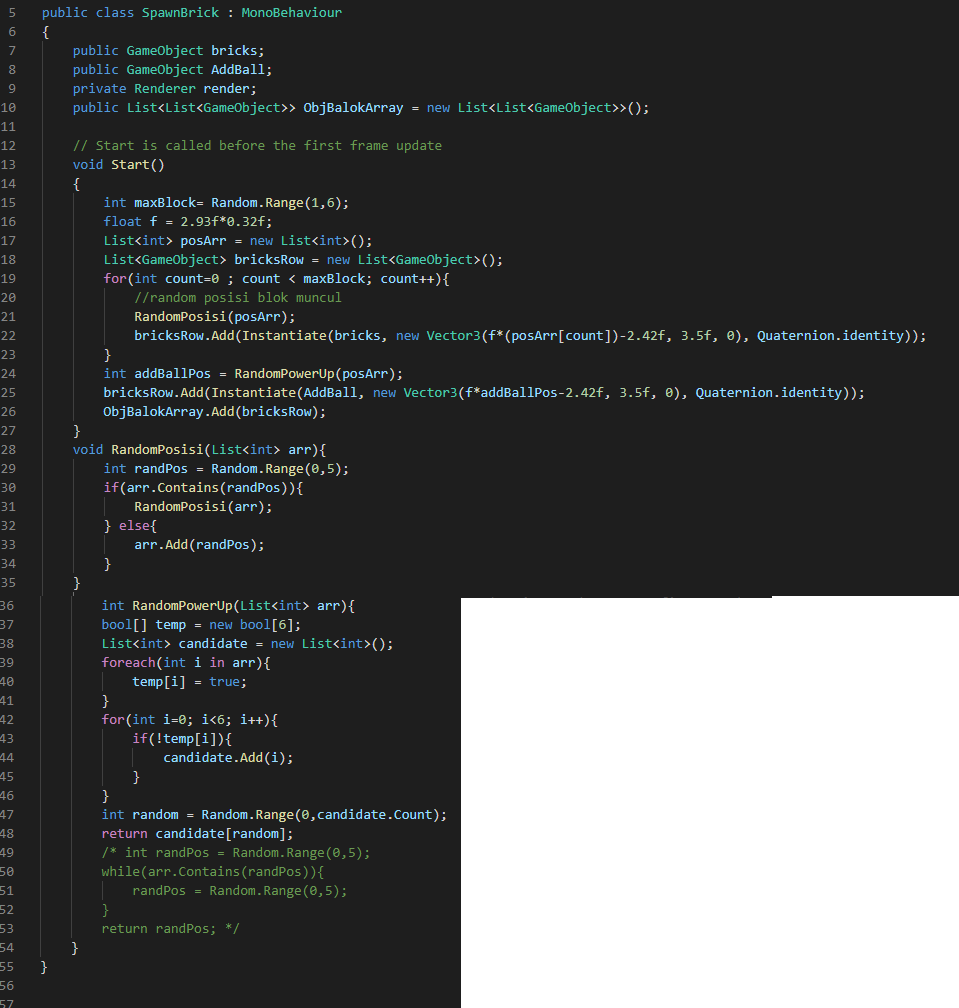
Untuk mengatur inputan arah dari *mouse* digunakan dibuatlah *script* InputManagerScript.cs. Dalam script ini untuk mengetahui apakah bola sedang diam, ditembakkan, dan mendapatkan arah ketika ditembakkan, menggunakan mouse yang sedang ditahan, maupun ketika dilepas dilihat pada *Start*() akan mencari *game object* dengan label *Player* dan diambil komponennya. Untuk lebih detailnya dapat dilihat pada Gambar1.17.

Selanjutnya akan membahas untuk memunculkan balok/*bricks* dan *power up* pertama kali dengan *script* SpawnBrick.cs. Untuk menentukan posisi dari *bricks* digunakan *random* untuk mengacak panjang 1 baris balok dengan posisi antara 0-5, jika pada posisi tertentu sudah terisi maka akan dirandom kembali.



Gambar1.17 Struktur BatasAkhir*.*cs

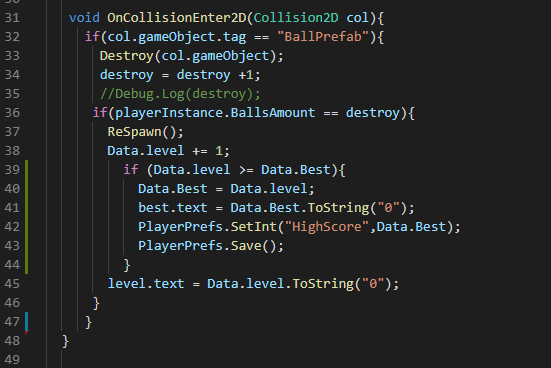
Begitu pula untuk posisi *power up* dintetukan terlebih dahulu apakah dalam 1 baris tersebut masih ada yang kosong, jika ada maka dia akan diletakkan pada tempat yang kosong. Start() akan memunculkan objek prefabs dengan posisi yang sudah didapatkan tadi. Secara keseluruhan 1baris akan ada balok dan *power up*, seluruh nilai tersebut ditampung kedalam bricksRow yang merupakan *array list*. Kemudian 1 bricksRow akan ditampung lagi kedalam ObjBalokArray yang merupak *array list* yang berisikan dari banyak bricksRow. Untuk struktur lebih jelasnya dapat dilihat dalam Gambar1.18

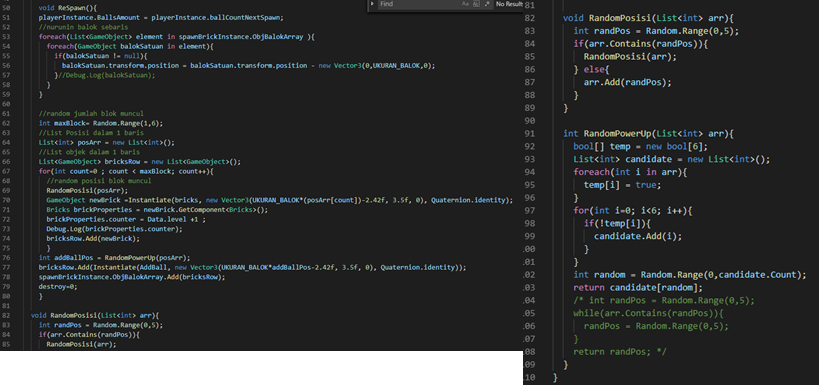


Gambar1.18 Struktur SpawnBrick*.*cs

Script yang mengatur jalannya permainan hampir semuanya ada pada ground.cs. Pada ground.cs terdapat *function ReSpawn*() hampir bekerja sama seperti SpawnBrick.cs namun sebagai tambahan bertugas untuk mengubah posisi y dari tiap baris balok menjadi turun. Pada ground.cs juga mengatur kenaikan level serta *counter* pada balok, kemudian menyimpan dan memanggil *high score* juga.

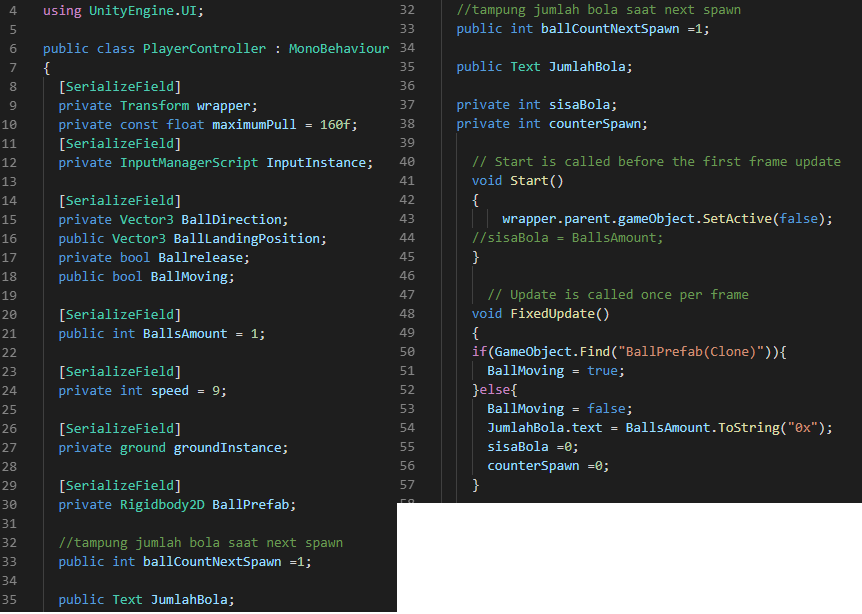


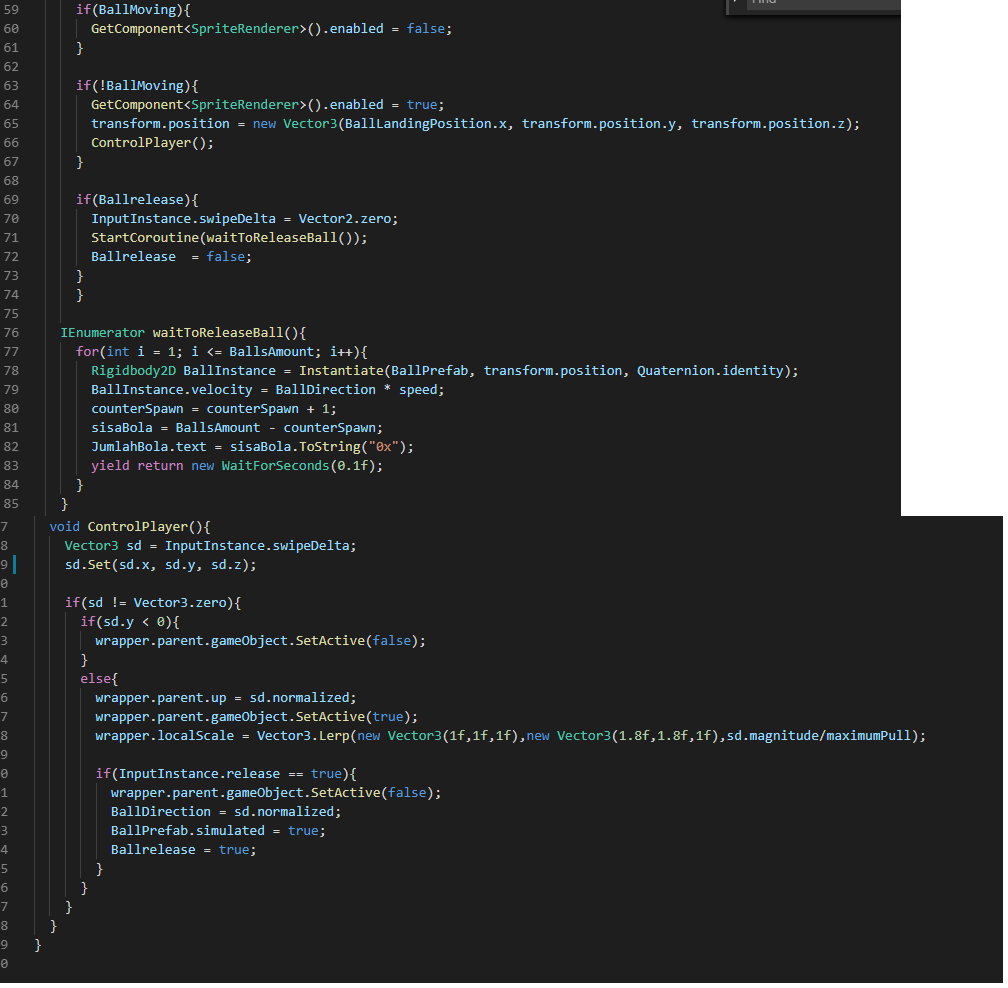




Gambar1.19 Struktur ground*.*cs

Script terakhir adalah PlayerController.cs, yang mengatur bagaimana pemunculan garis bantu untuk menembak dan pemunculan dari BallPrefab. Selain itu juga mengubah tulisan bola yang dimiliki dan ketika menembak dapat mengetahui tinggal tersisa berapa bola lagi yang akan dihasilkan dalam 1 kali tembakan.





Gambar1.20 Struktur PlayerController*.*cs

**1.7 Link URL**

**Download apk :**

<https://drive.google.com/file/d/1xGgUfSomDIMh8fl8iIafmGA3qBew4Rrb/view>

**Download Source Code & Asset :**

<https://drive.google.com/file/d/1_tkt4D1TC6PB7Er6uwZcURpK3NE6quC-/view>

**YouTube :**

<https://www.youtube.com/watch?v=bkk5UyA3a38&feature=youtu.be>