Membuat Game Android dengan Unity 3D

Edy Winarno ST, M.Eng, Ali Zaki, & SmitDev Community

PENERBIT PT ELEX MEDIA KOMPUTINDO

KOMPAS GRAMEDIA

Membuat Game Android dengan Unity 3D

Edy Winarno ST, M.Eng, Ali Zaki, & SmitDev Community

Diterbitkan pertama kali oleh
Penerbit PT Elex Media Komputindo
Kelompok Gramedia, Anggota IKAPI, Jakarta 2015

715052258
ISBN: 978-602-02-7718-9

[eEp]

Dicetak oleh Percetakan PT Gramedia, Jakarta
Isi di luar tanggung jawab percetakan

KATA PENGANTAR

Berkat Android, kini ponsel tidak hanya bisa dipakai untuk bertelepon, tapi juga bisa digunakan untuk bermain game. Penggunaan Android untuk bermain game pun makin merata, terutama di kalangan anak-anak dan remaja.

Unity 3D sendiri adalah game engine yang cukup populer. Game engine merupakan sebuah software yang digunakan untuk membuat atau mengembangkan game. Unity 3D banyak digunakan oleh kalangan pembuat game indie maupun orang yang baru belajar membuat game.

Buku ini menjelaskan bagaimana menggunakan Unity 3D untuk membuat game. Anda akan dipandu langkah demi langkah dalam membuat game, hingga menghasilkan sebuah game yang siap dimainkan di Android.

Penulis menyadari bahwa buku ini tidak luput dari kesalahan dan masih jauh dari sempurna. Untuk itu penulis membuka diri untuk segala bentuk tanggapan dan pertanyaan pembaca berkaitan dengan buku ini. Untuk berkonsultasi secara langsung dengan penulis, silakan layangkan email ke winarno@smitdev.com atau ali@smitdev.com.

Semarang, September 2015

Edy Winarno ST, M.Eng

Ali Zaki

SmitDev Community

Daftar Isi

Kata Pengantar v
Daftar Isi vii

Bab 1 Mengenal Unity 3D 1
1.1 Game Engine 1
1.2 Unity 3D 3
1.3 Instalasi Unity 3
1.4 Menggunakan Unity 10
1.4.1 Menu 15
1.4.2 Shortcut 15
1.4.3 Jendela Lembar Kerja 18
1.5 Unity dengan Game Android 20

Bab 2 Rancangan Awal Game Survival 23
2.1 Mengenal Game Survival 23
2.2 Membuat Peta dan Menguji Pergerakan Karakter 24
2.3 Menambahkan Musuh dan Sistem Membunuh Musuh 35
2.4 Membuat Animasi Senjata 46

Bab 3 Memperbaiki Tampilan Peta 57
3.1 Membuat Efek Kabut 57
3.2 Mengganti Bentuk Peta 59
3.3 Menambahkan Gunung 65
3.4 Menambahkan Material 69
3.5 Membuat Objek Pohon 80

Bab 4 Membuat Sistem Game 89
4.1 Menambahkan Pilihan Senjata 89
4.2 Membuat Sistem Kendali Musuh 98
4.3 Menambahkan Suara 102

4.4 Meningkatkan Sistem Kendali Musuh 107
4.5 Membuat Nyawa pada Karakter 115
4.6 Membuat Pintu 120
4.7 Membuat Menu Respawn 128
4.8 Membuat Karakter Terluka Saat Jatuh 134
4.9 Membuat Perubahan Siang Malam 141
4.10 Memperbaiki Desain Tampilan Peta 147
4.11 Perbaikan Objek pada Peta 156

BAB 5 FINISHING 159

5.1 Menambahkan Tangan untuk Karakter 159
5.1.1 Membuat Tangan Tidak Dapat Menembus Objek 166
5.1.2 Menambahkan Animasi Gerakan Tangan 169
5.1.3 Menambahkan Senjata pada Tangan 179
5.2 Memperbaiki Sistem Senjata 183
5.3 Membuat Game Menjadi Stand Alone 191
5.4 Kompilasi Game untuk Android 197
5.4.1 Menginstal JDK 197
5.4.2 Menginstal SDK 198
5.4.3 Compile Game Menjadi APK 202

TENTANG PENULIS 208

1 MENGENAL UNITY 3D

Unity adalah game engine yang cukup dikenal. Game engine ini telah banyak digunakan baik dari kalangan pelajar maupun penggiat game indie. Anda dapat menggunakan Unity untuk membuat game 2D atau 3D. Pada bab ini Anda akan dikenalkan terlebih dahulu apa itu Unity dan dasar-dasar cara menggunakannya.

1.1 Game Engine

Game engine adalah sebuah software dengan framework yang digunakan untuk membuat atau mengembangkan game. Banyak hal yang dimiliki game engine untuk membuat game, seperti membuat gambar karakter, efek tabrakan, koding, musik, animasi, dan lain sebagainya.

Kelebihan membuat game menggunakan game engine adalah Anda akan lebih mudah dalam membuat game. Hal ini karena game engine telah menyediakan berbagai kebutuhan untuk membuat game. Selanjutnya, Anda hanya perlu mengatur jalannya game seperti yang Anda inginkan. Game yang dapat dibuat oleh game engine tidak hanya game untuk komputer, tetapi juga dapat digunakan untuk konsol atau mobile.

Ada banyak game engine saat ini. Bagi Anda yang tertarik membuat game dengan game engine, Anda dapat mencoba beberapa game engine berikut:

- Unity
- Cry Engine
- Construct
- GameMaker Studio
- RPG Maker
- Unreal
- Luminous Studio
- Crystal Tool

Gambar 1.1 Final Fantasy XV menggunakan Luminous Studio

Dari uraian di atas, Anda dapat melihat banyak sekali game engine. Dari semua game engine, untuk bisa membuat game ada persamaan syarat yang diperlukan, yaitu Anda perlu memahami script dari setiap game engine. Hal ini karena, Anda mengatur berjalannya game dengan menggunakan script.

Pada buku ini Anda akan dijelaskan cara membuat game menggunakan Unity. Game engine ini merupakan game engine yang cukup banyak digunakan oleh pembuat game indie maupun orang yang baru belajar membuat game.

1.2 Unity 3D

Unity memang bukan game engine yang digunakan untuk membuat game kelas atas seperti Final Fantasy XV yang sedang dibuat menggunakan Luminous atau Far Cry yang menggunakan Cry Engine. Akan tetapi, bagi Anda yang ingin belajar membuat game, Unity merupakan pilihan yang tepat. Hal ini karena Unity memberikan kebebasan Anda untuk membuat game yang Anda inginkan. Selain, itu Unity juga cukup kompleks untuk membuat berbagai macam jenis game. Sebut saja Anda dapat membuat game dengan tema balap, pertarungan, olah raga, atau role playing.

Kelebihan utama dari Unity yang cocok bagi Anda yang belajar membuat game adalah tersedianya versi gratis. Selain versi gratis, Unity juga menyediakan asset store yang berisi asset atau perlengkapan untuk membuat game yang dapat langsung digunakan. Asset yang tersedia di Unity antara lain bentuk karakter, tampilan peta, skrip, dan lain sebagainya.

Gambar 1.2 Deus EX: The Fall yang dibuat menggunakan Unity

1.3 Instalasi Unity

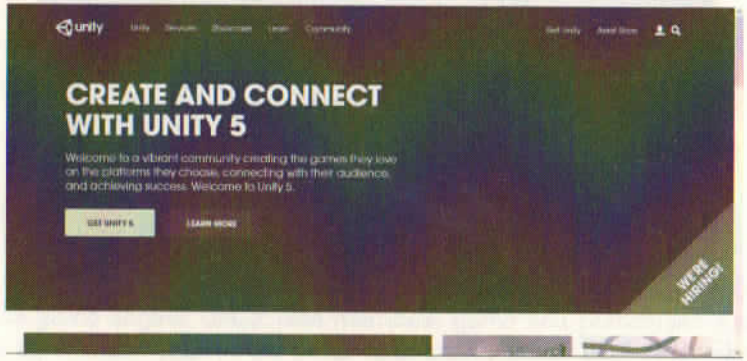
Sebelum Anda menginstal Unity, Anda perlu mengetahui spesifikasi komputer yang dibutuhkan untuk menginstal Unity. Berikut spesifikasi untuk menginstal Unity:

- Sistem operasi Windows XP SP2+, 7 SP1+, 8, 10 dan Mac OS X 10.8+. Tidak termasuk Windows Vista. Sedangkan pada Windows dan OS X server belum dicoba.
- Kartu grafis yang mendukung Direct X 9 atau kartu grafis setelah 2004.

Setelah Anda mengetahui spesifikasi komputer yang dibutuhkan, selanjutnya Anda akan menginstal Unity ke dalam komputer. Saat penulisan buku ini, Unity terbaru adalah versi 5. Tetapi, dalam proses penulisan buku ini menggunakan Unity versi 4. Unity versi 5 mengalami perubahan dalam hal skrip pemrograman. Hal ini berarti Anda harus menyesuaikan pemrograman dari skrip Unity versi sebelumnya. Pada Unity versi 4 dan sebelumnya, Anda dapat menggunakan skrip pemrograman yang sama.

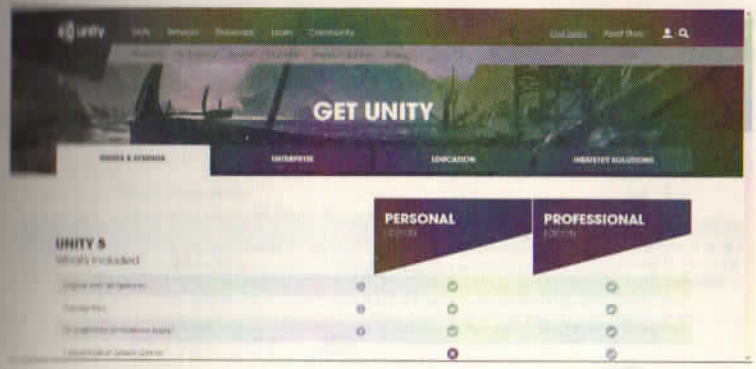
Untuk meng-intall Unity, ikuti langkah-langkah berikut:

1. Buka situs Unity di alamat **https://unity3d.com/**.
2. Anda akan melihat tulisan **Download Unity 5** sebagai shortcut untuk men-download Unity 5 seperti Gambar 1.3. Untuk men-download Unity versi 4, klik **Get Unity**.



Gambar 1.3 Tampilan halaman utama situs Unity

3. Untuk mendapatkan versi gratis, Anda pilih tab **Indies & Studio**.
4. Selanjutnya, Anda pilih kolom **Personal**.

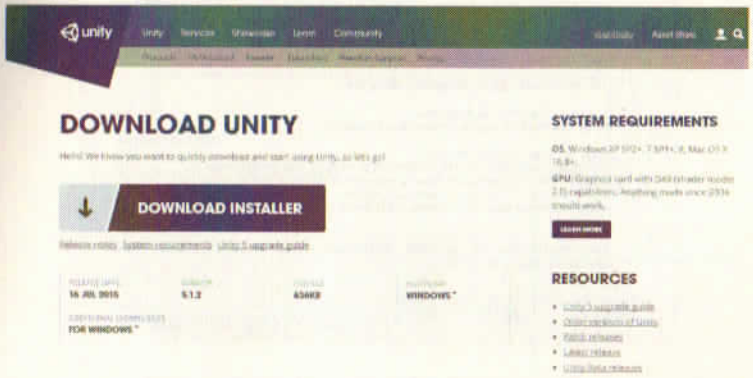


Gambar 1.4 Pilihan jenis Unity

5. Klik tombol **Free Download** di bagian bawah kolom untuk men-download unity secara gratis.

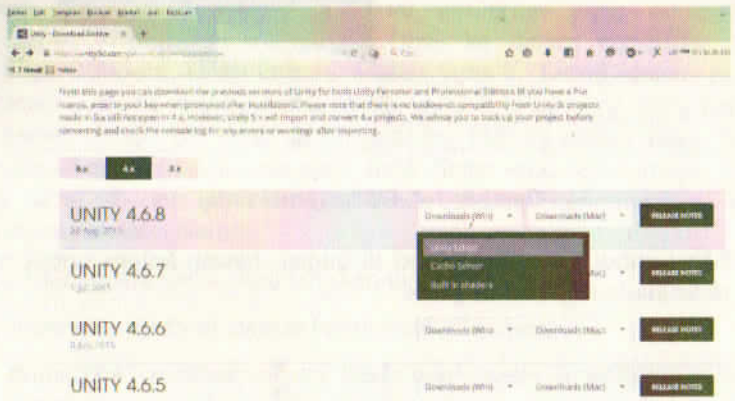
Gambar 1.5 Klik Free Download

6. Selanjutnya, Anda akan dibawa ke halaman download installer.



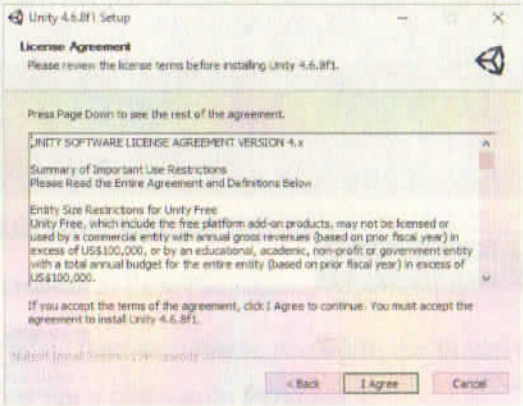
Gambar 1.6 Halaman download installer

7. Klik pada Resources > Older version of Unity.
8. Anda akan melihat pilihan versi Unity.
9. Pilih versi terbaru dari Unity versi 4 dan klik **Download (Win) > Unity Editor**.



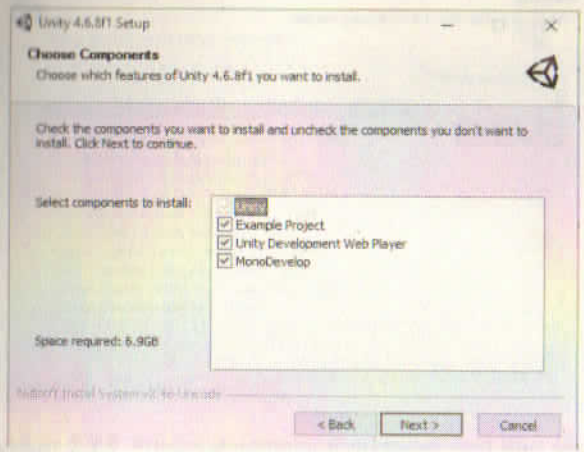
Gambar 1.7 Halaman awal proses instalasi

10. Tunggu hingga proses download selesai dan klik file installer.
11. Klik **Next** dan Anda akan dibawa ke halaman **License Agreement**.



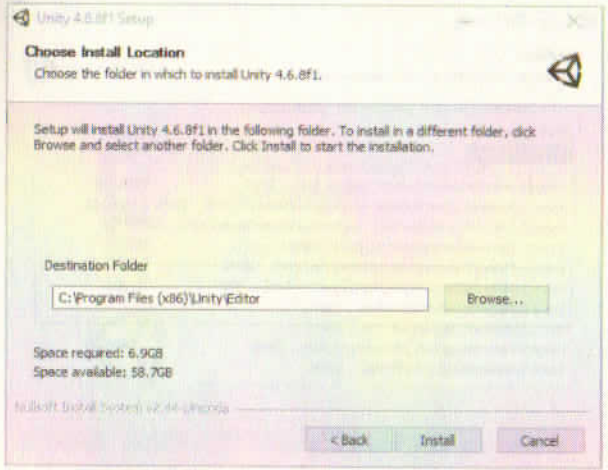
Gambar 1.8 Halaman License Agreement

12. Klik **I Agree** dan Anda akan diberikan pilihan komponen yang akan di-download. Jika Anda menginginkan contoh project Unity, Anda dapat memberi tanda pada semua pilihan.



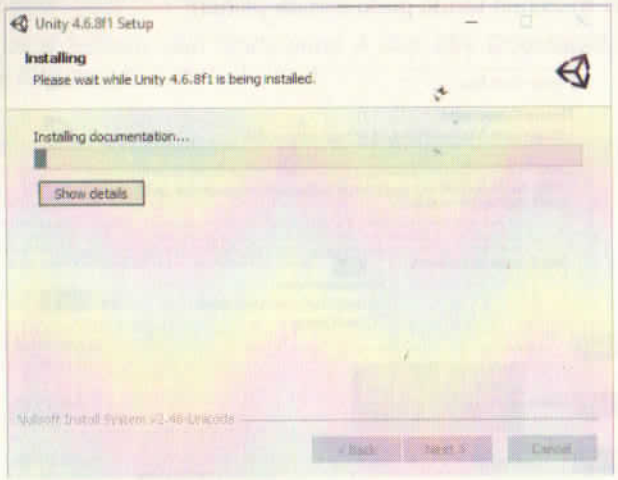
Gambar 1.9 Memilih komponen untuk Unity

13. Klik **Next** dan Anda dapat memilih lokasi instalasi Unity.



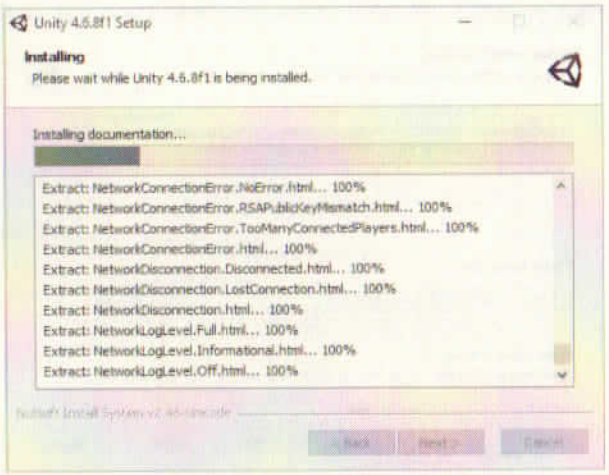
Gambar 1.10 Memilih lokasi instalasi Unity

14. Klik **Install** dan proses instalasi akan berlangsung.



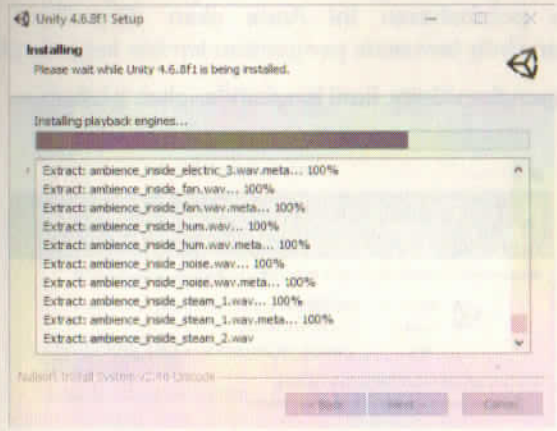
Gambar 1.11 Proses download komponen Unity

15. Tunggu proses download komponen yang diikuti proses instalasi komponen Unity.



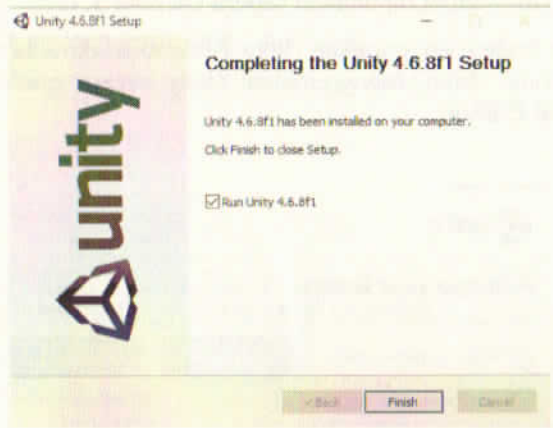
Gambar 1.12 Proses instalasi unity dimulai

16. Proses instalasi juga akan menginstal komponen tambahan dalam Unity.



Gambar 1.13 Proses download komponen lain dari Unity

17. Setelah proses intstalasi selesai, Anda akan melihat halaman seperti Gambar 1.14. Klik **Finish** untuk menyelesaikan proses instalasi. Anda dapat memberi tanda pada **Run Unity** jika ingin langsung menggunakan Unity.

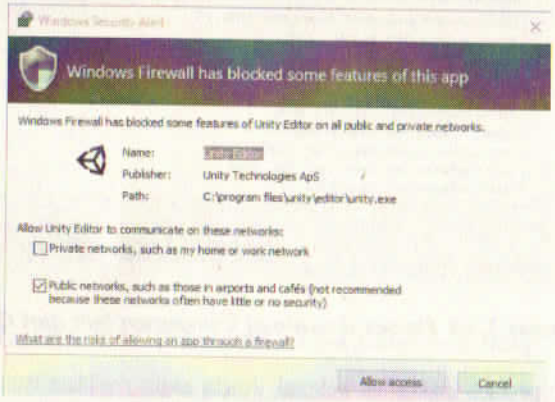


Gambar 1.14 Proses instalasi selesai

1.4 Menggunakan Unity

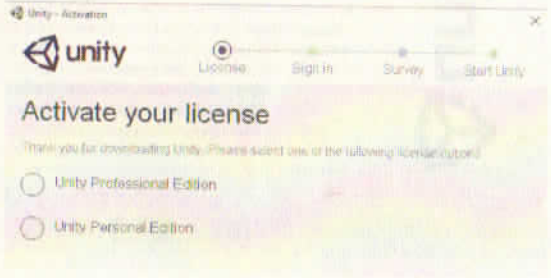
Setelah terpasang di komputer, Anda dapat langsung menggunakan Unity. Pada pembahasan ini Anda akan dikenalkan dasar-dasar menggunakan Unity termasuk pengenalan lembar kerja dari Unity.

Untuk menggunakan Unity, ikuti langkah-langkah berikut:



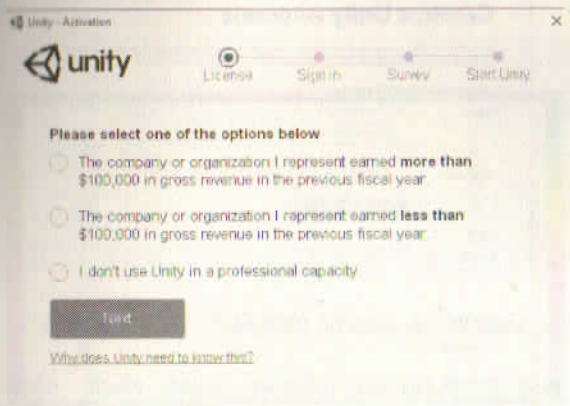
Gambar 1.15 Membuka firewall

1. Klik ikon Unity. Terkadang Anda harus membuka blok dari firewall supaya Unity dapat digunakan seperti Gambar 1.15.
2. Seblum Anda menggunakan Unity, Anda akan diminta memilih tipe akun Unity. Untuk menggunakan Unity secara gratis, klik **Unity Personal Edition**.



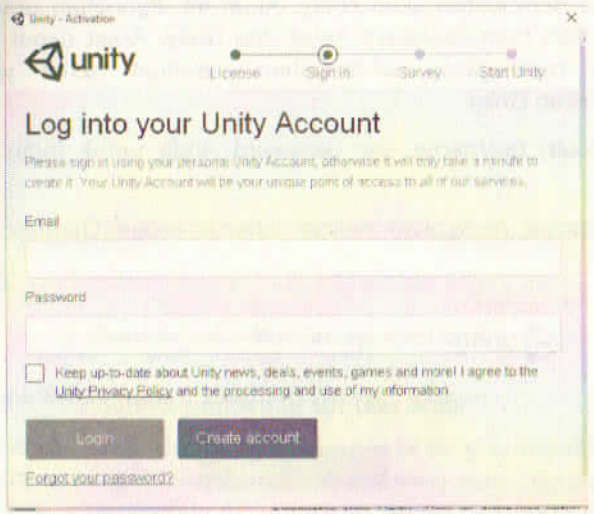
Gambar 1.16 Memilih tipe akun Unity

3. Selanjutnya, Anda akan diberikan pilihan tujuan penggunaan Unity. Klik pilihan paling bawah supaya dapat menggunakan unity secara gratis. Klik **Next**.



Gambar 1.17 Memilih tujuan penggunaan Unity

4. Selanjutnya, Anda akan diminta untuk memasukkan akun Unity.

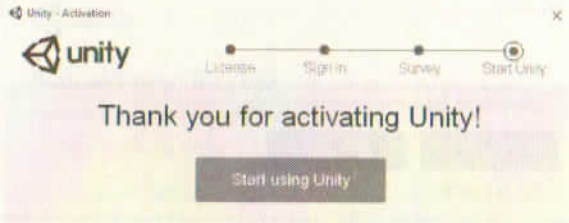


Gambar 1.18 Login akun Unity

5. Jika Anda tidak memiliki akun Unity, klik **Create account**. Kemudian Anda akan diantar ke situs pembuatan akun Unity.

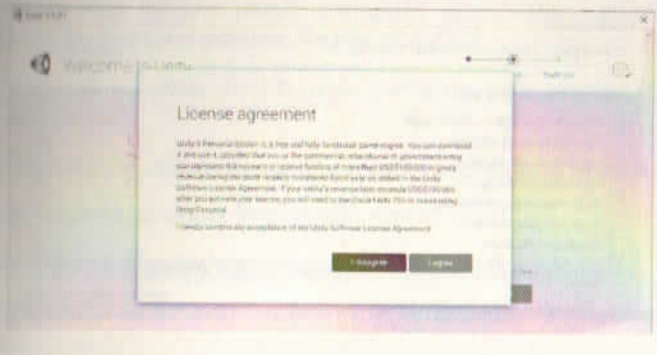
Gambar 1.19 Mengisi data

6. Setelah mengisi data, klik **Create Account** untuk menyelesaikan proses pembuatan akun Unity. Akun ini digunakan untuk aktivasi Unity dan men-download Asset dari Unity. Asset dapat digunakan untuk melengkapi bahan-bahan pembuat game yang telah disediakan Unity.
7. Masukkan username dan password Anda untuk login ke dalam Unity.
8. Selanjutnya, Anda akan melihat halaman seperti Gambar 1.20.



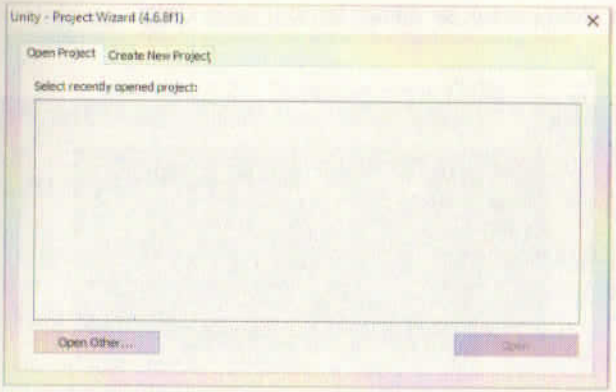
Gambar 1.20 Memilih versi Unity

9. Klik **Next** dan akan muncul **License Agreement**.



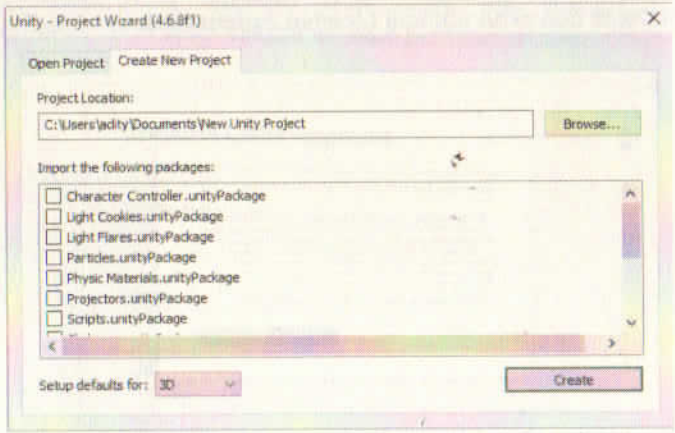
Gambar 1.21 Tampilan License Agreement

10. Selanjutnya, Anda dapat memilih membuat project baru atau menampilkan project lama.



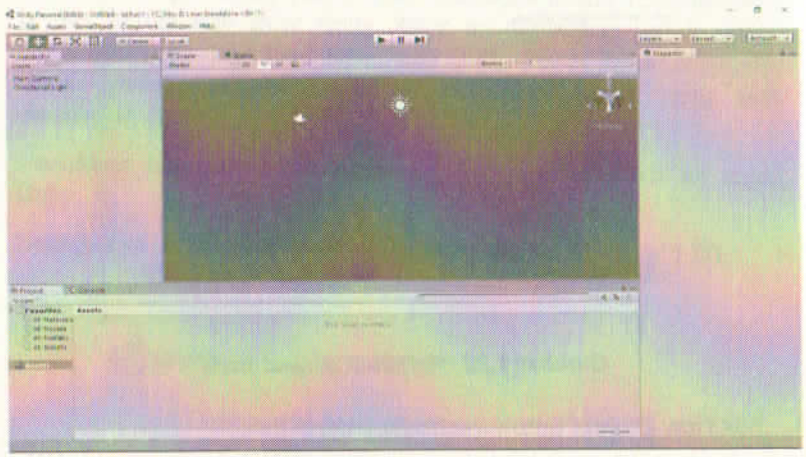
Gambar 1.22 Membuat project baru

11. Klik **New Project** untuk membuat project baru.
12. Selanjutnya, Anda dapat memberi nama projek, memilih tempat menyimpan projek, dan memilih asset yang akan digunakan. Jika Anda tidak ingin menggunakan asset, Anda tidak perlu klik bagian asset. Asset ini dapat ditambahkan saat Anda sedang membuat projek.



Gambar 1.23 Pengaturan project

13. Klik **Create Project** untuk loading data project dan asset yang Anda pilih.
14. Anda akan masuk ke dalam lembar kerja Unity.



Gambar 1.24 Tampilan lembar kerja Unity

Setelah Anda membuka Unity, Anda perlu mengenal lembar kerja dari Unity. Secara umum, lembar kerja Unity dibagi menjadi menu, shortcut, dan jendela lembar kerja.

1.4.1 Menu

Menu merupakan bagian untuk mengatur pembuatan game secara menyeluruh. Pada menu terdapat kategori berikut:

- **File**. Digunakan untuk membuat project baru, menyimpan, load, dan konversi project.
- **Edit**. Digunakan untuk mengedit project yang sedang Anda kerjakan.
- **Asset**. Digunakan untuk import dan export asset.
- **Game Object**. Digunakan untuk membuat objek pada Unity.
- **Component**. Digunakan untuk menambahkan komponen pada project.
- **Window**. Digunakan untuk mengatur tampilan jendela pada lembar kerja.
- **Help**. Digunakan untuk mengetahui manual dari Unity.

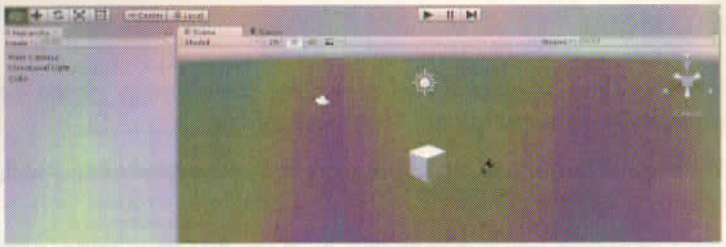
Gambar 1.25 Tampilan menu Unity

1.4.2 Shortcut

Shortcut merupakan bagian dari Unity yang digunakan untuk memudahkan Anda dalam membuat project. Shortcut merupakan jalan singkat dari beberapa pilihan menu.

Jika Anda lihat pada lembar kerja Unity, pada bagian shortcut terdiri dari tiga bagian, yaitu kiri, tengah, dan kanan. Pada bagian kiri, Anda dapat menggunakan shortcut untuk pengaturan objek pada jendela Scene. Shortcut tersebut sebagai berikut:

- **Gambar tangan**. Digunakan untuk menggeser tampilan layar project.

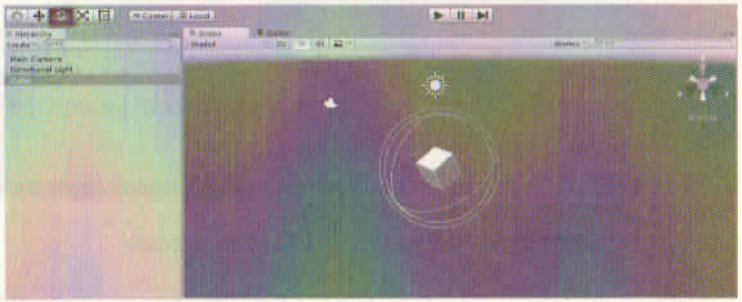


Gambar 1.26 Menggunakan shortcut gambar tangan

- **Shortcut arah**. Digunakan untuk menggeser objek berdasarkan koordinat sumbu x, y, dan z.

Gambar 1.27 Menggunakan shortcut menggeser objek

- **Shortcut rotasi**. Digunakan untuk memutar objek berdasarkan koordinat sumbu x, y, dan z.

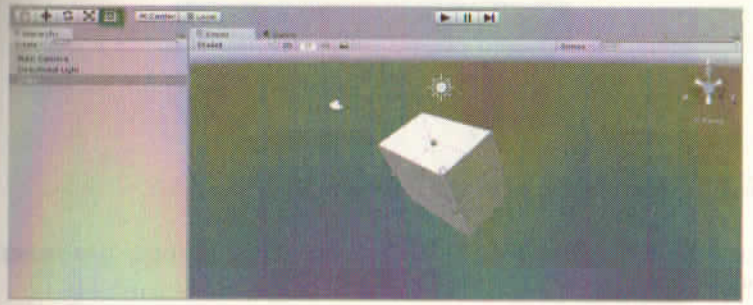


Gambar 1.28 Menggunakan shortcut memutar objek

- **Shortcut skala**. Digunakan untuk mengubah ukuran skala objek berdasarkan koordinat x, y, dan z.

Gambar 1.29 Menggunakan shortcut skala

- **Shortcut skala horizontal**. Digunakan untuk mengubah ukuran objek sisi horizontal.

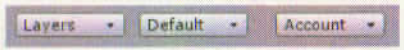


Gambar 1.30 Menggunakan shortcut scala horizontal

Pada bagian tengah terdapat tombol untuk menjalankan dan menghentikan game. Tombol ini digunakan untuk menguji berjalannya game pada jendela Game sebelum game dikompilasi menjadi *stand alone*.

Gambar 1.31 Menu shortcut menjalankan game

Pada bagian kanan terdapat Layer, Layout, dan Account. Layer digunakan untuk mengatur tampilan layer pada jendela Scene. Layout digunakan untuk mengatur susunan layout jendela kerja. Account digunakan untuk login dan logout akun Unity.

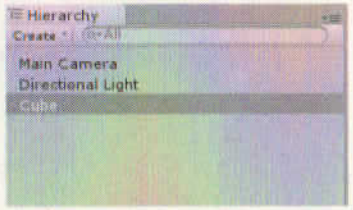


Gambar 1.32 Shortcut pengaturan Layer, Layout, dan Account

1.4.3 Jendela Lembar Kerja

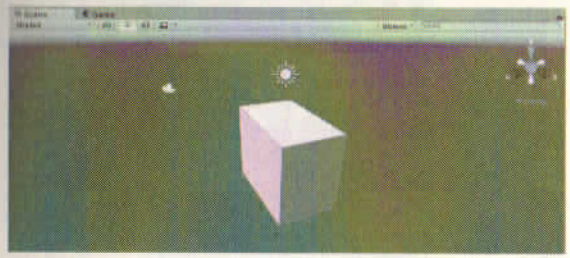
Secara default, jendela lembar kerja dibagi menjadi empat bagian seperti berikut:

- **Hierarchy**: Jendela yang menampilkan daftar objek yang ada di jendela Scene.



Gambar 1.33 Jendela Hierarchy

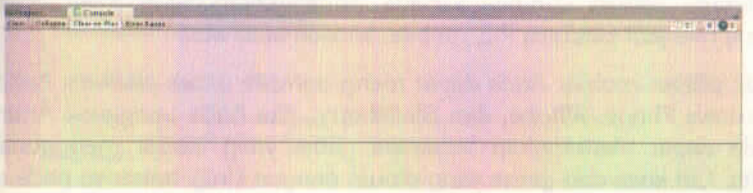
- **Jendela Scene/Game**: Jendela Scene menampilkan seluruh onjek yang akan digunakan dalam game. Jendela Game digunakan untuk menjalankan objek yang telah diletakkan pada jendela Scene.



Gambar 1.34 Jendela Scene/Game

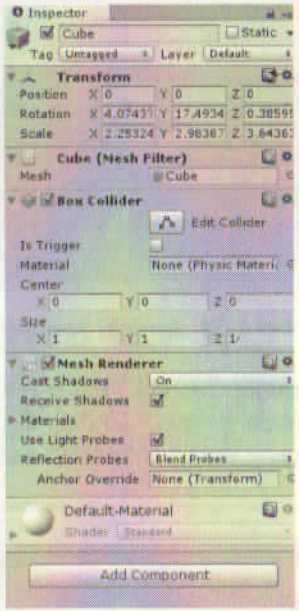
- **Jendela Project/Console**: Jendela Project menampilkan semua asset yang Anda import pada project. Jendela Console digunakan untuk mengetahui bagian yang error/bermasalah saat game akan dijalankan (*Run*).

Gambar 1.35 Jendela Project



Gambar 1.36 Jendela Console

- **Jendela Inspector**: Digunakan untuk mengatur asset yang dimasukkan ke dalam layer Scene.



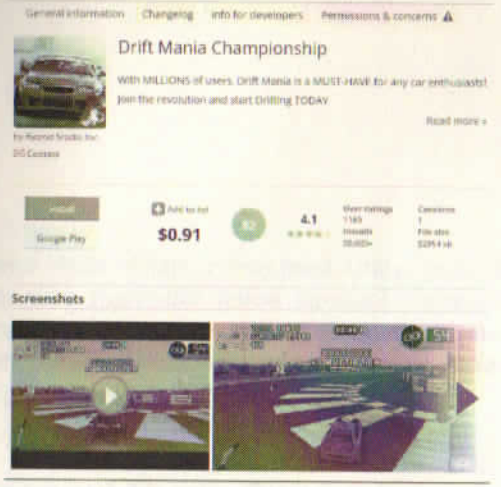
Gambar 1.37 Jendela Inspect

1.5 Unity dengan Game Android

Dalam proses pembuatan game dengan game engine, pada tahap akhir tentu Anda akan meng-*compile* game tersebut supaya dapat dimainkan orang tanpa harus membuka Unity terlebih dahulu. Unity memberikan beberapa pilihan compile kepada Anda. Anda dapat meng-compile game menjadi platform PC, mobile, konsol, atau web.

Pada pilihan mobile, Anda dapat meng-compile untuk platform Android, Windows Phone, iPhone, dan Blackberry. Jika Anda pengguna Android, Anda dapat menemukan beberapa game yang dibuat menggunakan Unity. Ciri khas dari game yang dibuat dengan Unity baisanya pada awal game ditampilkan logo Unity, meskipun logo ini dapat dihilangkan jika Anda menggunakan Unity versi berbayar.

Secara umum, proses compile game menjadi mobile dalam platform Android mirip dengan proses compile untuk platform lainnya. Anda perlu memasukkan perintah yang menggunakan touchscreen untuk menggerakkan karakter. Selain itu, pada proses compile Anda akan diminta untuk menginstal software development kit dari perangkan tersebut. Pada Android, Anda akan diminta menginstal SDK.



Gambar 1.38 Game Drift Mania Championship

Gambar 1.39 Game Ragna Rock

2 RANCANGAN AWAL GAME SURVIVAL

Untuk membantu Anda dalam menguasai Unity, pada buku ini Anda akan dijelaskan cara membuat game *survival*. Sebelum Anda mulai membuat game, Anda akan membuat rancangan awal dari game tersebut. Tujuan pembuatan rancangan awal supaya Anda mendapatkan gambaran dari game yang akan Anda buat.

2.1 Mengenal Game Survival

Survival secara sederhana dapat diartikan bertahan hidup. Pada game survival, Anda akan mengendalikan karakter untuk bertahan hidup menghadapi berbagai rintangan hingga tujuan yang ditetapkan oleh game tercapai.

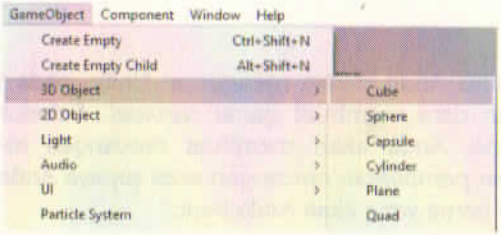
Pada pembuatan game survival kali ini Anda akan menempatkan karakter di dalam sebuah peta. Karakter diharuskan mengalahkan berbagai musuh untuk mencapai tujuan yang ditetapkan oleh game. Selain itu, dalam game, sudut pandang yang digunakan adalah sudut pandang orang pertama. Jadi Anda tidak akan melihat bentuk karakter, tetapi hanya tampilan pemandangan game dan senjata yang digunakan untuk mengalahkan musuh.

Pada bab ini, Anda akan membuat rancangan dari game survival terlebih dahulu. Anda akan dijelaskan cara membuat peta yang akan dijadikan tempat karakter bertualang serta cara karakter dapat mengalahkan musuh yang menghadang.

2.2 Membuat Peta dan Menguji Pergerakan Karakter

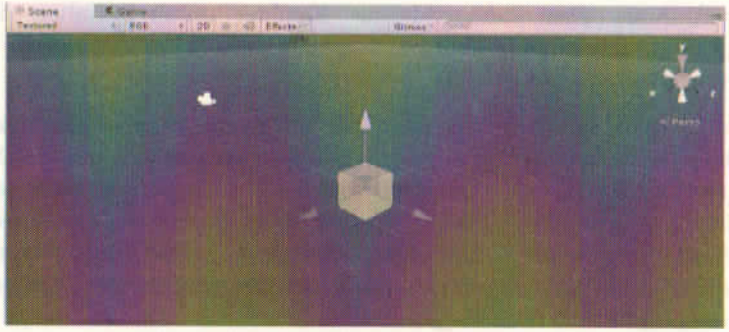
Pada awal perancangan game, Anda akan mencoba membuat peta sederhana sebagai tempat bergeraknya karakter. Selanjutnya, Anda juga akan menambahkan karater dari asset bawaan Unity untuk mencoba pergerakan karakter di dalam peta. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Buka program Unity.
2. Klik GameObject > 3D Object > Cube.



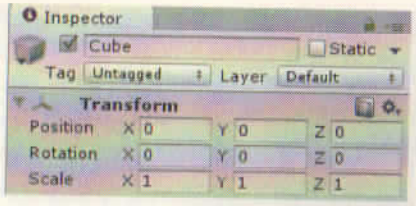
Gambar 2.1 Klik GameObject > 3D Object > Cube

3. Anda akan melihat betuk kubus di jendela **Scene**.



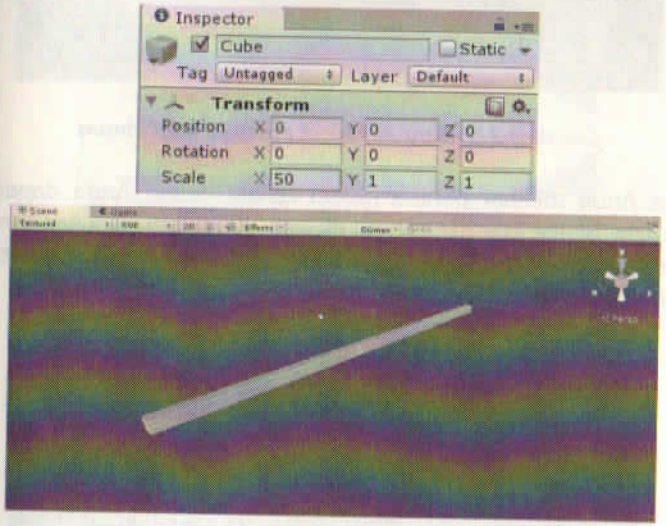
Gambar 2.2 Tampilan kubus di jendela Scene

4. Pastikan posisi kubus berada di tengah layar dengan cara ubah nilai **x**, **y**, dan **z** menjadi "0" pada jendela **Inspection** bagian **Position**.



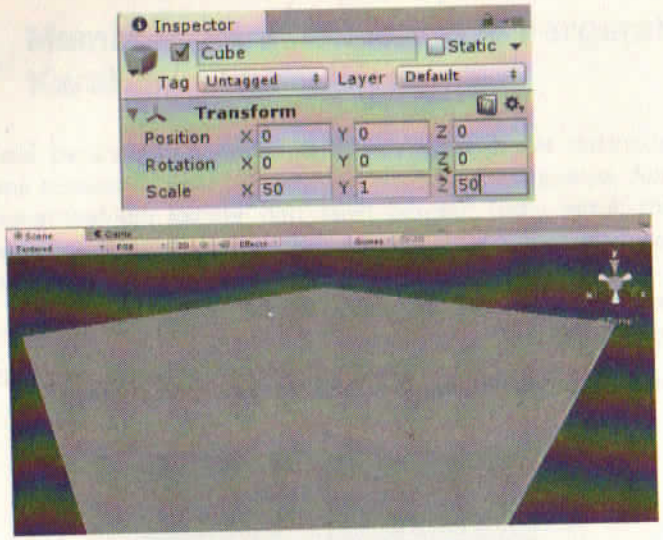
Gambar 2.3 Mengatur posisi x, y, dan z menjadi 0

5. Masih pada jendela **Inspector**. Pada bagian **Scale** ubah nilai **x** menjadi "50".
6. Anda akan melihat bentuk kubus menjadi lebih panjang.



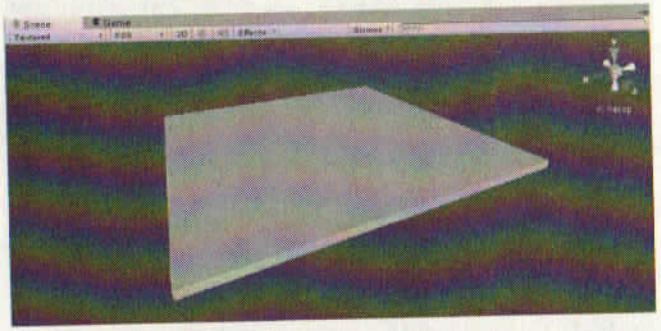
Gambar 2.4 Mengubah nilai x menjadi 50

7. Selanjutnya, ubah nilai **z** menjadi "50".
8. Anda akan melihat bentuk object menjadi kotak persegi empat sama sisi.



Gambar 2.5 Bentuk object menjadi segi empat

9. Jika Anda melihat bentuk object terlalu besar, Anda dapat zoom jendela Scene dengan scroll mouse. Gunakan tool tangan untuk menggeser sudut pandang dan klik kanan untuk menggeser pandangan secara vertikal.



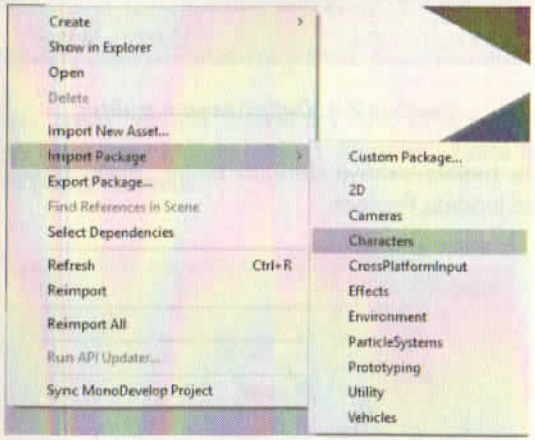
Gambar 2.6 Mengubah sudut pandang tampilan peta

10. Pada jendela **Hierarchy** ubah nama object "Cube" menjadi "GroundPlane". Caranya, klik kanan nama object **Cube** dan pilih **Rename**. Pilih nama baru menjadi "GroundPlane".

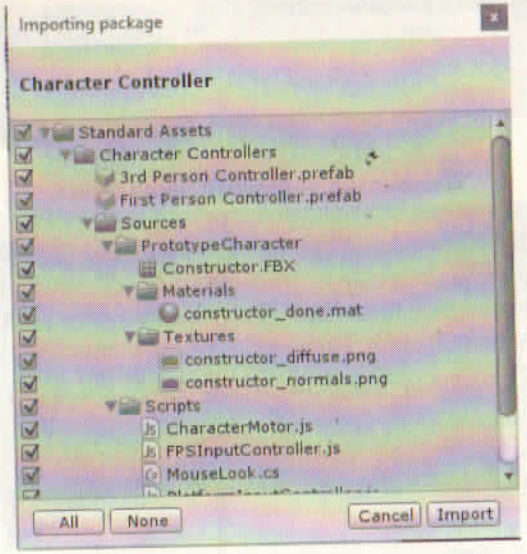
Gambar 2.7 Mengubah nama object

11. Selanjutnya, Anda akan menambahkan asset karakter ke dalam Unity. Asset ini digunakan untuk menguji pergerakan karakter pada peta. Untuk melakukannya, Klik kanan pada jendela **Project**. Kemudian pilih **Import Package > Character** seperti Gambar 2.8.

12. Anda akan melihat daftar asset dari karakter seperti Gambar 2.9. Klik **Import**.



Gambar 2.8 Klik Import Package > Character

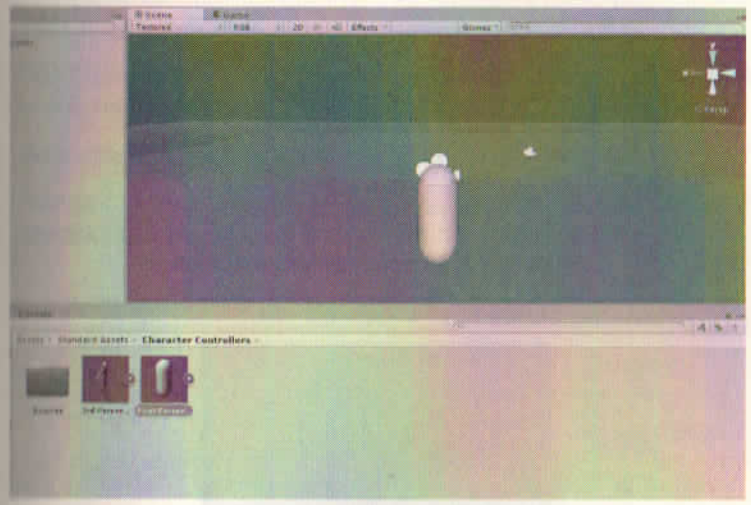


Gambar 2.9 Daftar asset karakter

13. Jika Anda melakukannya dengan benar, Anda akan melihat asset baru pada jendela **Project**.

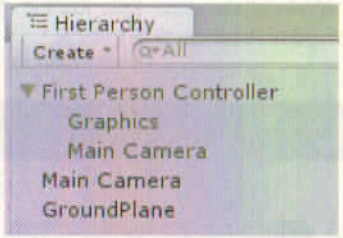
Gambar 2.10 Menambahkan asset karakter

14. Selanjutnya, masukkan karakter ke dalam jendela Scene. Caranya pada asset buka folder **Asset > Standart Assets > Charaters > First Person Controller > Prefabs**. Klik objek dan drag ke jendela **Scene**.



Gambar 2.11 Menambahkan karakter pada jendela Scene

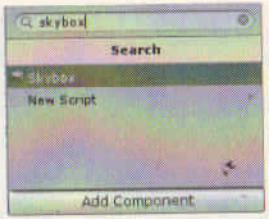
15. Selanjutnya, pada jendela **Hierarchy** klik object **First Character Controller** yang merupakan object karakter. Anda akan melihat komponen **Main Camera**.



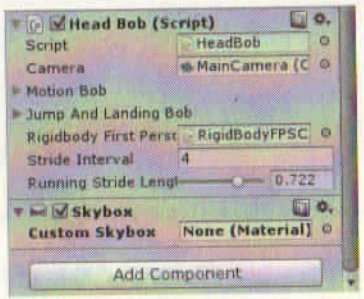
Gambar 2.12 Klik object untuk menampilkan komponen Camera

16. Pada jendela **Inspector** dari object **Main Camera**, Klik **Add Component** dan ketikkan **Skybox**.

17. Anda akan melihat komponen Skybox. Klik komponen sehingga komponen ditambahkan dalam jendela **Inspector**.

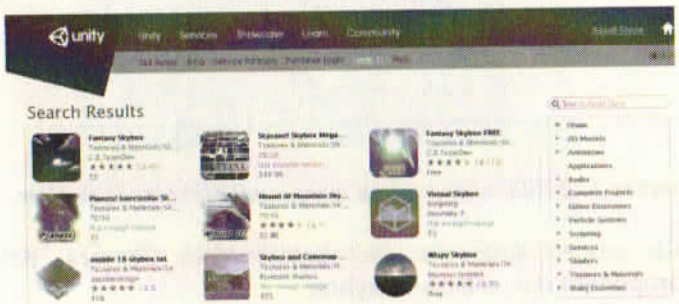


Gambar 2.13 Mencari komponen Skybox



Gambar 2.14 Komponen Skybox berhasil ditambahkan

18. Masukkan asset skybox dengan klik kanan pada jendela **Project**. Kemudian pilih **Import Assets > Skybox**.



Gambar 2.15 Pencarian asset skybox

19. Jika Anda tidak punya asset skybox, Anda dapat men-downloadnya di assetstore Unity di https://www.assetstore.unity3d.com/en/. Ketik "Skybox" pada kolom pencarian seperti Gambar 2.15.

20. Pilih asset yang "free" untuk mendapatkan asset gratis.
21. Klik **Open in Unity** untuk membuka aset di Unity. Anda akan dipilih untuk membuka asset di **Unity Editor**.
22. Saat asset ditampilkan di Unity Editor, klik **Download**.
23. Tunggu proses download. Anda akan melihat asset siap untuk dimasukkan ke dalam Unity dengan cara klik **Import**.

Gambar 2.16 Unity Editor

24. Kembali pada jendela **Inspector** dari **Camera**. Pada bagian**Skybox** klik pada tanda bulat untuk menampilkan pilihan skybox.

Gambar 2.17 Memilih skybox

25. Jika Anda lihat pada jendela **Game**, Anda akan melihat tampilan langit berubah.

Gambar 2.18 Mengubah tampilan langit

26. Jika jendela Scene terlihat gelap, tambahkan cahaya dengan klik **GameObject > Light > Directional Light**.
27. Jalankan game dengan buka jendela **Game** dan klik tombol **Play**.
28. Gerakkan karakter dengan tombol arah, loncat dengan **Space**, dan berputar dengan mouse.

Gambar 2.19 Menjalankan karakter

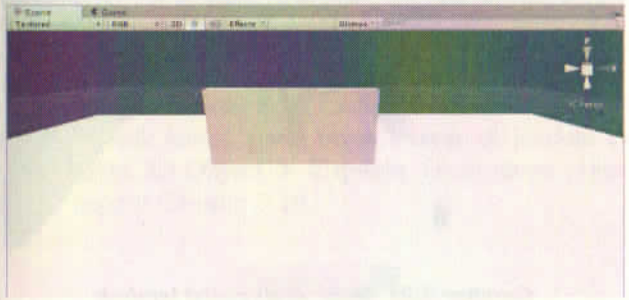
29. Jika karakter bergerak melebihi peta, karakter akan jatuh.

Gambar 2.20 Karakter jatuh dari peta

30. Untuk menghentikan game klik kembali tombol **Play**.
31. Selanjutnya, Anda akan membuat tembok. Caranya, klik **GameObject > 3D Object > Cube**.
32. Anda akan melihat kubus baru di layar **Scene**.

Gambar 2.21 Menambahkan objek kubus

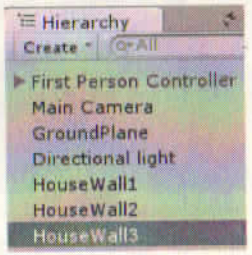
33. Buat objek kubus menjadi pipih dengan mengganti nilai **x** dengan "10" dan **y** dengan "5" pada jendela **Inspector** bagian **Scale**.



Gambar 2.22 Mengubah kubus menjadi pipih

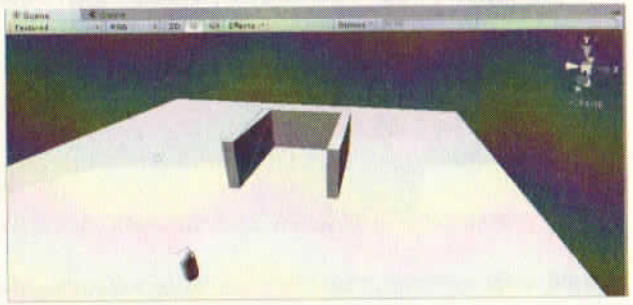
34. Pastikan objek dinding menyentuh tanah. Jika objek dinding masih terlihat melayang, gunakan tool shortcut arah atau mengubah nilai y pada jendela **Inspector** bagian **Position**.
35. Selanjutnya, ubah nama objek tembok menjadi "HouseWall1".

36. Perbanyak objek tembok dua kali dan ubah namanya menjadi "HouseWall2" dan "HouseWall3". Caranya klik kanan objek **HouseWall1** dan pilih **Duplicate**.



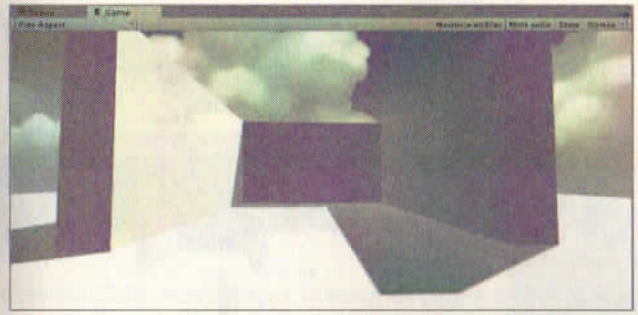
Gambar 2.23 Objek tembok diperbanyak

37. Ketiga objek tembok saat ini saling bertumpukan menjadi satu. Oleh karena itu, Anda perlu menggeser dua objek tembok menggunakan tool shortcut arah dan rotasi untuk membentuk posisi seperti Gambar 2.24.



Gambar 2.24 Mengubah posisi tembok

38. Jalankan kembali karakter untuk melihat hasil posisi tembok dari sudut pandang karakter.

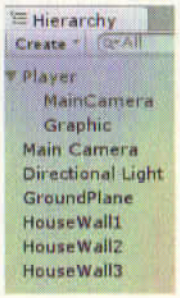


Gambar 2.25 Melihat bentuk tembok dari sudut pandang karakter

2.3 Menambahkan Musuh dan Sistem Membunuh Musuh

Dalam sebuah game survival, untuk meningkatkan tantangan Anda dapat menambahkan musuh. Selain menambahkan musuh, Anda juga tentu akan memikirkan cara membunuh musuh tersebut. Pada pembahasan ini, Anda akan membuat musuh dan membunuh musuh tersebut. Untuk lebih jelas, ikuti langkah-langkah berikut:

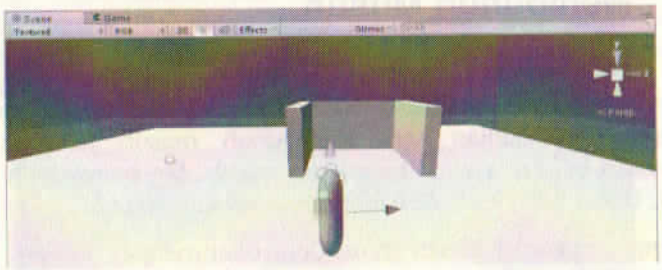
1. Ganti nama objek **First Person Controller** menjadi "Player".
2. Jika objek Player Anda tidak memiliki tampilan, buat tampilan dengan cara klik kanan pada objek **Player** di jendela **Hierarchy**. Kemudian pilih **3D Object > Capsule**. Ubah nama objek menjadi "Graphic" seperti Gambar 2.26.



Gambar 2.26 Menambahkan bentuk pada objek karakter

3. Buat objek baru dengan klik **GameObject** > **Create Empty**.
4. Anda akan melihat objek baru yang tidak memiliki bentuk.

Gambar 2.27 Menambahkan objek kosong

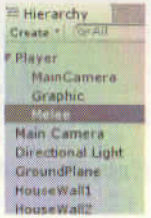


Gambar 2.28 Tampilan bentuk objek karakter

5. Beri nama objek tersebut dengan "Melee". Objek ini digunakan sebagai pengaturan supaya karakter dapat menyerang musuh.
6. Pindahkan objek sehingga ada di depan objek karakter. Untuk memudahkan, Anda dapat menyalin posisi objek Player ke posisi objek Melee yang ada di jendela **Inspector** bagian **Position** dan geser posisi ke depan objek karakter.

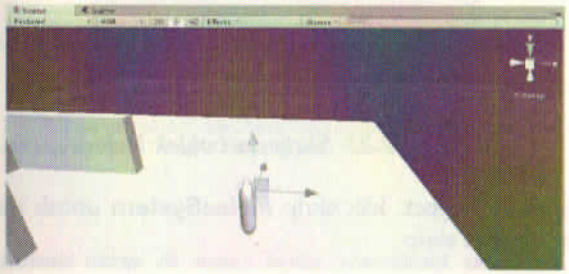
Gambar 2.29 Posisi objek kosong di dalam objek karakter

7. Pada jendela **Hierarchy**, drag objek **Melee** ke dalam objek **Player** sehingga objek Melee menjadi *child* dari objek Player. Hal ini akan membuat objek Melee bergerak mengikuti pergerakan objek Player.



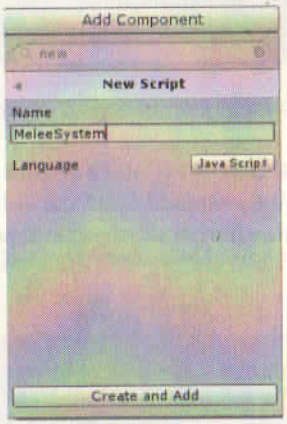
Gambar 2.30 Membuat objek Melee child dari objek Player

8. Pastikan pada jendela **inspector** bagian **Rotation** dari objek **Player** sumbu x, y, dan z adalah 0.
9. Geser posisi objek **Melee** agak ke kanan dari objek **Player** dan buat **Rotaion** objek menjadi 0 pada sumbu x, y, dan z.



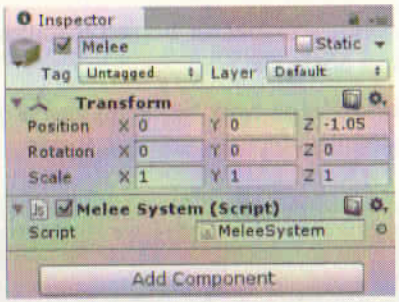
Gambar 2.31 Menggeser posisi objek Melee

10. Pada jendela **inspector** dari objek **Melee**, klik **Add Component** dan ketik "Script" kemudian pilih **New Script**.
11. Isi nama skrip dengan "MeleeSystem" dan pilih bahasa pemrograman dengan **Javascript**.



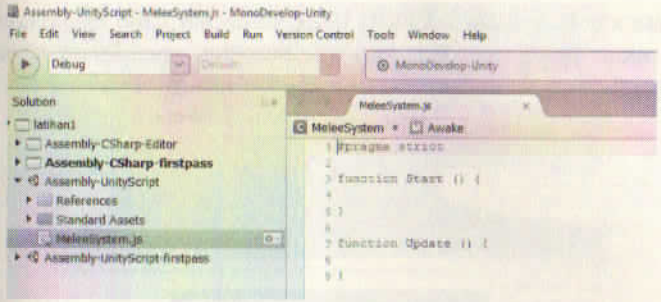
Gambar 2.32 Membuat skrip program

12. Anda akan melihat komponen baru dari objek Melee.



Gambar 2.33 Skrip dari objek Melee

13. Pada jendela **Project**, klik skrip **MeleeSystem** untuk menampilkan aplikasi menulis skrip.



Gambar 2.34 Tampilan skrip objek Melee

14. Hapus baris 3 sampai 9 dan buat variabel baru seperti berikut:

```
#pragma strict

var Damage: int = 50;
var Distance : float;
```

Berdasarkan skrip di atas, Anda menambahkan variabel Damage dan Distance. Variabel Damage digunakan untuk membuat kerusakan kepada musuh. Pada skrip di atas senjata karakter akan memberikan kerusakan sebesar 50 untuk setiap serangan. Pada variabel Distance digunakan untuk menentukan jarak serangan. Float digunakan supaya jarak dapat ditentukan dalam desimal.

15. Selanjutnya, Anda akan membuat serangan karakter dapat mengenai musuh dengan mengetikkan skrip berikut:

```
function Update ()
{
        if (Input.GetButtonDown("Fire1"))
        {
                var hit : RaycastHit;
                if (Physics.Raycast (transform.position,
transform.TransformDirection(Vector3.forward), hit))
                {
                        Distance = hit.distance;

        hit.transform.SendMessage("ApplyDammage", Damage,
SendMessageOptions.DontRequireReceiver);
                }
        }
}
```

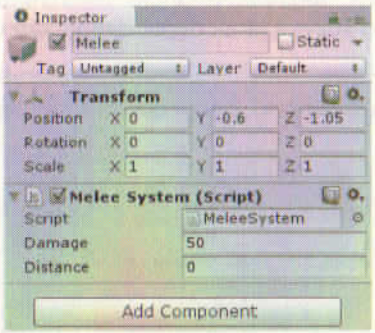
Berdasarkan skrip di atas, Anda membuat perintah menembak dengan tombol mouse (Fire1). Selain itu, Anda juga membuat serangan dilakukan ke arah depan atau arah karakter. Variabel

Distance akan menunjukkan jarak serangan yang dilakukan. Jarak ini akan berkurang jika karakter semakin mendekati objek yang menghalangi serangan.

```
#pragma strict

var Damage : int = 50;
var Distance : float;

function Update ()
{
    if (Input.GetButtonDown("Fire1"))
    {
        var hit : RaycastHit;
        if (Physics.Raycast (transform.position, transform.TransformDirection(Vector3.forward), hit))
        {
            Distance = hit.distance;
            hit.transform.SendMessage("ApplyDamage", Damage, SendMessageOptions.DontRequireReceiver);
        }
    }
}
```

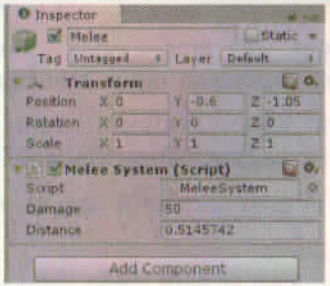
Gambar 2.35 Skrip lengkap dari MeleeSystem

16. Jika Anda lihat pada jendela **Inspector** dari objek **Melee**, Anda akan melihat komponen baru yang berisi Damage dan Distance. Komponen Damage akan bernilai tetap, sedangkan komponen Distance akan berubah tergantung jarak karakter dengan objek penghalang.



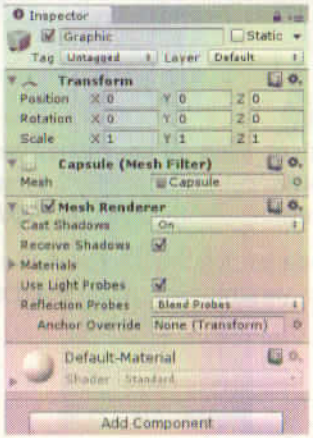
Gambar 2.36 Tampilan komponen Damage dan Distance

17. Jika Anda jalankan dan Anda klik kiri, nilai Distance akan berubah sesuai jarak karakter dengan objek penghalang.



Gambar 2.37 Perubahan nilai Distance

18. Jika karakter Anda mendadak tidak dapat digerakkan buka bagian jendela **Inspector** objek **Graphic**. Kemudian klik tanda gear pada **Capsule Collider** dan pilih **Remove Component**.



Gambar 2.38 Komponen objek Graphic

19. Saat ini Anda telah membuat karakter yang dapat menyerang musuh, akan tetapi serangan tersebut tidak memiliki batasan. Oleh karena itu, Anda akan membuat batas maksimal jarak serangan. Hal ini bertujuan supaya karakter dapat membunuh musuh dari jarak tertentu saja. Caranya, tambahkan variabel baru pada **MeleeSystem**.

```
var Damage : int = 50;
var Distance : float;
var MaxDistance : float = 1.5;
```

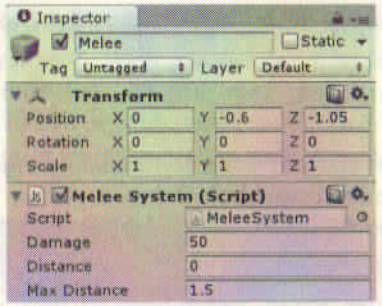
Berdasarkan skrip di atas, Anda menambahkan variabel MaxDistance yang digunakan untuk menentukan masksimal jarak serangan, yaitu 1,5.

20. Selanjutnya, pada bagian fungsi **Update** buat menjadi seperti berikut.

```
function Update ()
{
        if (Input.GetButtonDown("Fire1"))
        {
                var hit : RaycastHit;
                if (Physics.Raycast (transform.position,
transform.TransformDirection(Vector3.forward), hit))
                {
                        Distance = hit.distance;
                        if(Distance < MaxDistance);
                        {

        hit.transform.SendMessage("ApplyDammage", Damage,
SendMessageOptions.DontRequireReceiver);
                        }
                }
        }
}
```

Berdasarkan skrip di atas, Anda menambahkan rumus Distance kurang dari MaxDistance. Hal ini berarti serangan akan berhasil jika variabel Distance nilainya lebih kecil dari MaxDistance.

21. Jika Anda lihat pada jendela **Inspector** dari objek **Melee**, Anda akan melihat komponen variabel baru, yaitu **MaxDistance**.

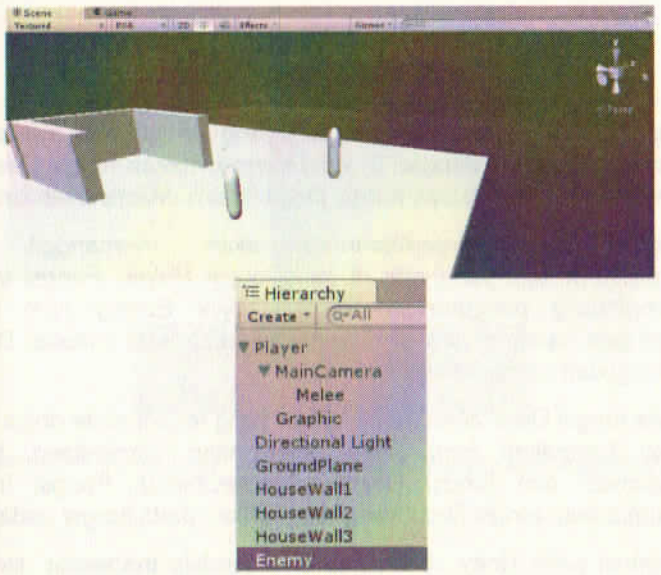


Gambar 2.39 Variabel baru pada objek Melee

22. Sekarang Anda akan membuat objek musuh. Caranya klik **CreateObject > 3D Object > Capsule**.

23. Tempatkan objek musuh di tempat yang Anda inginkan dan beri nama "Enemy".

24. Pada jendela **Inspector** objek **Enemy** tambahkan komponen skrip dengan nama "EnemyLogic".



Gambar 2.40 Tampilan objek musuh

25. Selanjutnya, hapus skrip default dan tambahkan variabel baru seperti berikut:

```
var Health = 100;
```

Variabel di atas sebagai nyawa dari musuh dengan jumlah 100. Jika Anda lihat kembali variabel Damage, Anda telah menentukan variabel dengan nilai 50. Artinya, untuk membunuh musuh perlu dua kali serangan.

26. Selanjutnya, buat fungsi untuk menentukan persyaratan musuh mati.

```
function Update ()
{
        if(Health <= 0)
        {
                Dead();
        }
}

function ApplyDammage (Damage : int)
{
        Health -= Damage;
}

function Dead ()
{
        Destroy (gameObject);
}
```

Pada fungsi Update menjelaskan yang terjadi jika nyawa objek kurang dari sama dengan 0, yaitu memunculkan fungsi Dead yang berarti mati. Akan tetapi, fungsi Dead belum dikenali oleh Unity.

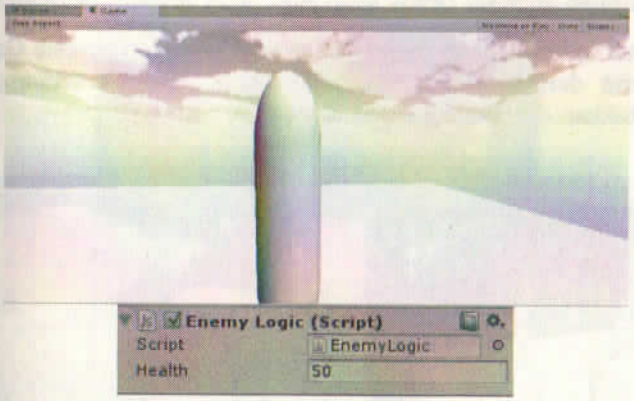
Pada fungsi ApplyDammag akan memanggil fungsi ApplyDammage yang ada di skrip objek Player. Fungsi ini akan menghitung pengurangan nyawa objek Enemy saat terkena serangan karakter dari jarak tertentu, yaitu saat variabel Distance kurang dari variabel MaxDistance.

Pada fungsi Dead akan menjelaskan yang terjadi pada objek Enemy saat dinyatakan mati, yaitu objek akan menghilang. Hal ini dijelaskan dari fungsi Destroy (gameObject). Fungsi ini juga menjelaskan fungsi Dead yang ditampilkan pada fungsi update.

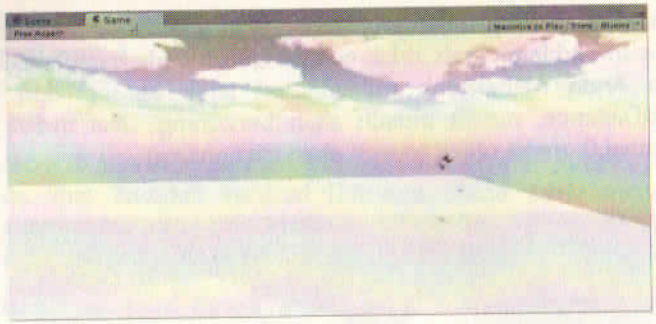
27. Kembali pada Unity. Jika Anda lihat jendela **Inspector** dari objek **Enemy**, Anda akan melihat sekarang objek memiliki nyawa. Komponen nyawa akan berkurang saat terkena serangan dari karakter.

28. Klik tombol **Play** untuk menguji game. Jika Anda menembakkan musuh dari jarak jauh, tidak ada pengaruh pada musuh. Sebaliknya jika Anda menembak musuh dari jarak kurang dari variabel MaxDistance, nyawa musuh akan berkurang. Jika nyawa musuh sampai 0, maka objek musuh akan menghilang.

Gambar 2.41 Pada jarak jauh, serangan tidak mepengaruhi nyawa musuh



Gambar 2.42 Pada jarak dekat, serangan mengurangi nyawa musuh

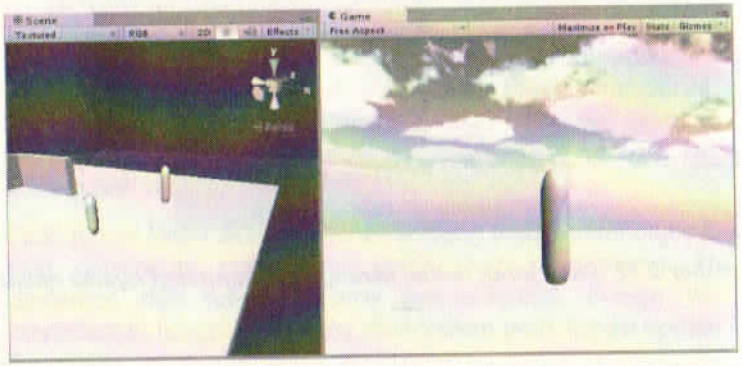


Gambar 2.43 Saat nyawa musuh 0, objek musuh menghilang

2.4 Membuat Animasi Senjata

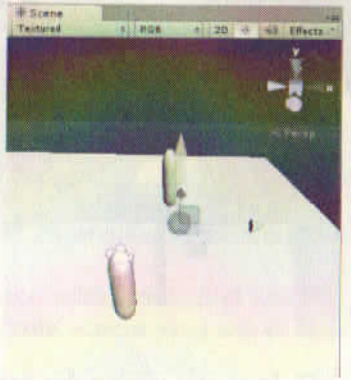
Untuk membuat serangan karakter menjadi terlihat, Anda akan menambahkan senjata sebagai alat untuk membunuh musuh. Senjata ini akan bergerak dan akan mengurangi nyawa musuh jika senjata mengenai musuh. Untuk menggerakkan musuh, Anda akan membutuhkan animasi. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Untuk memudahkan membuat senjata pisahkan jendela **Game** dan **Scene** dengan cara drag pada nama "Game" sehingga dapat diletakkan di sebelah jendela **Scene**.



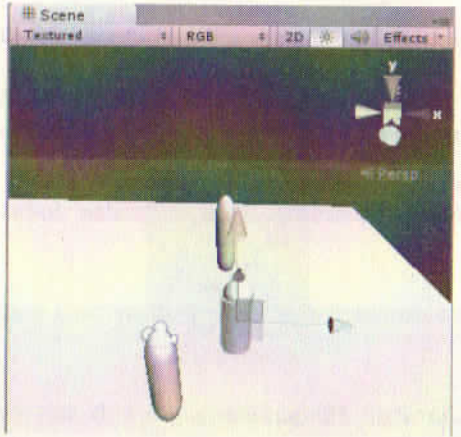
Gambar 2.44 Memishakan jendela Game dan Scene

2. Buat objek baru dengan klik **GameObject > 3D Object > Sphere**. Kemudian pindahkan objek mendekati objek **Player**.



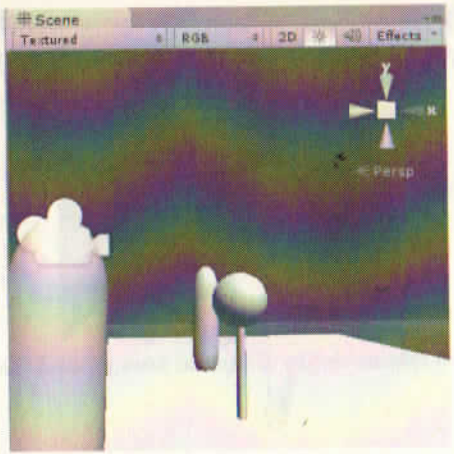
Gambar 2.45 Meletakkan objek baru di dekat objek Player

3. Buat lagi objek baru dengan pilih **GameObject > 3D Object > Cylinder**. Letakkan objek di bawah objek bola.



Gambar 2.46 Meletakkan objek silinder di bawah objek bola

4. Atur skala dan posisi dari objek bola dan silinder sehingga proporsional dengan objek Player.

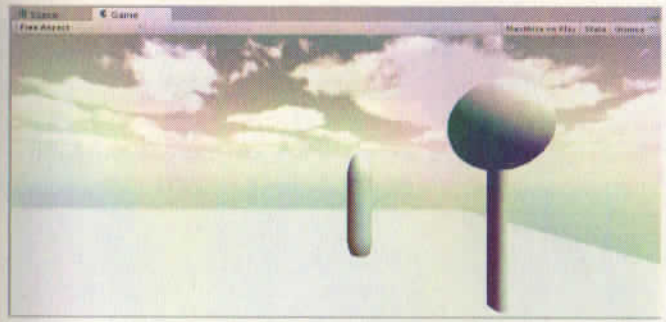


Gambar 2.47 Mengatur ukuran dan posisi objek

5. Pada jendela **Hierarchy**, drag objek silinder sehingga menjadi child dari objek bola.
6. Ubah nama objek bola menjadi "Mace" dan objek silinder menjadi "Grip".

Gambar 2.48 Menyatukan objek bola dan silinder

7. Letakkan objek **Mace** di depan karakter agak ke kanan sehingga memiliki tampilan di jendela **Game** seperti Gambar 2.49.

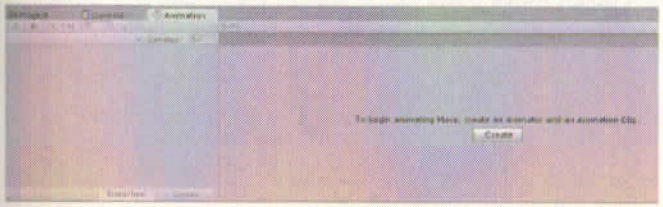


Gambar 2.49 Meletakkan objek Mace dekat karakter

8. Supaya objek Mace selalu mengikuti karakter, jadikan objek **Mace** child dari objek **MainCamera** yang ada di dalam objek **Player**.

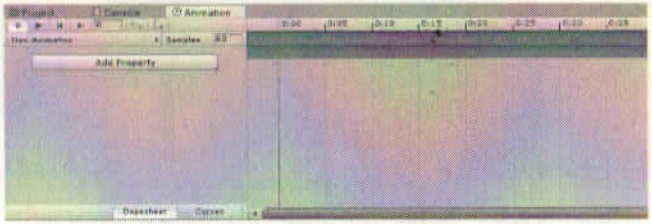
Gambar 2.50 Membuat objek Mace child dari objek MainCamera

9. Hilangkan **Sphere Collider** dari jendela **Inspector** objek **Mace** dan **Cylinder Collider** dari jendela **Inspector** objek **Grip**.
10. Klik objek **Mace**. Kemudian klik **Window** > **Animation**. Drag jendela tersebut sehingga ada di sebelah jendela **Project** dan **Console**.



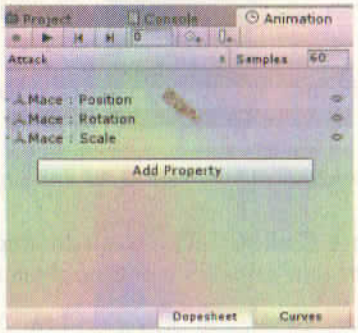
Gambar 2.51 Menampilkan jendela Animation

11. Klik **Create** pada jendela **Animation** sehingga dimunculkan kotak dialog untuk menyimpan animasi. Beri nama animasi dengan "Attack" dan klik **Save** untuk menutup kotak dialog.



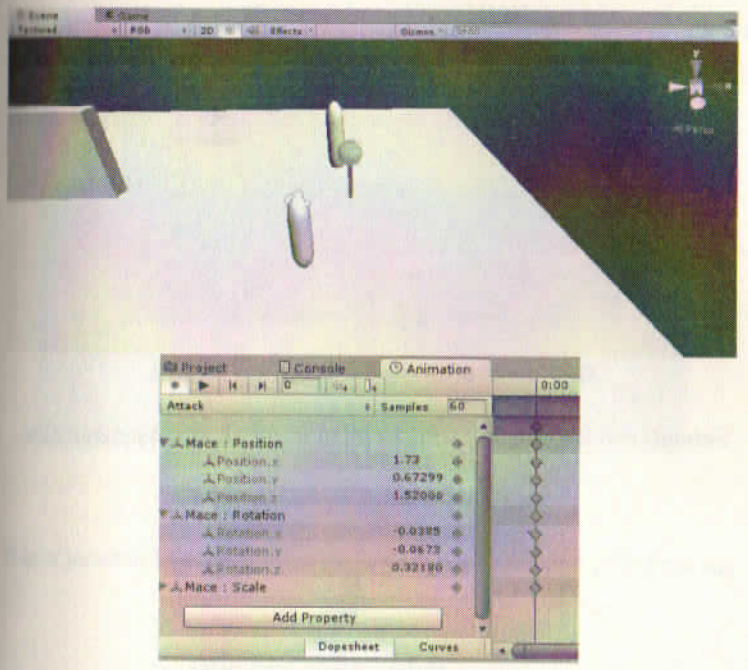
Gambar 2.52 Tampilan jendela Animation

12. Selanjutnya, klik **Add Property** > **Transform** dan pilih semua komponen Transform, yaitu **Position**, **Rotaion**, dan **Scale**.



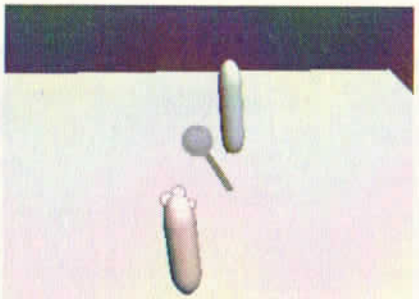
Gambar 2.53 Menambahkan objek senjata dalam jendela Animation

13. Posisi objek Mace pada titik 0 seperti pada Gambar 2.54.

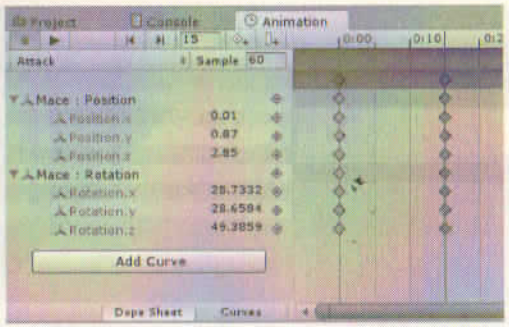


Gambar 2.54 Posisi awal senjata

14. Posisi senjata saat menekan mouse akan seperti Gambar 2.55. Posisi ini dibuat pada titik detik ke 10.

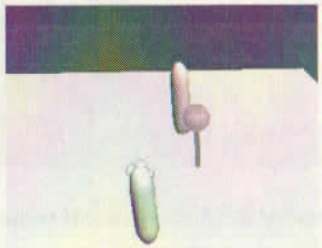


Gambar 2.55 Posisi senjata saat menekan mouse

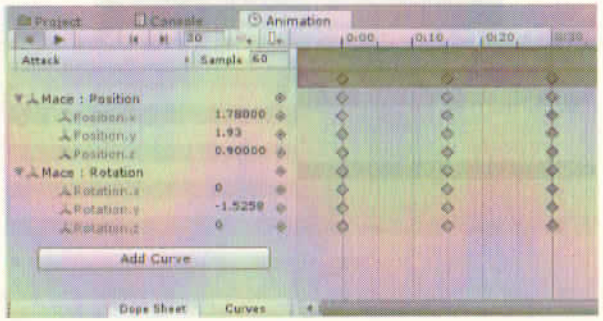


Gambar 2.56 Pengaturan posisi senjata

15. Setelah mouse dilepas, senjata akan kembali ke posisi semula.

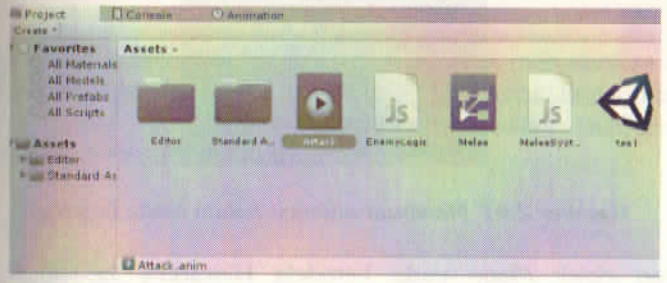


Gambar 2.57 Posisi senjata kembali



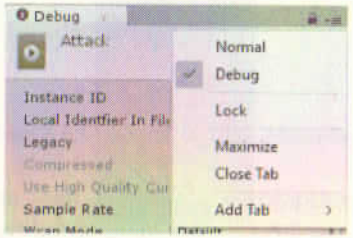
Gambar 2.58 Pengaturan posisi senjata kembali ke posisi semula

16. Klik tombol **Play** untuk mencoba menjalankan animasi dan klik tombol merah bulat atau **Ctrl+S** untuk menyimpan animasi.
17. Buka kembali jendela **Project** dan klik **Attack**.



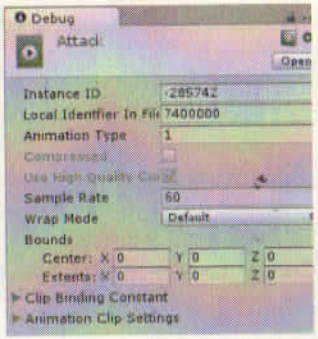
Gambar 2.59 Klik animasi Attack

18. Pada jendela **Hierarchy** klik tombol kanan atas dan pilih **Debug.**



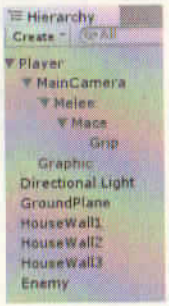
Gambar 2.60 Memilih mode Debug

19. Pada Unity 5, pada bagian **Legacy** beri tanda untuk membuat animasi dalam mode Legacy. Pada Unity versi lama ketik 1 pada **Animation Type**.



Gambar 2.61 Membuat animasi dalam mode Legacy

20. Drag objek **Mace** pada kemdela **Hierarchy** ke objek **Melee** sehingga sekarang objek **Mace** menjadi child dari objek **Melee**.



Gambar 2.62 Membuat objek Mace menjadi child dari Objek Melee

21. Buka kembali skrip **MeleeSystem**. Tmabakan variabel "Mace".

```
var Mace : Transform;
```

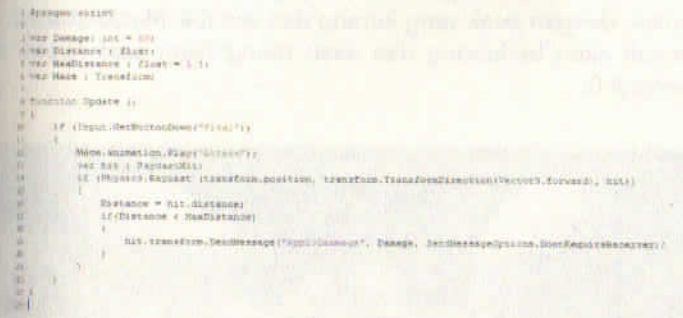
22. Selanjutnya tambahkan animasi **Attack** ke dalam skrip seperti berikut:

```
function Update ()
{
        if (Input.GetButtonDown("Fire1"))
        {
                Mace.animation.Play("Attack");
                var hit : RaycastHit;
                if (Physics.Raycast (transform.position,
transform.TransformDirection(Vector3.forward), hit))
                {
                        Distance = hit.distance;
```



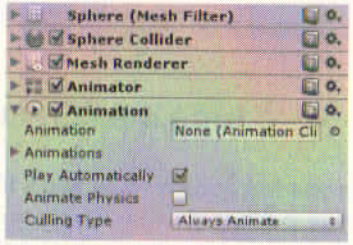
```
                if(Distance < MaxDistance)
                {
                        hit.transform.SendMessage("ApplyDammage", Damage,
SendMessageOptions.DontRequireReceiver);
                }
        }
}
```

Bagian yang ditebalkan adalah perintah untuk menjalankan animasi Attack. Perintah ini berlaku untuk Unity versi lama. Sedangkan pada Unity versi 5, Anda dapat mengetikkan: GetComponent <Animation>().Play("Attack");.



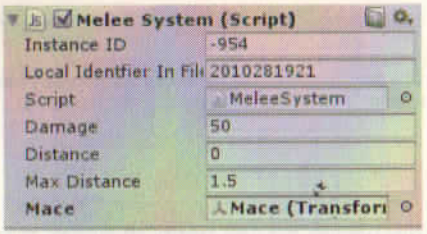
Gambar 2.63 Skrip lengkap MeleeSystem

23. Klik objek **Mace** dan tambahkan komponen **Animation**. Pada bagian **Animation** tambahkan animasi **Attack**.



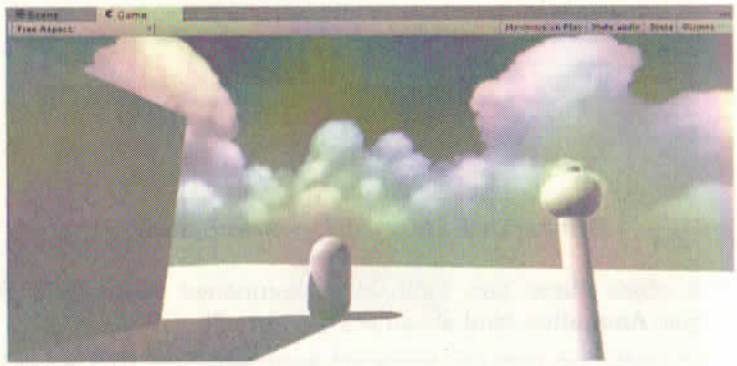
Gambar 2.64 Memasukkan animasi Attack ke objek Mace

24. Klik objek **Melee**. Anda akan melihat komponen **Mace**. Pilih **Mace** ke dalam komponen.



Gambar 2.65 Menambahkan pilihan Mace

25. Klik tombol **Play** untuk menguji game. Ketika Anda menekan tombol mouse, senjata akan bergerak. Jika Anda menyerang musuh dengan jarak yang kurang dari variable **MaxDistance**, nyawa musuh akan berkurang dan akan menghilang saat nyawa musuh menjadi 0.



Gambar 2.66 Menguji game

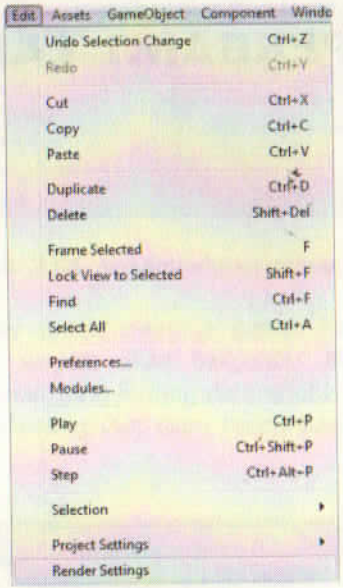
3 Memperbaiki Tampilan Peta

Saat ini Anda tentu telah mendapat gambaran game yang akan dibuat pada buku ini. Untuk mengembangkan rancangan dasar dari game, langkah pertama yang akan Anda buat adalah peta. Pada bab sebelumnya peta hanya berbentuk segi empat dan karakter dapat terjatuh dari peta. Pada bab ini Anda akan memperbaiki tampilan peta sehingga menjadi lebih menarik dan ada pembatas yang membuat karakter tidak dapat terjatuh dari peta.

3.1 Membuat Efek Kabut

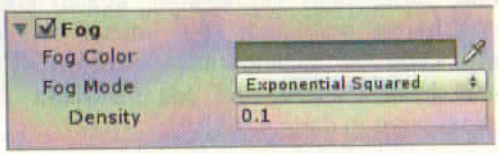
Pada pembahasan ini Anda akan menambahkan efek kabut pada peta. Tujuannya, sebagai pambatas jarak pandang karakter sekaligus untuk membuat efek misteri pada game yang dapat menjadi nilai tambah dalam suatu game survival. Untuk melakukannya, ikuti langkah-langkah berikut:

1. Klik **Edit** > **RenderSetting**. Pada Unity versi 5 menu ini ada di **Windows** > **Lightning**.



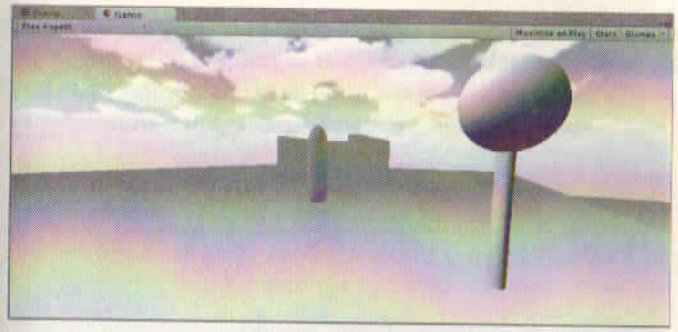
Gambar 3.1 Klik Edit > Render Setting

2. Lihat pada bagian **Fog**. Anda akan melihat pengaturan tampilan efek kabut.



Gambar 3.2 Pengaturan Fog

3. Beri tanda pada **Fog** seperti Gambar 3.2 untuk menampilkan efek kabut.
4. Pada bagian **Fog Color** biarkan bewarna abu-abu untuk menyesuaikan dengan kondisi kabut yang sebenarnya.
5. **Density** merupakan tingkat ketebalan kabut. Pada Gambar 3.2 diisi dengan "0.1" supaya kabut tidak terlalu tebal.
6. Jika Anda klik tombol **Play**, Anda akan melihat efek kabut terlihat dalam game.

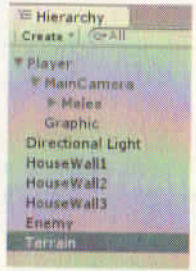


Gambar 3.3 Efek kabut dalam game

3.2 Mengganti Bentuk Peta

Pada saat membuat rancangan game, Anda membuat peta dengan menggunakan objek. Pada pengembangan game ini Anda akan mengganti peta tersebut dengan Terrain. Terrain merupakan objek yang digunakan khusus untuk membuat peta. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Hapus objek **GroundPlane** yang sebelumnya digunakan sebagai peta.
2. Selanjutnya, klik **GameObject** > 3D **Object** > **Terrain**.
3. Anda akan melihat objek **Terrain** di jendela **Hierarchy**.

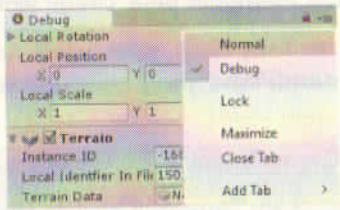


Gambar 3.4 Objek Terrain pada jendela Hierarchy

4. Jika Anda lihat pada jendela **Scene**, Anda akan melihat peta yang lebih besar dari sebelumnya.

Gambar 3.5 Tampilan Terrain dengan objek karakter di kiri bawah

5. Klik objek **Terrain** untuk memunculkan pengaturan pada jendela **Inspector**. Jika kondisi jendela **Inspector** masih dalam keadaan debug, klik pada ikon gear dan pilih **Normal**.

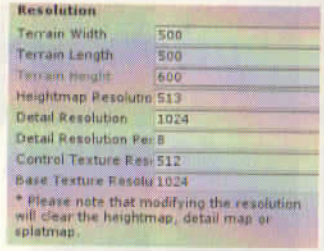


Gambar 3.6 Memilih mode Normal

6. Pada bagian **Terrain** klik ikon plaing kanan untuk menampilkan pengaturan terrain.

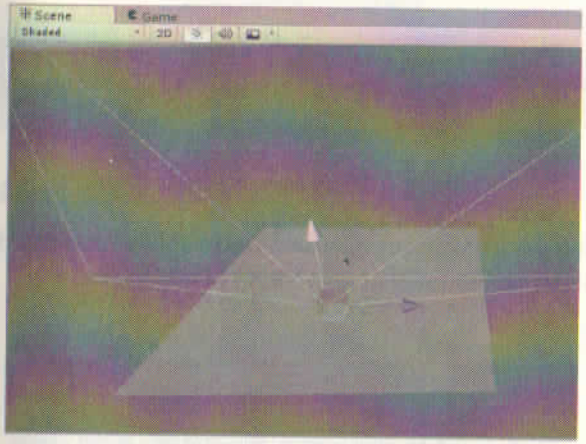
Gambar 3.7 Klik ikon Terrain Setting

7. Lihat pada bagian **Resolution**. Atur pangjang, lebar, dan tinggi terrain seperti Gambar 3.8. Tinggi terrain digunakan untuk membatasi ketinggian objek yang dibuat pada terrain.



Gambar 3.8 Pengaturan panjang, lebar, dan tinggi terrain

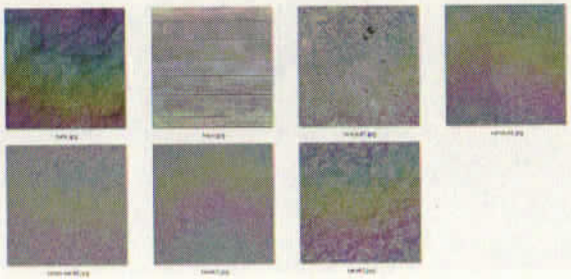
8. Jika Anda lihat pada jendela **Scene**, Anda melihat objek **Player** berada terlalu ke kiri bawah. Oleh karena itu, Anda perlu menggeser terrain sehingga objek Player ada di tengah. Gunakan tool **Position** atau mengubah **Position** pada jendela **Inspector**.



Gambar 3.9 Objek Player berada di tengah terrain

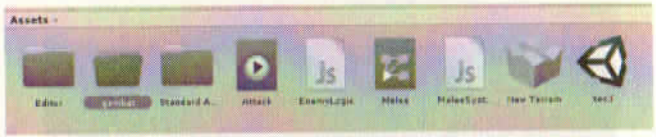
9. Untuk membuat pola pada peta, Anda perlu menambahkan gambar pola bentuk tanah. Anda dapat mengambil gambar ini dari internet dengan kata kunci "texture". Anda juga dapat men-download gambar untuk yang digunakan pada buku ini di

https://drive.google.com/file/d/0BwzsZbp4m4sUbmNXWFBWejl RnM/view?usp=sharing. Contoh gambar yang digunakan seperti pada Gambar 3.10.



Gambar 3.10 Pilihan pola permukaan tanah

10. Masukkan gambar ke dalam Unity dengan membuat folder gambar terlebih dahulu. Caranya, klik kanan pada jendela **Project** dan pilih **Create > Folder**.

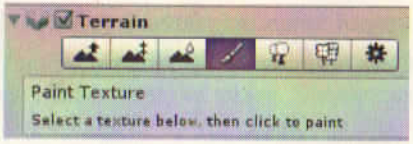


Gambar 3.11 Membuat folder baru

11. Selanjutnya, masukkan gambar pada folder dengan klik folder. Kemudian klik kanan dan pilih **Import New Asset**.

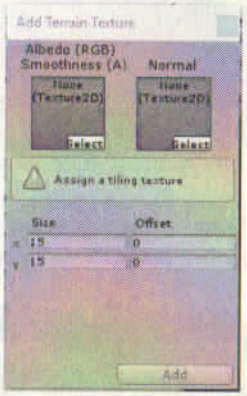
Gambar 3.12 Menambahkan gambar ke dalam Unity.

12. Pada jendela Inspector dari Terrain, klik Paint Texture.



Gambar 3.13 Memilih Paint Texture

13. Klik **Edit Texture** sehingga muncul kotak dialog **Add Terrain Texture**.

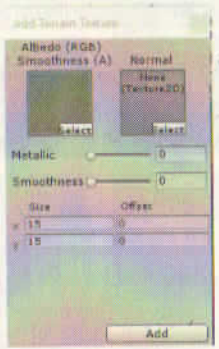


Gambar 3.14 Kotak dialog Add Terrain Texture

14. Klik **Select** pada kotak kiri dan ketikkan rumput.

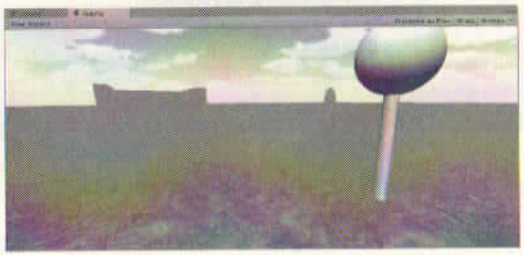
Gambar 3.15 Pilihan texture

15. Pilih tekstur rumput sehingga muncul di kotak dialog **Add Terrain Texture**.



Gambar 3.16 Gambar texture muncul di kotak dialog Add Terrain Texture

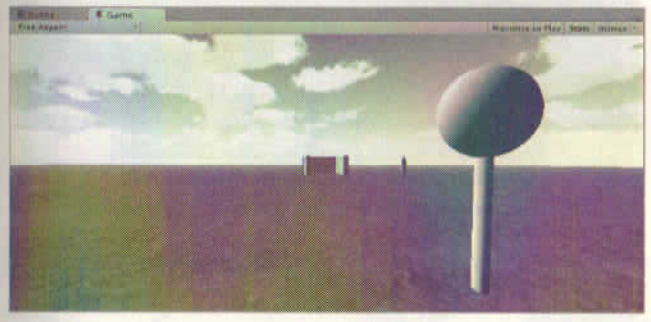
16. Pada bagian **Size** buat **x** dan **y** menjadi "15". Pengaturan ini merupakan ukuran texture setiap kotaknya. Kemudian klik **Add**.



Gambar 3.17 Tekstur pada peta game

Gambar 3.18 Kotak tekstur pada peta game

17. Jika Anda merasa efek kabut terlalu tebal, Anda dapat mengurangi angka dari **Density**. Selain itu Anda juga dapat membuat objek **Light** menjadi lebih dekat ke objek **Player** supaya peta menjadi lebih terang.

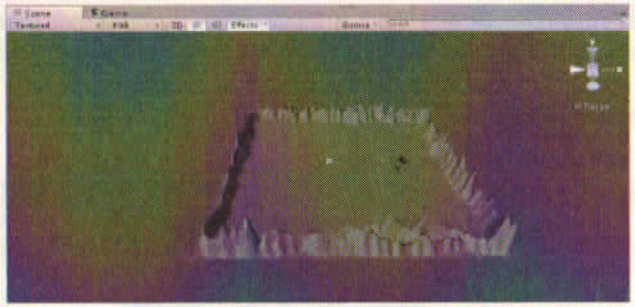


Gambar 3.19 Tampilan peta dengan Densiti kabut 0.05

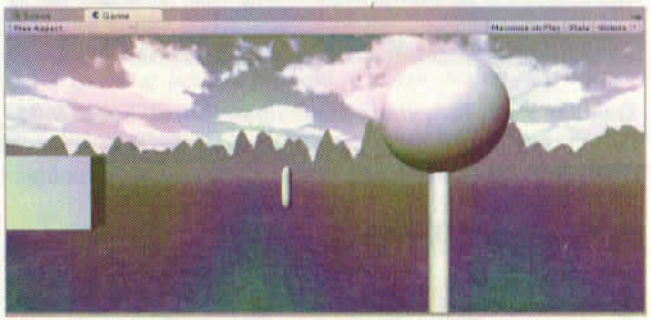
3.3 Menambahkan Gunung

Untuk membatasi bagian pinggir peta, Anda akan menambahkan gunung. Gunung ini akan melingkari peta sehingga karakter tidak dapat lompat hingga keluar dari peta. Untuk lebih jelas, ikuti langkah-langkah berikut:

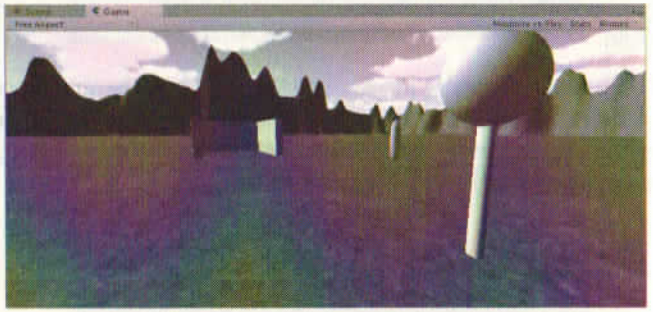
1. Klik objek **Terrain** sehingga menampilkan jendela **Inspector** dari Terrain.
2. Pada bagian **Terrain** Anda akan melihat beberapa tool untuk membuat perubahan pada terrain. Gunakan tool **Raise/Lower Terrain** atau **Paint Height Terrain** untuk membuat gunung. Contoh pembuatan gunung seperti Gambar 3.20.



Gambar 3.20 Gunung memutari peta

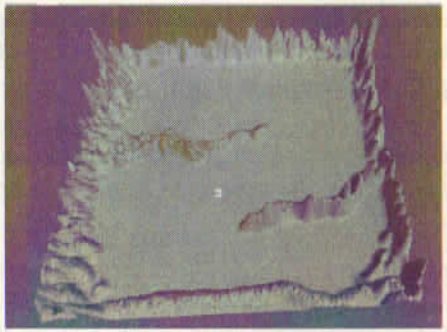


Gambar 3.21 Tampilan gunung dengan efek kabut dalam game



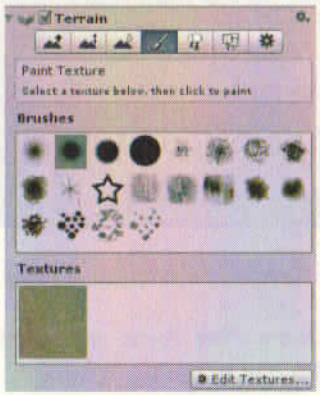
Gambar 3.22 Tampilan gunug tanpa efek kabut dalam game

3. Tambahkan gambar gunung di tengah peta.



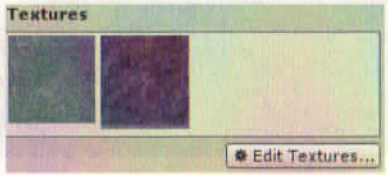
Gambar 3.23 Menambahkan gambar gunung

4. Selanjutnya, pilih tool **Paint Texture**.



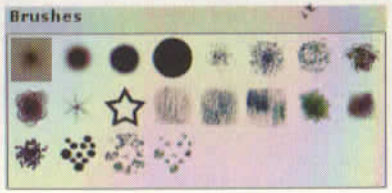
Gambar 3.24 Pengaturan Paint Texture

5. Klik **Edit Texture > Add Texture** untuk mengganti gambar texture dan pilih gambar batu.

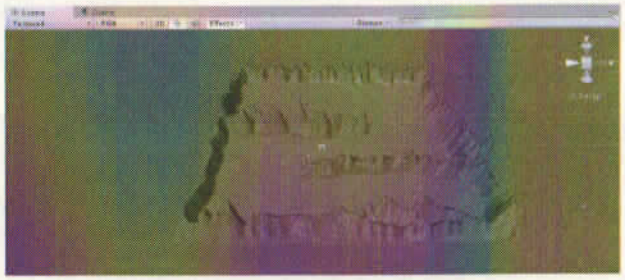


Gambar 3.25 Texture baru dengan pola batu

6. Pilih texture batu yang baru ditambahkan dan pilih tool dengan lingkaran paling kecil. Kemudian terapkan pada peta untuk membuat pola pada bagian gunung.



Gambar 3.26 Memilih lingkaran paling kecil



Gambar 3.27 Hasil menerampkan tekstur pada gunung

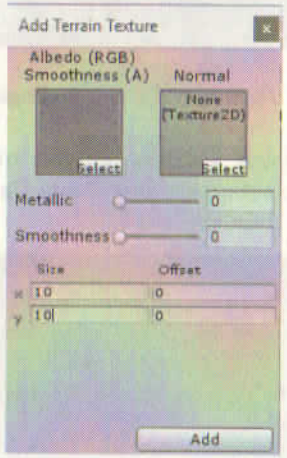
7. Jika Anda klik tombol **Play** Anda akan melihat perbedaan warna tekstur antara tanah dan gunung.

Gambar 3.28 Perbedaan tekstur gunung dan tanah

3.4 Menambahkan Material

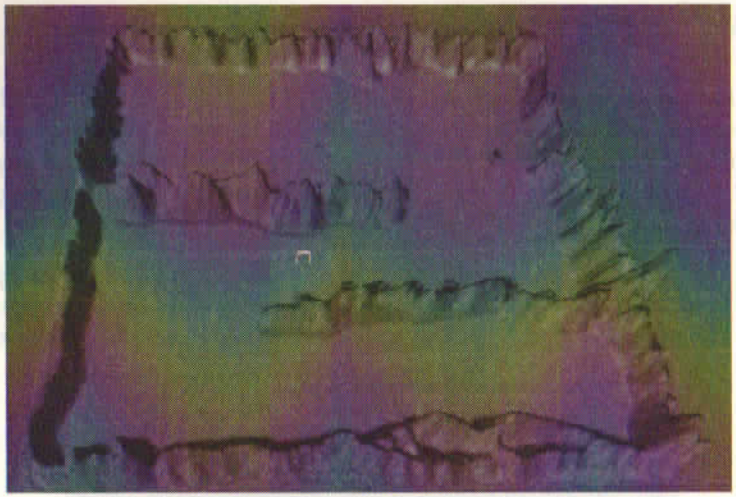
Untuk membuat peta menjadi lebih berisi, Anda akan menambahkan beberapa material. Material yang dapat ditambahkan adalah membuat jalan setapak. Selain itu Anda juga akan menghias bentuk dinding yang putih menjadi memiliki pola. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Bagian pertama yang akan Anda buat adalah membuat jalan setapak. Jalan ini akan memiliki pola tanah berwarna cokelat. Caranya klik objek **Terrain** untuk menampilkan jendela **Inspector**.
2. Tambahkan tekstur tanah dengan ukuran 10x10.



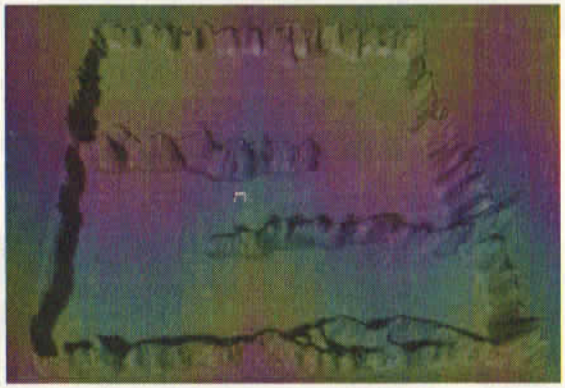
Gambar 3.29 Menambahkan tekstur tanah dengan ukuran 10x10

3. Gunakan tool lingkaran untuk membuat tanah melingkar di sekitar tempat berdirinya objek **Player**.



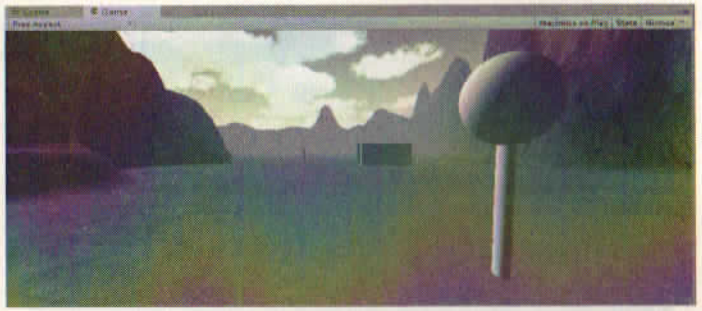
Gambar 3.30 Membuat daerah melingkar pada peta

4. Selanjutnya buat jalan setapak dengan menggunakan tool yang sama. Contoh jalan setapak seperti pada Gambar 3.31.



Gambar 3.31 Membuat jalan setapak

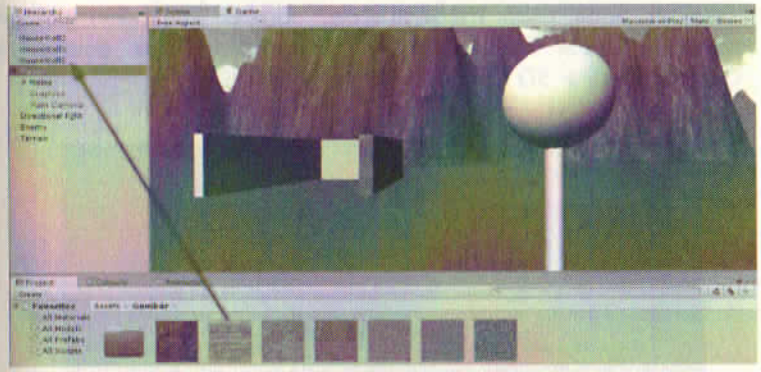
5. Jika Anda klik tombol **Play** Anda akan melihat tanah berwarna cokelat dengan pola seperti yang Anda gambar.



Gambar 3.32 Tampilan jalan setapak dalam game

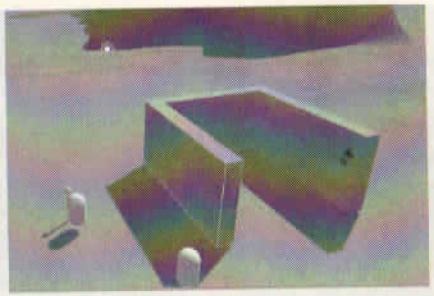
6. Selanjutnya, Anda akan membuat pola pada tembok. Caranya, drag gambar pola tembok ke objek tembok.

Gambar 3.33 Pola tembok



Gambar 3.34 Drag gambar ke objek supaya objek berubah warna

7. Lakukan langkah yang sama sehingga semua tampilan tembok berubah menjadi memiliki pola.

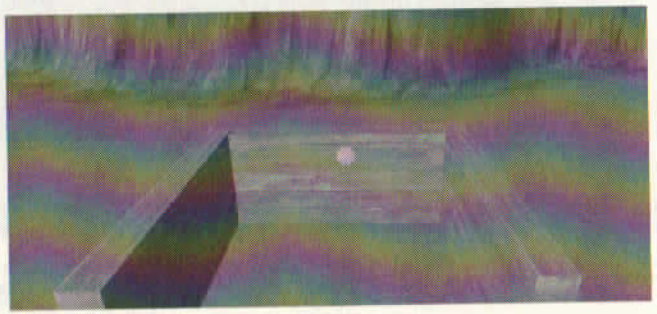


Gambar 3.35 Semua tombok telah memiliki pola

8. Jika Anda lihat pada jendela **Inspector**, Anda melihat material tembok juga telah berubah.

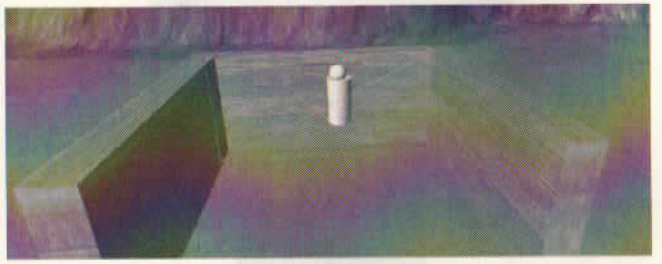
Gambar 3.36 Perbahan material tembok

9. Terakhir Anda akan menambahkan lampu yang ditempel pada tembok. Untuk melakukannya, buat objek bola dengan klik **GameObject > 3D Object > Sphere**.



Gambar 3.37 Menambahkan objek bola

10. Selanjutnya, tambahkan objek silinder sebagai tangkai lampu dengan klik **GameObject > 3D Object > Cylinder**.



Gambar 3.38 Menambahkan objek silinder

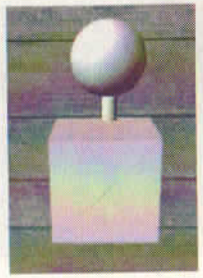
11. Atur ukuran silinder supaya proporsional dengan lampu. Gunakan tool **Scale**.

Gambar 3.39 Memperkecil ukuran silinder

12. Tempelkan silinder pada objek bola.

Gambar 3.40 Menempelkan silinder ke objek bola

13. Tambahkan objek kotak dengan klik **GameObject > 3D Object > Cube**.

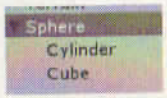


Gambar 3.41 Menambahkan objek kotak

14. Sesuaikan ukuran objek kotak dengan dua objek sebelumnya dan gabungkan semua objek.

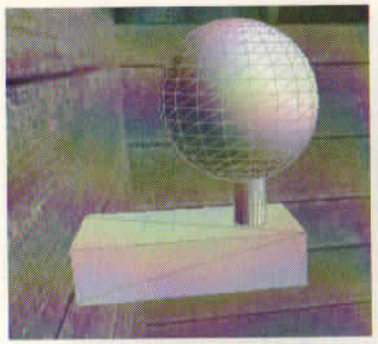
Gambar 3.42 Menggabungkan ketiga objek

15. Gabungkan objek silinder dan kotak ke dalam objek bola.



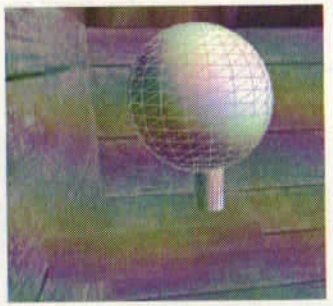
Gambar 3.43 Membuat objek silinder dan kotak menjadi child objek bola

16. Ubah nama objek bola menjadi "Lamp".
17. Tempelkan objek ke tembok.



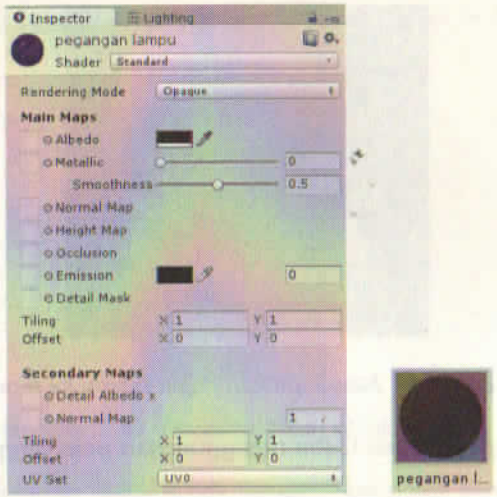
Gambar 3.44 Menempelkan objek lampu ke tembok

18. Beri pola pada objek **Cube** menggunakan pola yang sama dengan tembok.



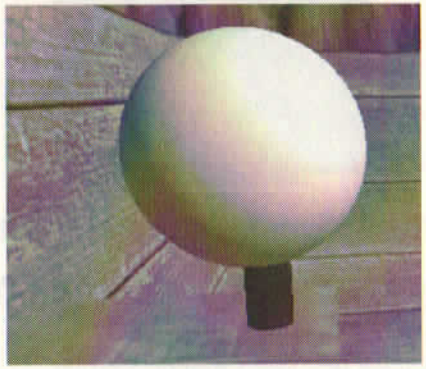
Gambar 3.45 Menambahkan pola pada objek cube

19. Buat material dengan klik kanan pada jendela **Project**. Kemudian pilih **Create Material**.
20. Beri nama material dengan "Pegangan lampu".
21. Beri warna hitam pada material dengan klik bagian warna pada jendela **Inspector**.



Gambar 3.46 Pengaturan warna dan hasil pewarnaan material

22. Drag material ke objek **Cylinder**.

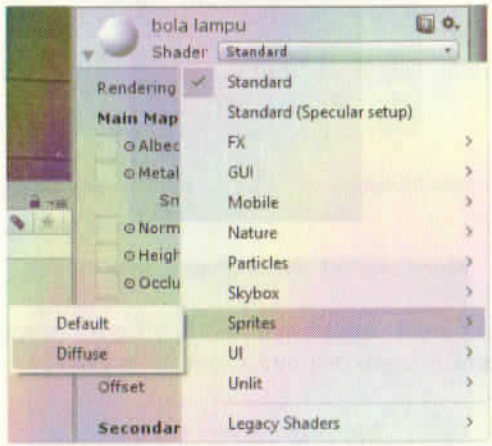


Gambar 3.47 Pegangan lampu menjadi hitam

23. Buat kembali material warna putih untuk bagian bola lampu.
24. Berikan warna putih pada material.

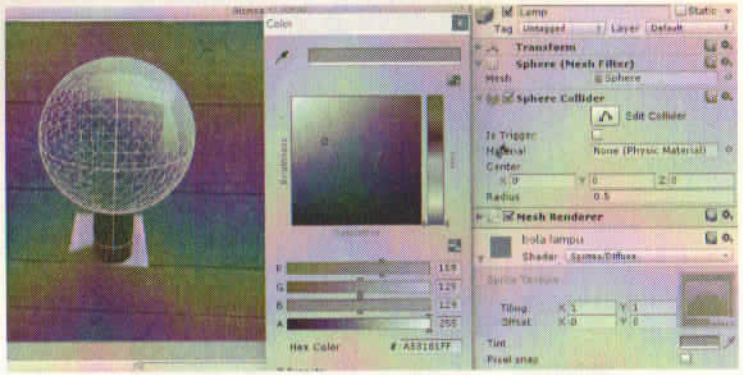
Gambar 3.48 Menambahkan material warna pada objek bola lampu

25. Pada jendela **Inspector** Anda akan melihat bagian **Material** dapat diedit. Klik bagian **Shader** dan pilih **Sprite** > **Diffuse**.



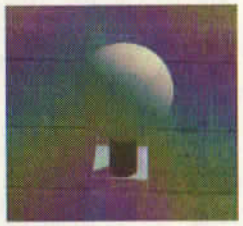
Gambar 3.49 Memberi warna pada bola lampu

26. Selanjutnya, klik bagian **Tint** sehingga ditampilkan kotak dialog untuk memilih warna.
27. Pilih warna untuk bola lampu dengan warna yang Anda inginkan.



Gambar 3.50 Pengaturan warna lampu

28. Hasilnya warna lampu berubah.



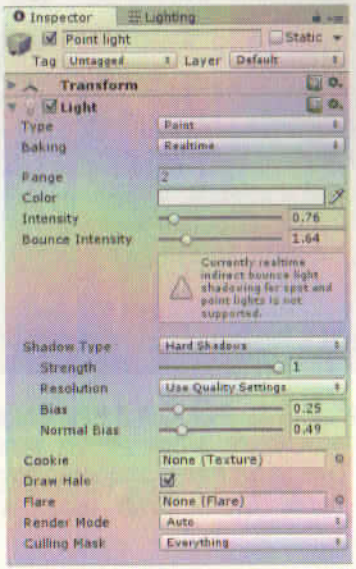
Gambar 3.51 Perubahan warna lampu

29. Selanjutnya, buat efek cahaya di dalam lampu dengan klik **GameObject > Light > point Light**.

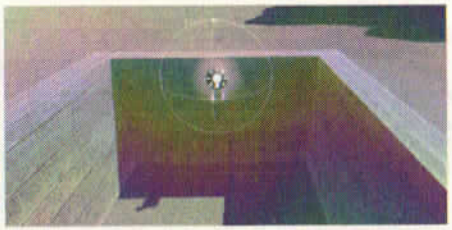
Gambar 3.52 Menambahkan efek cahaya

30. Letakkan cahaya di dalam objek bola lampu.

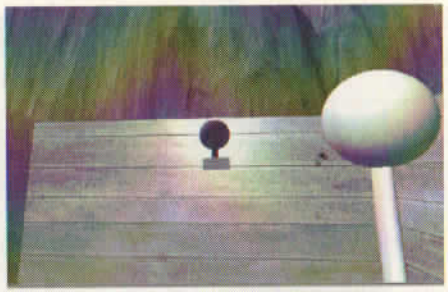
31. Pada jendela **Inspector** lakukan pengaturan seperti menambahkan bayangan dan ukuran cahaya seperti Gambar3.53.



Gambar 3.53 Pengaturan cahaya lampu



Gambar 3.54 Tampilan cahaya bola lampu

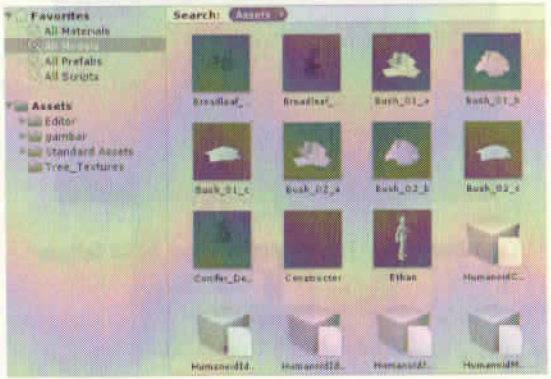


Gambar 3.55 Tampilan lampu dalam game

3.5 Membuat Objek Pohon

Untuk membuat objek pohon, ada dua cara yang dapat Anda lakukan. Pertama adalah menambahkan objek pohon yang telah jadi. Kedua, menggunakan asset Tree yang hanya menampilkan batang pohon dan ditambahkan objek daun. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Objek pohon sebenarnya telah ada saat Anda menambahkan asset Terrain. Untuk mencari objek pohon, pada jendela **Project** pilih kategori **All Models**.

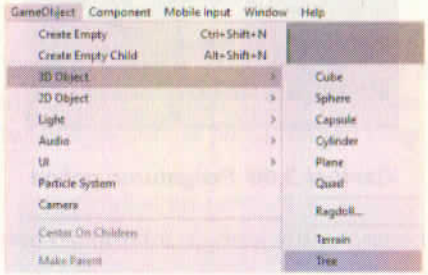


Gambar 3.56 Tampilan semua model yang ditambahkan ke dalam Unity

2. Berdasarkan Gambar 3.56, Anda dapat melihat beberapa objek pohon. Drag objek ke jendela **Scene** untuk menampilkan pohon.

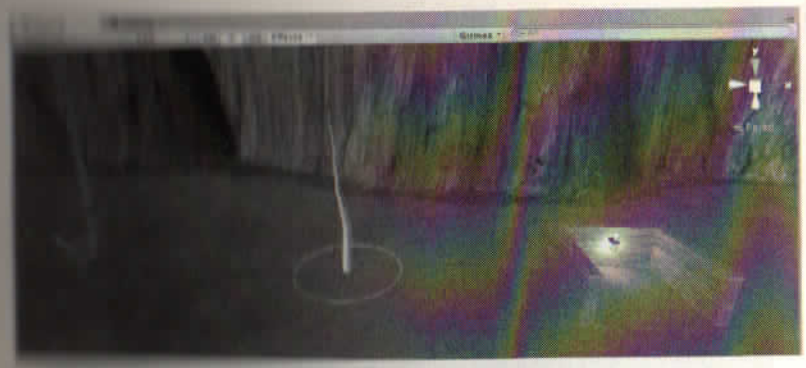
Gambar 3.57 Menambahkan objek pohon

3. Cara lain menambahkan pohon dengan klik **GameObject > 3D Object > Tree**.



Gambar 3.58 Klik GameObject > 3D Object > Tree

4. Anda akan melihat batang pohon di jendela **Scene**.



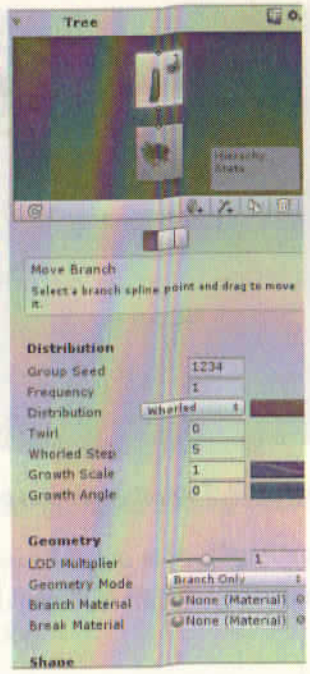
Gambar 3.59 Objek batang pohon

Pada panel **Inspector** dari objek **Tree** Anda melihat pengaturan batang dari pohon.

3.60 Pengaturan pohon

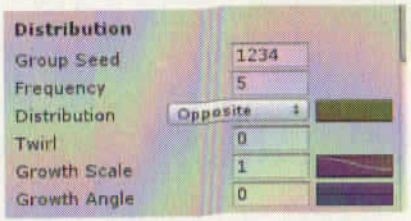
...engatur tampilan batang.

...0
...sekitar



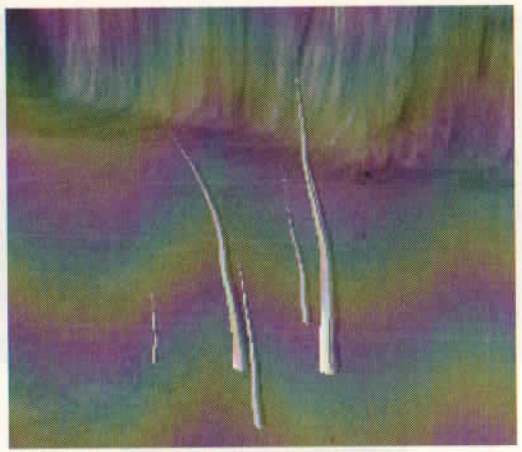
Gambar 3.61 Pengaturan batang

7. Pada bagian **Distribution**, isi bagian **Frequency** dengan "5" supaya batang pohon menjadi bercabang 5. Kemudian pada **Distibution** pilih **Opposite** sebagai bentuk percabangan yang tersebar.



Gambar 3.62 Pengaturan bagian Distribution

8. Anda akan melihat batang lima pohon yang tersebar dalam area melingkar.



Gambar 3.63 Tampilan batang pohon

9. Pada bagian **Geomatry** pilih pola untuk **Branch Material** dan **Break Material** dengan pola yang sesuai sehingga batang pohon menjadi memiliki pola.

Gambar 3.64 Batang pohon menjadi memiliki pola

10. Selanjutnya, klik pada gambar pohon pada Gambar 3.60 untuk mengatur tampilan pohon.

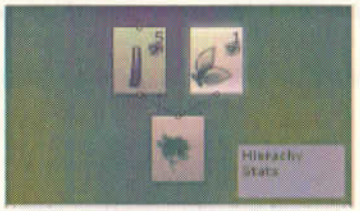
11. Pada bagian **Distribution** buat **Area Spread** menjadi "0" supaya cabang pohon menyatu menjadi satu pohon yang bercabang.

Gambar 3.65 Membuat cabang pohon menyatu

12. Tambahkan gambar daun pada **Gambar 3.60** untuk menambahkan dau pada batang pohon. Caranya, klik gambar daun di bawah kotak dialog.

Gambar 3.66 Klik gambar daun

13. Anda akan melihat tambahan gambar daun.

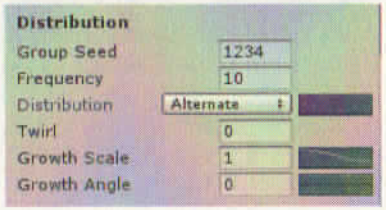


Gambar 3.67 Menambahkan pengaturan daun

14. Klik gambar daun di jendela **Scene** untuk mengatur posisi daun di pohon.

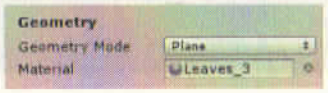
Gambar 3.68 Menyesuaikan posisi daun

15. Pada bagian **Distribution** buat **Frequency** dengan "10" untuk menentukan banyaknya daun. Kemudian pilih **distribution** dengan **Alternate** untuk memnentukan penyebaran daun.



Gambar 3.69 Mengatur penyebaran daun

16. Pada bagian **Geometry** pilih pola gambar daun yang sesuai.



Gambar 3.70 Menambahkan pola pada daun

17. Anda akan melihat tampilan pohon yang dilengkapi daun.

Gambar 3.71 Tampilan pohon

18. Anda dapat menggeser posisi daun dengan drag lingkaran.

Gambar 3.72 Tampilan akhir pohon

Gambar 3.73 Tampilan pohon dalam game

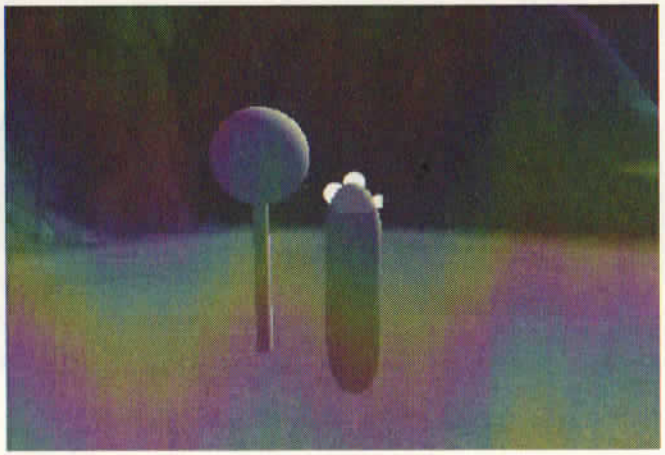
4 MEMBUAT SISTEM GAME

Pada bab ini Anda akan membuat sistem game. Sistem game yang akan dibuat antara lain menambahkan pilihan senjata, pengaturan musuh, dan pengaturan lebih lanjut terhadap karakter. Setelah Anda membuat sistem game, kondisi game akan menjadi lebih menarik. Untuk lebih jelas, ikuti pembahasan pada bab ini.

4.1 Menambahkan Pilihan Senjata

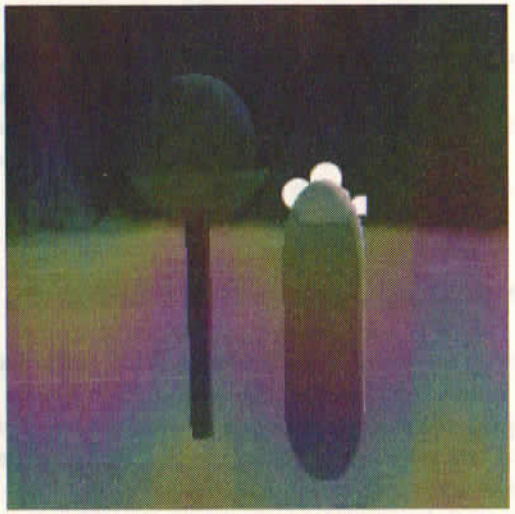
Sampai saat ini Anda hanya memili senjata Mace yang telah dibuat pada Bab 2. Pada bab ini Anda akan menambahkan senjata baru dengan tampilan yang jauh lebih baik. Senjata ini dapat Anda peroleh dari link download yang telah dituliskan pada Bab 3 yang berisi gambar tekstur. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Perbaiki bentuk tampilan objek Mace dengan meperbesar bentuk bola dan memperpendek bagian pegangan/Grip.



Gambar 4.1 Tampilan objek Mace yang baru

2. Tambahkan tekstur pada objek Mace dan Grip supaya senjata memiliki warna. Anda dapat menggunakan tekstur batu sebagai pewarna.

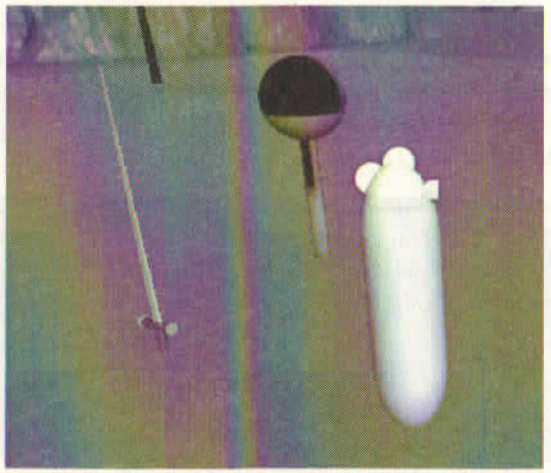


Gambar 4.2 Memberi tekstur apda mace

3. Tambahkan senjata baru dengan menambahkan asset **Sword** dari file yang Anda download.

Gambar 4.3 Menambahkan pedang

4. Tambahkan tekstur pedang dari file yang Anda download.



Gambar 4.4 Menambahkan tekstur pada pedang

5. Selanjutnya, buat skrip baru pada **Melee** dengan nama "WeaponSwitch" dengan menggunakan **Javascript**.

6. Buat variabel seperti berikut.

```
var Weapon1 : GameObject;
var Weapon2 : GameObject;
```

Variabel di atas akan menampilkan dua jenis senjata yang dapat digunakan oleh karakter.

7. Selanjutnya, tulis skrip seperti berikut.

```
function Update ()
{
        if (Input.GetKeyDown(KeyCode.Q))
        {
                SwapWeapon ();
        }
}
function SwapWeapon ()
{
        if (Weapon1.activeInHierarchy == (true))
        {
                Weapon1.SetActive(false);
                Weapon2.SetActive(true);
        }
        else
        {
                Weapon1.SetActive(true);
                Weapon2.SetActive(false);
        }
}
```

Berdasarkan skrip di atas, Anda dapat mengganti senjata dengan klik tombol **Q** pada keyboard. Pada fungsi SwapWeapon akan memberikan perintah hanya satu senjata yang muncul saat Anda menekan Q.

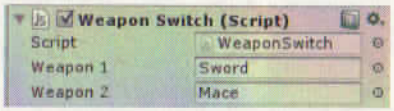
Pada fungsi SwapWeapon Anda akan melihat fungsi activeInHierarchy. Pada bagian ini akan memberikan pilihan saat senjata 1 aktif, maka senjata 2 akan dihilangkan. Hal ini juga berlaku sebaliknya saat enjata dua aktif, senjata satu dihilangkan.


```
#pragma strict

var Weapon1 : GameObject;
var Weapon2 : GameObject;

function Update ()
{
    if (Input.GetKeyDown(KeyCode.Q))
    {
        SwapWeapon ();
    }
}
function SwapWeapon ()
{
    if (Weapon1.activeInHierarchy == (true))
    {
        Weapon1.SetActive(false);
        Weapon2.SetActive(true);
    }
    else
    {
        Weapon1.SetActive(true);
        Weapon2.SetActive(false);
    }
}
```

Gambar 4.5 Tampilan lengkap skrip WeaponSwitch

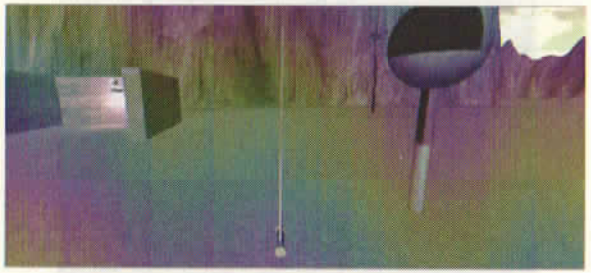
8. Jika Anda lihat jendela **Inspector** dari objek **Melee** Anda melihat komponen baru yang mendaftarkan dua senjata untuk karakter.

9. Masukkan senjata mace dan sword pada kotak yang berbeda.



Gambar 4.6 Mendaftarkan senjata

10. Jika Anda klik tombol **Play** Anda akan melihat senjata berhasil ditampilkan. Untuk mengganti senjata klik **Q**.



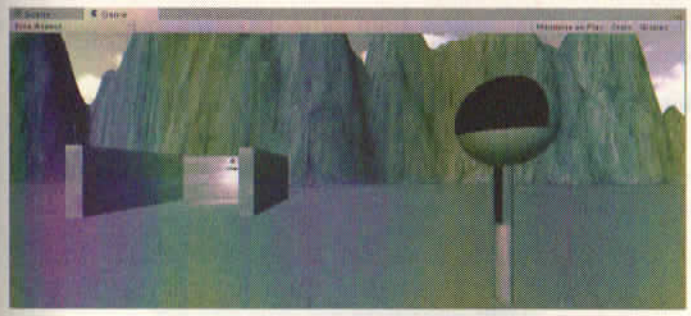
Gambar 4.7 Senjata berhasil ditampilkan

11. Saat Anda memulai game, Anda akan melihat kedua senjata ditampilkan secara bersamaan. Untuk menanganinya, cukup hilangkan tanda pada kotak inspector dari salah satu senjata yang tidak ingin ditampilkan saat awal permainan.

Gambar 4.8 Menghilangkan tanda untuk menyembunyikan senjata

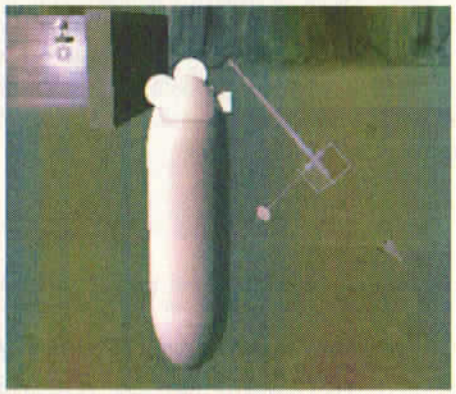
12. Anda akan melihat hanya ada satu senjata pada jendela Scene seperti Gambar 4.9.
13. Jika Anda jalankan kembali game, Anda hanya melihat satu senjata saja yang ditampilkan. Untuk mengganti senjata klik tombol Q seperti Gambar 4.10.

Gambar 4.9 Senjata yang ditampilkan hanya yang diberi tanda pada kotak Inspector



Gambar 4.10 Mengganti senjata dengan menekan Q

14. Senjata Mace dapat Anda gerakkan saat klik kiri mouse, akan tetapi senjata pedang tidak. Hal ini karena pada senjata pedang Anda belum menambahkan animasi gerakan.
15. Untuk membuat animasi, klik objek **Sword** dan klik jendela **Animation**.
16. Buat animasi dengan nama **Sword**.
17. Tambahkan komponen **Position** dan **Rotation** dari **Transform**.
18. Atur pergerakan objek **Sword** sebagai animasi pergerakan objek.

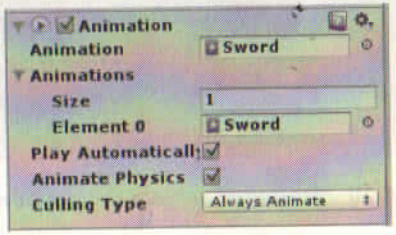


Gambar 4.11 Gerakkan mengayunkan pedang

19. Klik **Ctrl+S** untuk menyimpan animasi jika Anda selesai membuat animasi.

20. Selanjutnya, klik jendela **Inspector** dariobjek **Sword**. Tambahkan komponen **Animation**.

21. Masukkan animasi **Sword** yang telah dibuat.



Gambar 4.12 Menambahkan animasi pada objek Sword

22. Selanjutnya, klik objek **Melee** dan tambahkan **Javascript** dengan nama "MeleeSystemSword".

23. Buat variabel seperti berikut.

```
var Damage : int = 50;
var Distance : float;
var MaxDistance : float = 1.5;
var Weapon : Transform;
```

Variabel di atas sama dengan variabel saat Anda membuat MeleeSystem pada Bab 2. Akan tetapi, bagian var Mace diganti dengan var Weapon. Bagian ini juga dapat Anda terapkan pada MeleeSystem, sehingga pada jendela Inspector objek Melee tidak lagi menampilkan senjata dengan nama senjata (Mace) tetapi akan menampilkan Weapon.

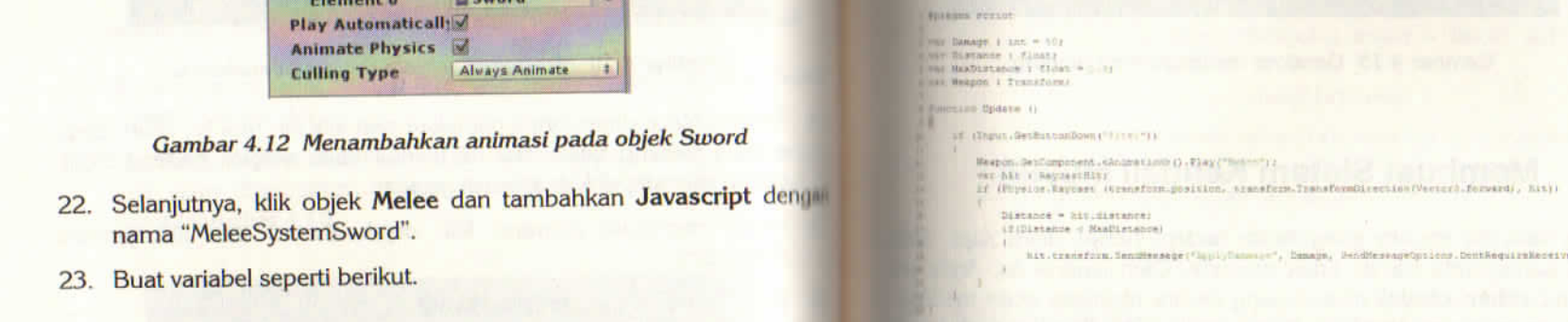
24. Selanjutnya, tulis skrip berikut.

```
function Update ()
{
        if (Input.GetButtonDown("Fire1"))
        {
                Weapon.animation.Play("Sword");
                var hit : RaycastHit;
                if (Physics.Raycast (transform.position,
transform.TransformDirection(Vector3.forward), hit))
                {
                        Distance = hit.distance;
                        if(Distance < MaxDistance)
                        {

        hit.transform.SendMessage("ApplyDammage", Damage,
SendMessageOptions.DontRequireReceiver);
```



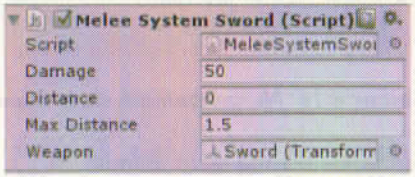
```
                        }
                }
        }
}
```

Penulisan skrip di atas juga sama dengan penulisan skrip MeleeSystem. Perbedaan yang terjadi adalah penggantian variabel Mace menjadi Weapon. Sedangkan pada bagian Play diganti dari Attack menjadi Sword.



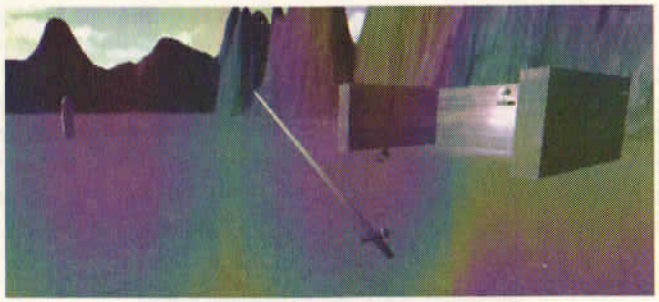
Gambar 4.13 Skrip lengkap sistem pergerakan pedang

25. Selanjutnya, pada jendela **Inspector** dari objek **Melee** daftarkan objek **Sword** pada **Melee System Sword**.



Gambar 4.14 mendaftarkan objek pada komponen Melee

26. Jalankan game dan Anda dapat menggerakkan pedang saat menekan klik kiri.

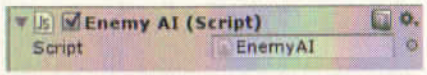


Gambar 4.15 Gerakan mengayunkan pedang

4.2 Membuat Sistem Kendali Musuh

Sampai saat ini musuh yang Anda hadapi hanya diam saja. Dalam sebuah game tentu hal ini tidak menarik. Oleh karena itu, Anda perlu membuat sistem kendali musuh yang secara otomatis akan melakukan perlawanan terhadap karakter. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Klik objek **Enemy** dan tambahkan **Javascript** dengan nama "EnemyAI".



Gambar 4.16 Menambahkan skrip musuh

2. Buka skrip dan hapus semua skrip default.
3. Selanjunya, buat variabel seperti berikut.

```
var Distance : float;
var Target : Transform;
var LookAtDistance = 25.0;
var AttackRange = 15.0;
var MoveSpeed =5.0;
var Damping = 6.0;
```

Variabel di atas digunakan untuk mengatur jarak antara musuh dengan karakter. Selain itu variabel di atas juga mengatur kecepatan gerak musuh. Variabel Damping adalah variabel

tambahan untuk menghaluskan gerakan karakter. Tulisan # pragma strict juga dihapus.

4. Selanjutnya, tuliskan skrip berikut.

```
function Update ()
{
        Distance = Vector3.Distance(Target.position,
transform.position);

        if (Distance < LookAtDistance)
        {
                renderer.material.color = Color.yellow;
                lookAt();
        }
        if (Distance > LookAtDistance)
        {
                renderer.material.color = Color.green;

        }
        if (Distance < AttackRange)
        {
                renderer.material.color = Color.red;
                attack ();
        }
}
function lookAt ()
{
        var rotation =
Quaternion.LookRotation(Target.position -
transform.position);
        transform.rotation =
Quaternion.Slerp(transform.rotation, rotation,
Time.deltaTime * Damping);
}

function attack ()
{
        transform.Translate(Vector3.forward * MoveSpeed *
Time.deltaTime);
}
```

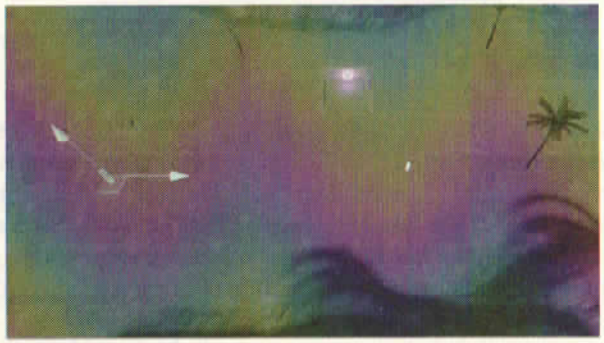
Berdasarkan skrip di atas, untuk mengetahui seberapa dekat karakter terhadap musuh diberikan warna sesuai jarak. Jika karakter di luar jangkauan (LookAtDistance) musuh, objek Enemy akan berwarna hijau. Jika karakter dalam jarak jangkauan musuh, maka objek Enemy berwarna kuning. Jika karakter ada di dalam jangkauan (AttackRange) serangan musuh, musuh akan berwarna merah dan mengejar karakter.

Fungsi lookAt menjelaskan perubahan reaksi objek Enemy terhadap objek Player. Fungsi attack menjelaskan pengaturan kecepatan gerak objek Enemy.

```
#pragma strict

var Distance : float;
var Target : Transform;
var LookAtDistance = 20.0;
var AttackRange = 10.0;
var MoveSpeed = 3.0;
var Damping = 6.0;

function Update ()
{
    Distance = Vector3.Distance(Target.position, transform.position);

    if (Distance < LookAtDistance)
    {
        renderer.material.color = Color.yellow;
        LookAt();
    }
    if (Distance > LookAtDistance)
    {
        renderer.material.color = Color.green;
    }
    if (Distance < AttackRange)
    {
        renderer.material.color = Color.red;
        attack ();
    }
}
function LookAt ()
{
    var rotation = Quaternion.LookRotation(Target.position - transform.position);
    transform.rotation = Quaternion.Slerp(transform.rotation, rotation, Time.deltaTime * Damping);
}

function attack ()
{

}

{
    transform.Translate(Vector3.forward * MoveSpeed * Time.deltaTime);
}
```

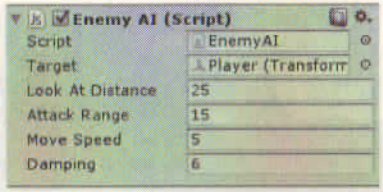
Gambar 4.17 Tampilan skrip lengkap

5. Untuk menguji skrip di atas, letakkan objek **Enemy** agak jauh dari objek **Player**.



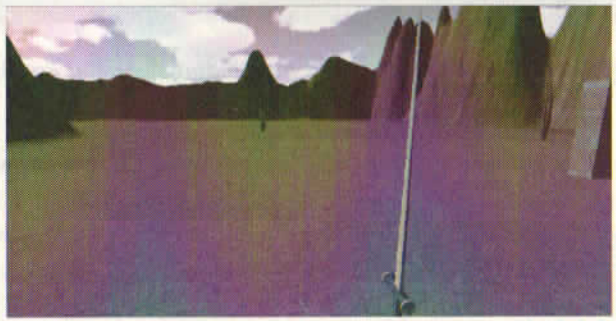
Gambar 4.18 Memberi jarak objek Enemy (kiri) dengan objek Player (kanan)

6. Daftarkan objek **Player** ke komponen skrip objek **Enemy**.

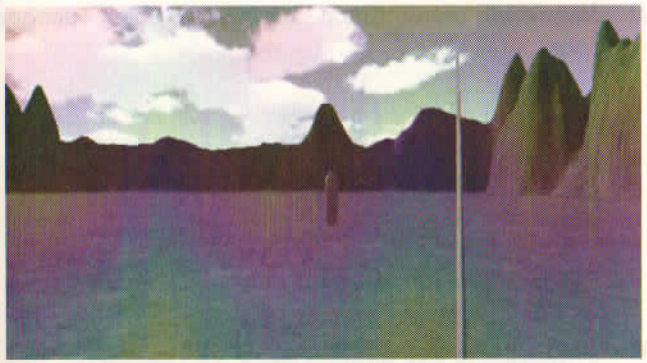


Gambar 4.19 Mendaftarkan objek Player ke objek Enemy

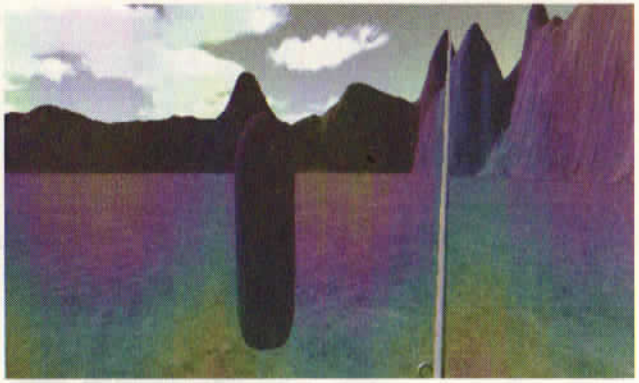
7. Klik tombol **Play** untuk menguji reaksi objek Enemy.



Gambar 4.20 Musuh berwarna hijau saat karakter di luar jangkauan musuh



Gambar 4.21 Musuh menjadi kuning saat karakter masuk dalam wilayah musuh



Gambar 4.22 Musuh berwarna merah dan mengejar karakter

8. Jika saat objek Enemy berwarna kuning dan objek Player menjauh musuh, objek enemy dapat menjadi berwarna hijau kembali.

4.3 Menambahkan Suara

Untuk membuat game menjadi lebih ramai, Anda dapat menambahkan suara dalam game. Pada pembahasan ini Anda akan menambahkan BGM (*background music*) saat karakter ada di posisi tertentu. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Buat **Javascript** baru tanpa membuat objek dengan nama "AudioZone".

Gambar 4.23 Membuat skrip untuk musik

2. Hapus skrip dasar dan tuliskan skrip berikut.

```
private var theCollider : String;

function OnTriggerEnter (other : Collider)
{
        theCollider = other.tag;
        if (theCollider == "Player")
        {
                GetComponent.<AudioSource>().Play();
                GetComponent.<AudioSource>().loop = true;
        }
}

function OnTriggerExit (other : Collider)
{
        theCollider = other.tag;
        if (theCollider == "Player")
        {
                GetComponent.<AudioSource>().Stop();
                GetComponent.<AudioSource>().loop = false;
        }
}
```

Variabel the Collider digunakan sebagai pendeteksi jika karakter memasuki wilayah yang akan dimainkan musik. Pada fungsi OnTriggerEnter mengatur untuk memainkan musik saat karakter memasuki wilayah. Fungsi OnTriggerExit mengatur untuk mematikan musik saat karakter keluar dari wilayah.

Audio digunakan pada Unity5. Pada Unity versi 5 Anda dapat menuliskan `GetComponent.<AudioSource>()` sehingga menjadi

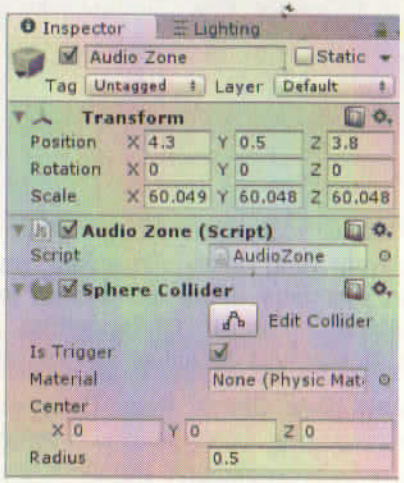
```
GetComponent.<AudioSource>().Play,
GetComponent.<AudioSource> ().Stop, dan
GetComponent.<AudioSource>().loop.
```

```
#pragma strict

private var theCollider : String;

function OnTriggerEnter (other : Collider)
{
    theCollider = other.tag;
    if (theCollider == "Player")
    {
        audio.Play();
        audio.loop = true;
    }
}

function OnTriggerExit (other : Collider)
{
    theCollider = other.tag;
    if (theCollider == "Player")
    {
        audio.Stop();
        audio.loop = false;
    }
}
```

Gambar 4.24 Skrip lengkap dari AudioZone

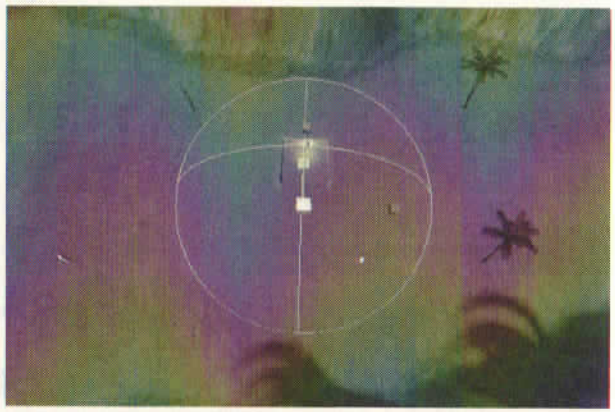
3. Selanjutnya, buat objek dengan klik **GameObject > Create Empty**.
4. Ganti nama objek dengan "Audio Zone".

5. Selanjutnya, tambahkan komponen collider pada objek **Audio Zone** dengan cara klik **Add Component** dan ketikkan "collider".
6. Selanjutnya, pilih **Sphere Colider** dan beri tanda pada **Is Trigger**.



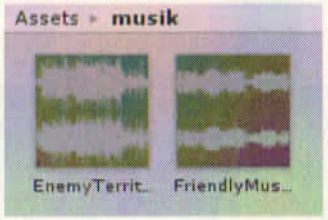
Gambar 4.25 Komponen objek Audio Zone

7. Selanjutnya, letakkan objek di daerah lingkaran seperti Gambar 4.26. Gunakan tool **Scale** untuk memperbesar ukuran objek dan **Position** utuk menggeser objek.



Gambar 4.26 Meletakkan objek Audio Zone

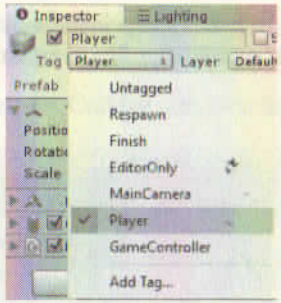
8. Masukkan asset musik yang Anda inginkan. Anda dapat menggunakan asset musik dari file yang Anda download pada Bab 3.



Gambar 4.27 Asset musik yang akan digunakan

9. Drag **FriendlyMusic** atau musik yang Anda inginkan pada jendela **Inspector** dari objek **Audio Zone**.
10. Selanjutnya, beri tanda pada **Play On Awake** seperti Gambar 4.28.
11. Buka jendela **Inspector** dari objek **Player**.
12. Pada bagian **Unttagged** menjadi **Player** seperti Gambar 4.29.
13. Klik tombol **Play** untuk menguji game.

Gambar 4.28 Komponen musik



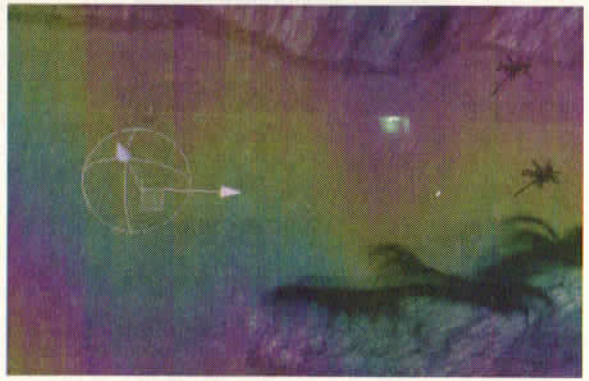
Gambar 4.29 Mengubah Tag objek Player

14. Jika Anda mengatur dengan benar, saat karakter Anda memasuki wilayah Audio Zone akan terdengar musik/BGM. Musik ini akan berhenti jika karakter menginggalkan wilayah Audio Zone. Hal ini dipengaruhi dari **Tag** objek karakter yang telah menjadi **Player**. Selain itu, jika karakter Anda seperti menabrak tembok saat akan masuk ke wilayah Audio Zone, berarti Anda belum memberi tanda pada bagian **Is Trigger** dari komponen **Sphere Collider**.
15. Salin objek **Audio Zone** dan beri nama "Audio Zone 2".
16. Letakkan objek **Audio Zone 2** di daerah objek **Enemy**.

Gambar 4.30 Meletakkan objek Audio Zone 2

17. Selanjutnya, ganti musik dengan asset **EnemyTerritory** atau musik yang Anda inginkan.

18. Saat ini Anda akan memiliki dua musik yang berbeda, yaitu pada daerah lingkaran dan pada wilayah musuh.
19. Jika Anda ingin musik pada wilayah musuh bergerak mengikuti pergerakan musuh, buat objek **Audio Zone 2** menjadi child dari objek **Enemy**.



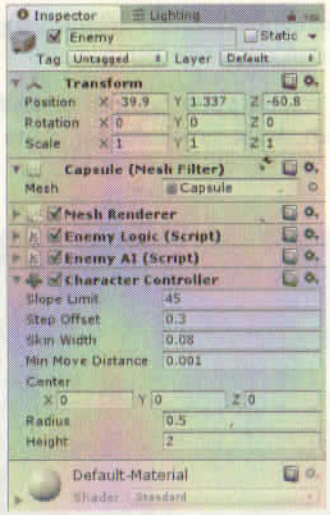
Gambar 4.31 Objek musik menjadi child dari objek musuh

20. Jika Anda jalankan dalam game, musik akan mengikuti pergerakan musuh. Hal ini berbeda dari sebelumnya, di mana saat Anda keluar dari wilayah musik, musik akan menghilang.

4.4 Meningkatkan Sistem Kendali Musuh

Pada pembahasan ini Anda akan meningkatkan kemampuan musuh. Sampai saat ini musuh tidak memberikan efek apapun kepada karakter saat musuh mendekati karakter. Anda akan menambahkan efek seperti musuh dapat membunuh karakter. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Hapus komponen **Capsule Collider** dari jendela **Inspector** objek **Enemy**.
2. Tambahkan komponen **Character Controller** dengan klik **Add Component** dan ketikkan "character controller".



Gambar 4.32 Perubahan komponen objek Enemy

3. Selanjutnya, non aktifkan skrip **EnemyAI** dengan menghilangkan tanda centang pada bagian kiri atas.
4. Buat skrip dengan **Javascript** dan beri nama "AdvancedAI".
5. Salin semua skrip dari **EnemyAI** dan hapus bagian #pragma strict.
6. Selanjutnya, buat variabel menjadi seperti berikut.

```
var Distance;
var Target : Transform;
var LookAtDistance = 25.0;
var ChaseRange = 15;
var AttackRange = 5.0;
var MoveSpeed =5.0;
var Damping = 6.0;
var controller : CharacterController;
var gravity : float = 20.0;
private var MoveDirection : Vector3 = Vector3.zero;
```

Pada variabel di atas, terdapat beberapa tambahan variabel. ChaseRange merupakan variabel untuk menentukan jarak antara musuh dan karakter supaya musuh mengejar karakter. Variabel AttackRange dikurangi menjadi 5. Variabel controller untuk mengendalikan pergerakan musuh. Variabel gravity mejaga supaya musuh tetap menempel di tanah. Variabel MoveDirection akan mengatur pergerakan objek menjadi 3 dimensi (Vector3).

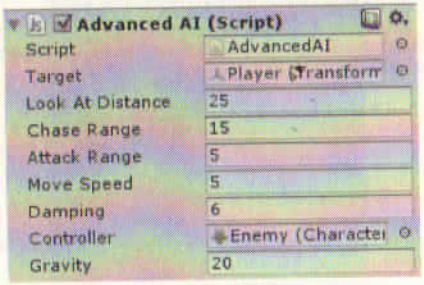
7. Selanjutnya, ketikkan skrip seperti berikut.

```
function Update ()
{
        Distance = Vector3.Distance(Target.position,
transform.position);

        if (Distance < LookAtDistance)
        {
                lookAt();
        }
        if (Distance > LookAtDistance)
        {
                GetComponent.<Renderer>().material.color =
Color.green;
        }
        if (Distance < ChaseRange)
        {
                chase ();
        }
}

function lookAt ()
{
        GetComponent.<Renderer>().material.color =
Color.yellow;
        var rotation =
Quaternion.LookRotation(Target.position -
transform.position);
        transform.rotation =
Quaternion.Slerp(transform.rotation, rotation,
Time.deltaTime * Damping);
}

function chase ()
{
        GetComponent.<Renderer>().material.color =
Color.red;

        moveDirection = transform.forward;
        moveDirection *= MoveSpeed;

        moveDirection.y -= gravity * Time.deltaTime;
        controller.Move(moveDirection * Time.deltaTime);
}
```

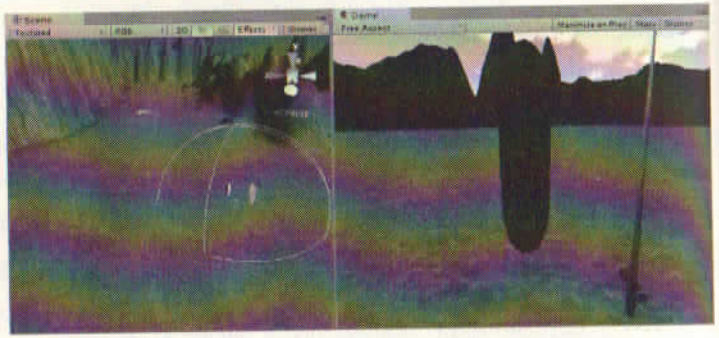
Ada beberapa perubahan dari skrip di atas. Perubahan warna yang sebelumnya di dalam fungsi if langsung dimasukkan ke dalam fungsi tujuan (lookAt dan chase). ChaseRange menggantikan AttackRange sehingga musuh akan mengejar karakter dalam jarak 1.5. Fungsi chase akan mengatur pergerakan musuh saat mengejar karakter.

8. Kembali ke Unity dan daftarkan **CharacterController** yang sebelumnya ditambahkan ke dalam **Advanced AI**.
9. Daftarkan objek **Player** ke dalam **Advanced AI**.

10. Jalankan tombol **Play** untuk menguji pergerakan musuh.

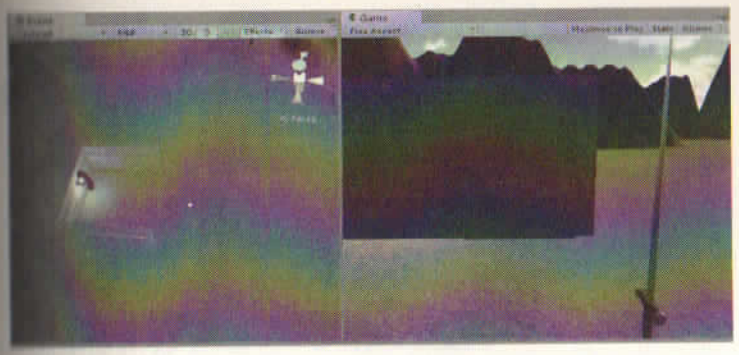


Gambar 4.33 Mendaftarkan Target dan Controlle ke Advanced AI



Gambar 4.34 Objek Enemy yang mengejar objek Player sambil tetap menempel tanah

11. Jika Anda coba menaruh objek **Enemy** di belakang tembok, objek akan mengikuti pergerakan karakter tetapi tidak dapat menembus tembok.



Gambar 4.35 Objek Enemy mengejar objek Player tetapi terhalang tembok

12. Selanjutnya, Anda akan menambahkan serangan pada musuh dengan tambahkan variabel berikut.

```
var attackRepeatTime = 1;

private var attackTime : float;
```

Variabel di atas digunakan supaya musuh dapat melakukan serangan berulang-ulang dengan jeda waktu tertentu.

13. Buat fungsi start seperti berikut.

```
function Start ()
{
        attackTime = Time.time;
}
```

14. Buat fungsi if di dalam fungsi update.

```
if (Distance < AttackRange)
        {
                attack ();
        }
        else
```

Letakkan fungsi ini di atas if (Distance < ChaseRange). Fungsi ini akan membuat objek Enemy berhenti mengejar objek Player jika objek akan menyerang objek Player. Jika objek Player berusaha menjaduhi objek Enemy, objek Enemy akan mengejar kembali objek Player dan akan berhenti saat objek Enemy akan menyerang objek Player.

Untuk membuat supaya objek Enemy berhenti tidak terlalu jauh dari objek Player, Anda dapat mengubah AttackRange menjadi 1.5. Lakukan perubah ini di jendela **Inspector** setelah Anda selesai membuat skrip Advance AI ini.

15. Buat fungsi attack seperti berikut.

```
function attack ()
{
        if (Time.time > attackTime)
        {
                Debug.Log ("Melakukan serangan");
                attackTime = Time.time + attackRepeatTime;
        }
}
```

Fungsi ini akan menyerang karakter jika karakter berada di dalam jarak serang musuh (AttackRange). Pada saat objek Enemy menyerang objek Player, Anda akan melihat pesan "Melakukan serangan" pada jendela Console. Serangan juga dilakukan dengan menggunakan jeda karena Anda telah menambahkan attackTime.

16. Buat fungsi ApplyDamage seperti berikut.

```
function ApplyDammage ()
{
        ChaseRange *= 30;
        MoveSpeed *= 2;
        LookAtDistance *= 40;
}
```

Fungsi ini akan mengatur pergerakan musuh saat musuh menyerang. Dengan pengaturan ini, Anda akan menjadi lebih sulit lari dari objek Enemy.

17. Skrip lengkap dari AdvancedAI seperti berikut.

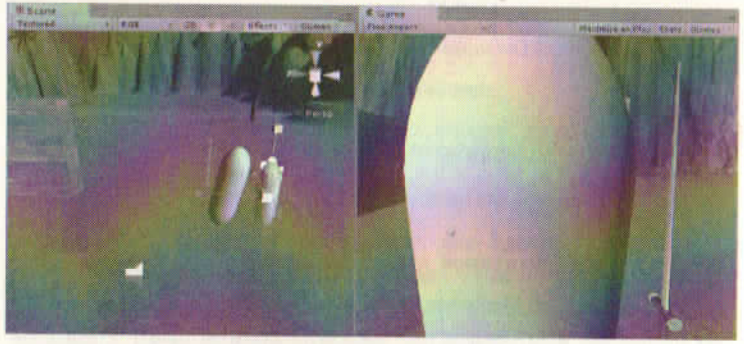
```
var LookAtDistance = 25.0;
var ChaseRange = 15;
var AttackRange = 1.5;
var MoveSpeed =5.0;
var Damping = 6.0;
var attackRepeatTime = 1;

private var attackTime : float;

var controller : CharacterController;
var gravity : float = 20.0;
private var moveDirection : Vector3 = Vector3.zero;

function Start ()
{
        attackTime = Time.time;
```



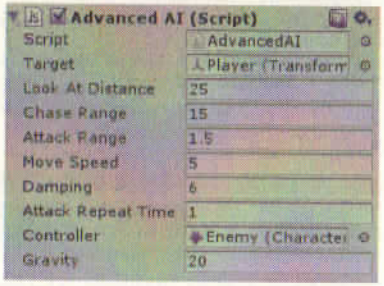
```
}

function Update ()
{
        Distance = Vector3.Distance(Target.position,
transform.position);

        if (Distance < LookAtDistance)
        {
                lookAt();
        }
        if (Distance > LookAtDistance)
        {
                renderer.material.color = Color.green;

        }
        if (Distance < AttackRange)
        {
                attack ();
        }
        else
        if (Distance < ChaseRange)
        {
                chase ();
        }
}
function lookAt ()
{
        renderer.material.color = Color.yellow;
        var rotation =
Quaternion.LookRotation(Target.position -
transform.position);
        transform.rotation =
Quaternion.Slerp(transform.rotation, rotation,
Time.deltaTime * Damping);
}

function chase ()
{
        renderer.material.color = Color.red;

        moveDirection = transform.forward;
        moveDirection *= MoveSpeed;

        moveDirection.y -= gravity * Time.deltaTime;
        controller.Move(moveDirection * Time.deltaTime);
}

function attack ()
{
        if (Time.time > attackTime)
        {
                Debug.Log ("Melakukan serangan");
                attackTime = Time.time + attackRepeatTime;
        }
}

function ApplyDammage ()
{
        ChaseRange *= 30;
        MoveSpeed *= 2;
        LookAtDistance *= 40;
```

}

18. Klik tombol **Play** untuk menguji game.
19. Pada saat musuh mengejar karakter, musuh akan berhenti mengejar (berwarna kuning) untuk menyerang karakter.

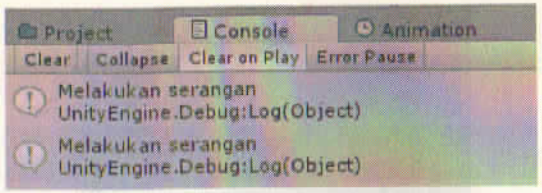


Gambar 4.36 Musuh berhenti mengejar karakter saat akan menyerang



Gambar 4.37 Attack Range diubah menjadi 1.5 supaya musuh berhenti tidak terlalu jauh dari karakter

20. Jika Anda lihat pada jendela **Controller** Anda akan menemukan pesan "melakukan serangan". Pesan ini sebagai tanda musuh sedang menyerang karakter. Tanda ini dilakukan sementara karena karakter belum memiliki nyawa. Pesan ini juga dilakukan dengan menggunakan jeda waktu.



Gambar 4.38 Pesan musuh sedang menyerang karakter

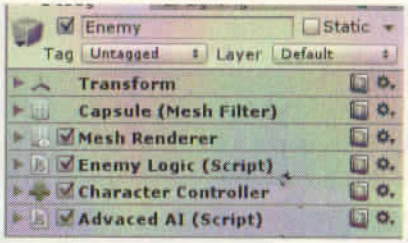
21. Anda juga akan kesuitan untuk menghindari musuh karena perubahan pada fungsi ApplyDamage.

Gambar 4.39 Musuh lebih agresif mengejar karakter

4.5 Membuat Nyawa pada Karakter

Pada pembahasan ini Anda akan menambahkan nyawa pada karakter supaya musuh dapat melukai karakter. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Klik objek **Enemy** dan hapus skrip **EnemyAI**.



Gambar 4.40 Menghapus skrip EnemyAI

2. Buka kembali skrip **AdvancedAI** dari objek **Enemy**.
3. Tambahkan variabel baru seperti berikut.

```
var Damage = 35;
```

4. Pada fungsi attack ubah menjadi seperti berikut.

```
function attack ()
{
        if (Time.time > attackTime)
        {
                Target.SendMessage("ApplyDammage",
Damage);
                Debug.Log ("Melakukan serangan");
                attackTime = Time.time + attackRepeatTime;
        }
}
```

Skrip di atas menjelaskan musuh akan melukai target dengan serangan sebesar variabel Damage, yaitu 40 poin.

5. Selanjutnya, buka skrip **EnemyLogic** dari objek **Enemy**.
6. Hapus bagian fungsi Update.
7. Selanjutnya, ubah bagian ApplyDamage menjadi seperti berikut.

```
function ApplyDammage (Damage : int)
{
        Health -= Damage;
        if (Health < 0)
        {
                Dead ();
        }
}
```

Skrip ini telah menambahkan kegunaan dari fungsi Update yang telah dihapus sebelumnya. Skrip ini menjelaskan pengurangan nyawa dari objek dan setelah nyawa menjadi 0 objek akan menjadi fungsi Dead.

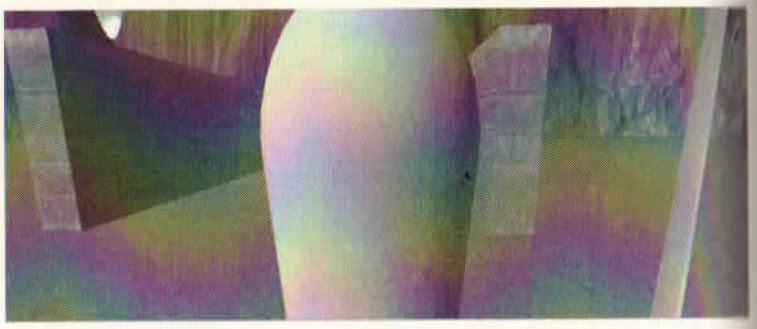
8. Salin skrip lengkap dari **EnemyLogic**.

```
#pragma strict

var Health = 100;

function ApplyDammage (Damage : int)
{
        Health -= Damage;
        if (Health < 0)
        {
                Dead();
        }
}

function Dead ()
{
        Destroy (gameObject);
}
```

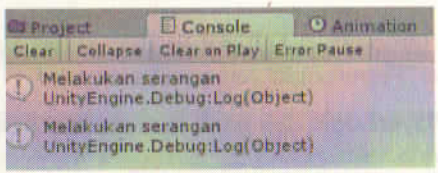
9. Selanjutnya, tambahkan komponen **Javascript** pada objek **Player** dengan nama "PlayerLogic" seperti Gambar 4.41.
10. Buka skrip **PlayerLogic** dan ganti semua skrip default dengan skrip yang telah Anda salin.

Gambar 4.41 Menambahkan skrip PlayerLogic

11. Klik tombol **Play** untuk menguji game.
12. Jika Anda mendekati musuh, musuh akan menyerang dan mengeluarkan pesan serangan.

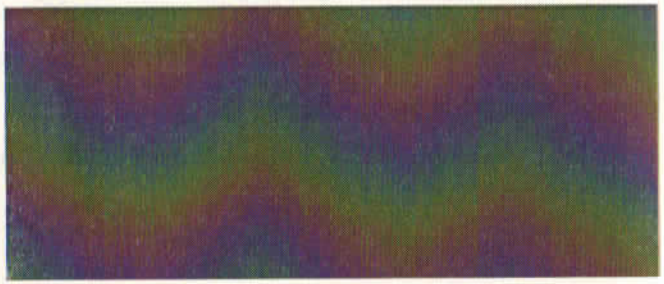


Gambar 4.42 Musuh menyerang karakter

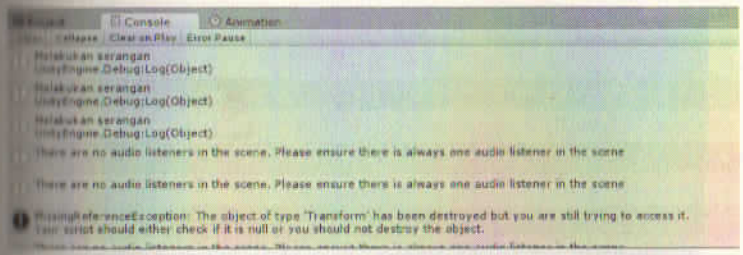


Gambar 4.43 Musuh menyerang karakter sebanyak dua kali

13. Setelah serangan ketiga, objek Player akan hilang dari jendela **Game**. Hal ini karena musuh telah melakukan serangan sebanyak 105 poin yang berarti melebihi nyawa objek Player yang hanya 100.
14. Anda juga akan mendapatkan pesan lain karena menghilangnya objek Player.



Gambar 4.44 Karakter menghilang dari jendela Game



Gambar 4.45 Setelah 3 serangan objek Player menghilang

15. Buka kembali skrip **PlayerLogic**.
16. Ubah fungsi Dead menjadi seperti berikut.

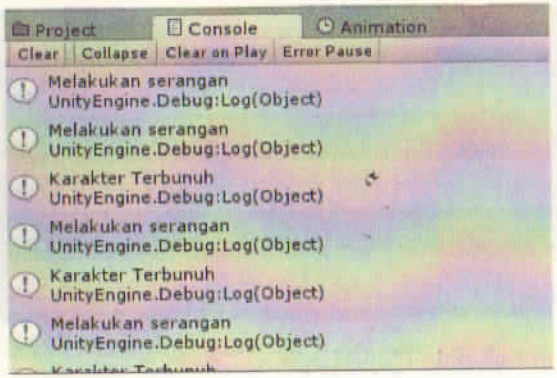
```
function Dead ()
{
        Debug.Log ("Karakter Terbunuh");
}
```

Skrip ini mengganti penghancuran objek Player saat mati menjadi hanya pesan bahwa "Karakter Terbunuh".

17. Skrip lengkap dari PlayerLogic sebagai berikut.

```
#pragma strict

var Health = 100;

function ApplyDammage (Damage : int)
{
        Health -= Damage;
        if (Health < 0)
        {
                Dead ();
        }
}

function Dead ()
{
        Debug.Log ("Karakter Terbunuh");
}
```

18. Klik **Play** untuk menguji game.
19. Saat nyawa karakter menjadi kurang dari sama dengan 0, muncul pesan "Karakter Terbunuh".



Gambar 4.46 Pesan Karakter Terbunuh saat nyawa karakter menjadi 0

20. Meskipun pesan muncul, objek karakter masih terlihat di jendela Game karena kondisi mati karakter telah diubah.

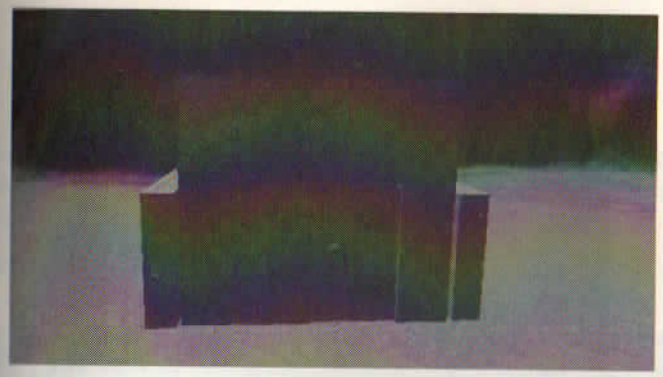
4.6 Membuat Pintu

Pada pembahasan ini Anda akan membuat pintu yang menutup tembok yang Anda buat sebelumnya. Untuk membuat pintu, ikuti langkah-langkah berikut:

1. Masukkan asset pembuat pintu dari file yang Anda download pada Bab 3.

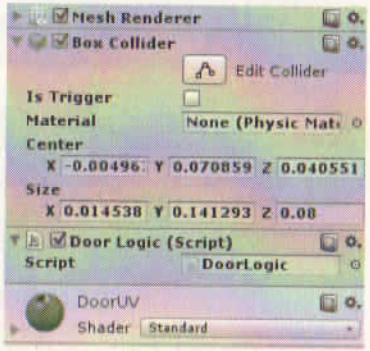
Gambar 4.47 Memasukkan asset pintu

2. Letakkan asset ke dalam jendela **Scene**.



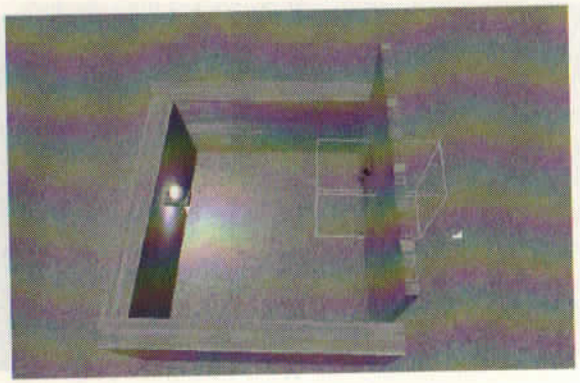
Gambar 4.48 Menambahkan asset pintu

3. Tambahkan collider pada objek pintu dan tembok. Pada objek **pCube2** tambahkan komponen **Mesh Collider**. Pada objek **polySurface1** tambahkan komponen **Box Collider**.



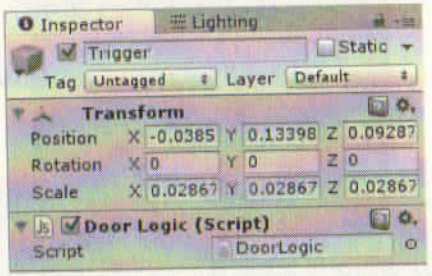
Gambar 4.49 Menambahkan Box Collider pada polySurface1

4. Buat objek baru dengan klik **GameObject > Create Empty**.
5. Tambahkan **Box Collider** dan buat supaya objek menjadi seukuran pintu dan berada di tengah pintu.
6. Buat objek memiliki lebar melebihi pintu.



Gambar 4.50 Membuat objek Trigger

7. Beri tanda pada **Is Trigger**.
8. Buat objek menjadi child dari **polySurface1**.
9. Beri nama objek dengan nama "Trigger".
10. Buat **Javascript** pada jendela **Inspector** dari objek **Trigger** deng nama "DoorLogic".



Gambar 4.51 Menambahkan DoorLogic pada objek Trigger

11. Ketikkan skrip berikut.

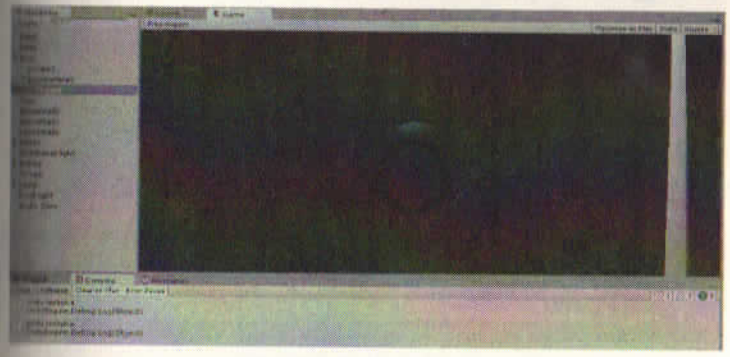
```
#pragma strict

function update ()
{

}

function OnTriggerEnter (theCollider : Collider)
{
```



```
        if (theCollider.tag == "Player")
        {
            Debug.Log("pintu terbuka");
        }
}
```

Pada fungsi OnTriggerEvent memberikan informasi jika objek dengan tag "Player" masuk ke dalam objek Trigger, akan muncul pesan pintu terbuka.

12. Klik tombol **Play** untuk menguji game.



Gambar 4.52 Pesan "pintu terbuka" saat karakter mendekati pintu

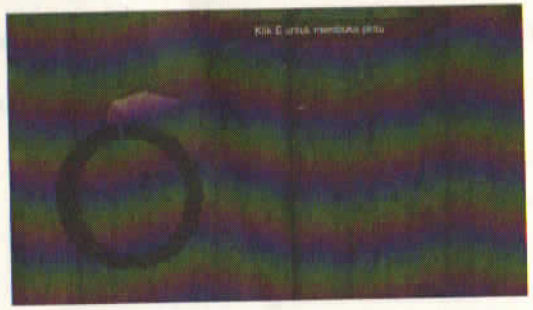
13. Kembali ke skrip DoorLogic dan ketikkan seperti berikut.

```
#pragma strict

private var drawGUI = false;

function update ()
{

}

function OnTriggerEnter (theCollider : Collider)
{
    if (theCollider.tag == "Player")
    {
        drawGUI = true;
    }
}

function OnTriggerExit (theCollider : Collider)
{
    if (theCollider.tag == "Player")
    {
        drawGUI = false;
    }
}
```

```
function OnGUI ()
{
        if (drawGUI == true)
        {
                GUI.Box (Rect (Screen.width*0.5-
51,15,250,25), "Klik E untuk membuka pintu");
        }
}
```

Berdasrkan skrip di atas, Anda menambahkan variabel drawGUI untuk menggambar pesan petunjuk membuka pintu. Pada fungsi OnTriggerEnter perubahan Debug.Log menjadi drawGUI bertujuan supaya pesan hanya muncul saat karakter berada di dalam objek trigger. Fungsi OnTriggerExit digunakan untuk menghapus pesan saat karakter meninggalkan objek Trigger. Fungsi OnGui menampilkan posisi pesan dan tampilan pesan petunjuk cara membuka pintu.

14. Klik tombol **Play** untuk menguji game.
15. Anda akan mendapatkan pesan untuk membuka pintu. Akan tetapi pintu belum dapat terbuka karena Anda belum membuat skrip membuka pintu.



Gambar 4.53 Pesan untuk membuka pintu muncul di layar

16. Jika Anda menjauhi pintu, pesan akan hilang kembali.
17. Selanjutnya, tambahkan variabel baru seperti berikut.

```
var theDoor = Transform;
var anim : Animation;
private var drawGUI = false;
private var doorIsClosed =true;
```


Variabel theDoor untuk mengatur pergerakan pintu. Variabel anim digunakan untuk memainkan animasi pergerakan pintu. Variabel doorIsClosed digunakan untuk membuat animasi pintu menutup.

18. Isi pada fungsi Update seperti berikut.

```
function update ()
{
        if (drawGUI == true &&
Input.GetKeyDown(KeyCode.E))
        {
                changeDoorState ();
        }
}
```

Fungsi ini membuat tombol **E** berfungsi untuk membuka pintu.

19. Selanjutnya, buat fungsi change DoorState seperti berikut.

```
function changeDoorState ()
{
        if (doorIsClosed==true)
        {
                anim.CrossFade("Open");
                doorIsClosed = false;
                yield WaitForSeconds (3);
                anim.CrossFade("Close");
                doorIsClosed = true;
        }
}
```

Fungsi ini memberikan perintah untuk menjalankan animasi membuka pintu. Anim.CrossFade digunakan pada Unity 5. Anda dapat menggunakan theDoor.animation.CrossFade pada Unity versi sebelumnya. Yield digunakan untuk meberi jeda antara animasi membuka dan menutup pintu.

20. Tampilan lengkap skrip membuka pintu seperti berikut.

```
#pragma strict
var theDoor : Transform;
private var drawGUI = false;
private var doorIsClosed =true;

function update ()
{
        if (drawGUI == true &&
Input.GetKeyDown(KeyCode.E))
        {
                changeDoorState();
        }
}

function OnTriggerEnter (theCollider : Collider)
{
```

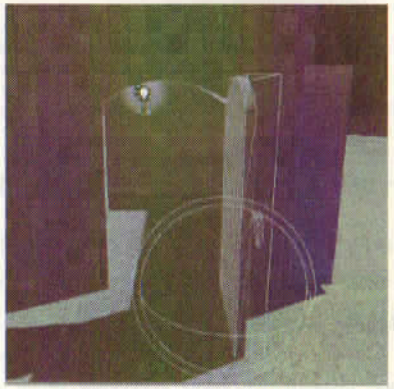
```
        if (theCollider.tag == "Player")
        {
                drawGUI = true;
        }
}

function OnTriggerExit (theCollider : Collider)
{
        if (theCollider.tag == "Player")
        {
                drawGUI = false;
        }
}

function OnGUI ()
{
        if (drawGUI == true)
        {
                GUI.Box (Rect (Screen.width*0.5-
51,15,250,25), "Klik E untuk membuka pintu");
        }
}

function changeDoorState()
{
        if (doorIsClosed==true)
        {
                theDoor.animation.CrossFade("Open");
                doorIsClosed = false;
                yield WaitForSeconds (3);
                theDoor.animation.CrossFade("Close");
                doorIsClosed = true;
        }
}
```

21. Pisahkan objek **Trigger** dari objek **polySurface1** sehingga objek **Trigger** tidak menjadi child lagi.
22. Selanjutnya, Anda akan membuat animasi membuka pintu. Klik objek **polySurface1** dan buka jendela **Animation**. Buat animasi dengan nama "Open".
23. Tambahkan komponen **Rotation** pada jendela **Animation** seperti Gambar 4.54.
24. Ubah **Center** menjadi **Pivot** pada pilihan di sebelah tool shortcut.

Gambar 4.54 Mengganti Center menjadi Pivot

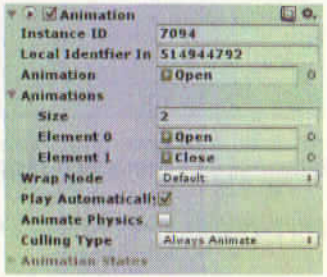
25. Buat pintu terbuka di titik 20 detik dan hapus key di titik 1 menit.
26. Klik **Ctrl+S** untuk menyimpan animasi.

27. Buat lagi animasi dengan nama "Close".
28. Gunakan posisi terakhir dari animasi Open sebagai titik 0.
29. Kemudian buat posisi pintu menutup di titik 20.



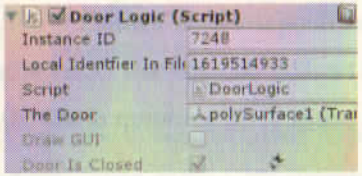
Gambar 4.55 Tool Rotation di kanan bawah objek polySurface1

30. Jadikan animasi Open dan Close sebagai Legacy jika keluar peringatan untuk membuat animasi menjadi Legacy.
31. Selanjutnya, tambahkan komponen **Animation** di objek **polySurface1**.
32. Daftarkan animasi seperti Gamabar 4.56.



Gambar 4.56 Mendaftarkan animasi

33. Pada objek **Trigger** daftarkan objek **polySurface1** dalam komponen **DoorLogic**.



Gambar 4.57 Mendaftarkan objek polySurface1

34. Klik tombol **Play** untuk menjalankan game.
35. Jika Anda melakukannya dengan benar, pintu akan terbuka saat Anda klik tombol **E** dan dalam beberapa detik akan tertutup kembali.

4.7 Membuat Menu Respawn

Menu ini merupakan kelanjutan saat karakter mati. Pada saat karakter mati, seharusnya permainan berhenti dan Anda dapat memilih untuk *respawn* atau memulai lagi dari awal. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Klik objek **Player** dan buat **Javascript** dengan nama "RespawnMenu".

Gambar 4.58 Menambahkan komponen skrip Respawn Menu

2. Buat variabel seperti berikut.

```
var lookAround01 : MouseLook;
var lookAround02 : MouseLook;
var charController : CharacterController;
static var playerIsDead = false;
```


Variabel lookAround digunakan untuk membatasi pergerakan karakter menggunakan mouse. Jika variabel ini tidak aktif, pengendalian karakter dengan mouse akan dibatasi.

3. Buat fungsi Start seperti berikut.

```
function Start ()
{
        lookAround01 =
gameObject.GetComponent(MouseLook);
        lookAround02 =
GameObject.Find("MainCamera").GetComponent(MouseLook);
        charController =
gameObject.GetComponent(CharacterController);
}
```

4. Kembali ke Unity. Pada jendela **Hierarchy** ubah nama objek Main Camera yang menjadi child dari objek **Player**. Ganti nama objek menjadi "MainCamera".
5. Kembali ke skrip **RespawnMenu**.
6. Tambahkan fungsi onGUI seperti berikut.

```
function OnGUI ()
{
        if (playerIsDead == true)
        {
                if (GUI.Button(Rect(Screen.width*0.5-50,
200-20, 100, 40), "Respawn"))
                {
                        RespawnPlayer();
                }

                if (GUI.Button(Rect(Screen.width*0.5-50,
240, 100, 40), "Menu"))
                {
                        Debug.Log("Return to Menu");
                }
        }
}
```

Skrip di atas akan menampilkan dua menu pada layar, yaitu Respawn di bagian atas dan Menu di bagian bawah.

7. Selanjutnya, buat fungsi Update di atas fungsi Start seperti berikut.

```
function Update ()
{
        if (playerIsDead == true)
        {
                lookAround01.enabled = false;
                lookAround02.enabled = false;
                charController.enabled = false;
        }
```

```
}
```

Fungsi ini akan mematikan semua gerakan mouse dan keyboard saat karakter dalam kondisi mati atau variabel characterIsDead aktif.

8. Tambahkan variabel baru seperti berikut.

```
var respawnTransform : Transform;
```

9. Buat fungsi characterisDead pada bagian paling bawah.

```
function RespawnPlayer ()
{
        transform.position = respawnTransform.position;
        transform.rotation = respawnTransform.rotation;
        gameObject.SendMessage("RespawnStats");
        lookAround01.enabled = true;
        lookAround02.enabled = true;
        charController.enabled = true;
        playerIsDead = false;
        Debug.Log("Player telah muncul kembali");
}
```

Fungsi di atas akan menampilkan pesan yang akan muncul saat karakter berhasil reespawn.

10. Skrip lengkap dari RespawnMenu seperti berikut.

```
#pragma strict

var lookAround01 : MouseLook;
var lookAround02 : MouseLook;
var charController : CharacterController;

var respawnTransform : Transform;

static var playerIsDead = false;

function Start ()
{
        lookAround01 =
gameObject.GetComponent(MouseLook);
        lookAround02 =
GameObject.Find("MainCamera").GetComponent(MouseLook);
        charController =
gameObject.GetComponent(CharacterController);
}

function Update ()
{
        if (playerIsDead == true)
        {
                lookAround01.enabled = false;
                lookAround02.enabled = false;
                charController.enabled = false;
```



```
        }
}

function OnGUI ()
{
        if (playerIsDead == true)
        {
                if (GUI.Button(Rect(Screen.width*0.5-50,
200-20, 100, 40), "Respawn"))
                {
                        RespawnPlayer();
                }

                if (GUI.Button(Rect(Screen.width*0.5-50,
240, 100, 40), "Menu"))
                {
                        Debug.Log("Return to Menu");
                }
        }
}

function RespawnPlayer ()
{
        transform.position = respawnTransform.position;
        transform.rotation = respawnTransform.rotation;
        gameObject.SendMessage("RespawnStats");
        lookAround01.enabled = true;
        lookAround02.enabled = true;
        charController.enabled = true;
        playerIsDead = false;
        Debug.Log("Player telah muncul kembali");
}
```

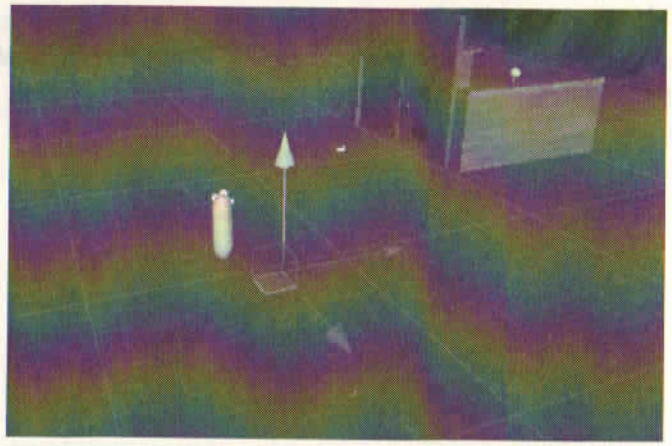
11. Selanjutnya, buka skrip **PlayerLogic** di objek **Player**. Ubah skrip seperti berikut.

```
#pragma strict

var MaxHealth = 100;
var Health : int;

function Start ()
{
        Health = MaxHealth;
}

function ApplyDammage (Damage : int)
{
        Health -= Damage;
        if (Health < 0)
        {
                Dead();
        }
}

function Dead ()
{
        RespawnMenu.playerIsDead = true;
        Debug.Log ("Karakter Terbunuh");
}
```

```
function RespawnStats ()
{
        Health = MaxHealth;
}
```

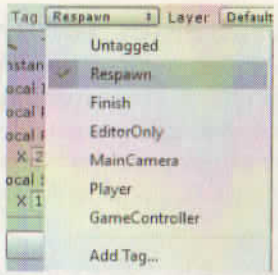
Bedasarkan skrip di atas, Anda akan memiliki dua variabel dari nyawa karakter, yaitu MaxHealth dan Health. MaxHealth adalah nyawa maksimum karakter. Health adalah nyawa karakter saat ini, di mana akan berkurang saat terkena serangan. Pada fungsi dead menjelaskan kondisi kemunculan respawn, yaitu saat karakter mati. Pada fungsi RespawnStats mengembalikan supaya nyawa karakter kembali penuh setelah respawn.

12. Selanjutnya, buat objek baru dengan klik **GameObject > Empty Object**.
13. Letakkan di dekat posisi karakter saat ini dan ganti nama menjadi "Respawn Position" seperti Gambar 4.59.
14. Anda dapat membuat tag khusus untuk objek ini dengan cara klik menu tag pada jendela **Inspector** dan pilih tag yang sesuai. Jika tidak ada tag yang sesuai, klik **Add Tag** dan tulis nama tag.



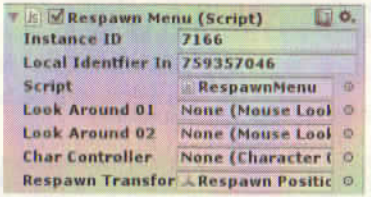
Gambar 4.59 Meletakkan posisi Respawn Position

15. Klik kembali objek dan pilih tag seperti Gambar 4.60.



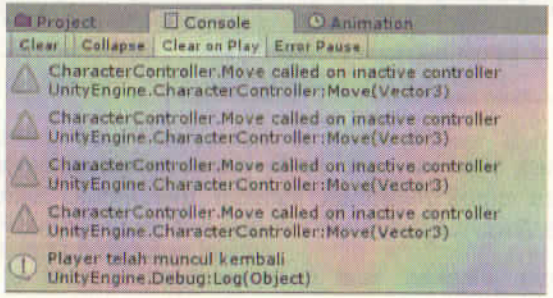
Gambar 4.60 Memilih tag

16. Daftarkan objek Respawn Location ke skrip Respawn Menu bagian Respawn Transform.



Gambar 4.61 Mendaftarkan game objek

17. Klik **Play** untuk menguji game. Jika Anda klik tombol **Respawn** saat karakter mati, Anda akan kembali muncul di posisi **Respawn Location** dengan nyawa penuh kembali.



Gambar 4.62 Pesan respawn karakter berhasil

4.8 Membuat Karakter Terluka Saat Jatuh

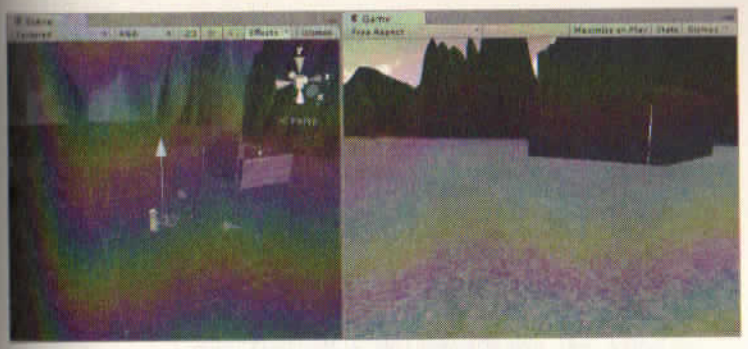
Pada saat karakter terjatuh, Anda dapat membuat karakter terluka. Pada pembahasan ini Anda akan membuat karakter terluka saat terjatuh. Luka ini menghasilkan pengurangan pada nyawa karakter tergantung dari seberapa tinggi karakter terjatuh. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Klik objek **Player** dan buat **Javascript** dengan nama "CrouchHeight".
2. Tuliskan skrip berikut.

```
#pragma strict

private var charController : CharacterController;
private var theTransform : Transform;
private var charHeight : float;

function Start ()
{
        theTransform = transform;
        charController =
GetComponent(CharacterController);
        charHeight = charController.height;
}

function Update ()
{
        var h = charHeight;

        if (Input.GetKey("c"))
        {
                h = charHeight*0.5;
        }

        var lastHeight = charController.height;
        charController.height =
Mathf.Lerp(charController.height, h, 5*Time.deltaTime);
        theTransform.position.y +=
(charController.height-lastHeight)/2;
}
```

   Bersarkan skrip di atas, Anda dapat menekan tombol C pada keyboard untuk membuat karakter menunduk. Saat menunduk karakter Anda menjadi lebih pendek dari seharusnya.

3. Jika Anda jalankan, saat Anda menekan tombol **C**, Anda akan melihat pandangan karakter menjadi lebih rendah.



Gambar 4.63 Posisi karakter menunduk

4. Perlu Anda perhatikan. Karakter dapat jatuh dari permukaan tanah jika posisi karakter terlalu rendah. Anda dapat membuktikannya dengan membuat objek seperti tembok dan membuat karakter berdiri di objek tersebut. Jika karakter merunduk terlalu lama, karakter akan jatuh dari objek.

Gambar 4.64 Karakter terjatuh dari objek

5. Selanjutnya, buat **Javascript** di objek **Player** dan beri nama "WalkerEnchanted".
6. Masukkan variabel berikut untuk mengatur pergerakan karakter saat terjatuh dari permukaan landai.

```
var walkSpeed = 6.0;
var runSpeed = 11.0;
var crouchSpeed = 3.0;
var limitDiagonalSpeed = true;
var enableRun = true;
var enableCrouch = true;
```

```
var jumpSpeed = 8.0;
var gravity = 20.0;
var enableFallingDamage = true;
var fallingDamageThreshold = 10.0;
var fallingDamageMultiplier = 2;
var slideWhenOverSlopeLimit = false;
var slideOnTaggedObjects = false;
var slideSpeed = 12.0;
var airControl = false;
var antiBumpFactor = .75;
var antiBunnyHopFactor = 1;
```

7. Selanjutnya, tambahkan private variabel seperti berikut.

```
private var moveDirection = Vector3.zero;
private var grounded = false;
private var controller : CharacterController;
private var myTransform : Transform;
private var speed : float;
private var hit : RaycastHit;
private var fallStartLevel : float;
private var falling = false;
private var slideLimit : float;
private var rayDistance : float;
private var contactPoint : Vector3;
private var playerControl = false;
private var jumpTimer : int;
private var charHeight : float;
```

8. Buat fungsi Start seperti berikut.

```
function Start ()
{
        controller = GetComponent(CharacterController);
        myTransform = transform;
        speed = walkSpeed;
        rayDistance = controller.height * .5 +
controller.radius;
        slideLimit = controller.slopeLimit - .1;
        jumpTimer = antiBunnyHopFactor;
        oldPos = transform.position;
}
```

9. Buat fungsi FixedUpdate seperti berikut.

```
function FixedUpdate()
{
        var inputX = Input.GetAxis("Horizontal");
        var inputY = Input.GetAxis("Vertical");
        var inputModifyFactor = (inputX != 0.0 && inputY
!= 0.0 && limitDiagonalSpeed)? .7071 : 1.0;

        if (grounded)
        {
                var sliding = false;
                if (Physics.Raycast(myTransform.position,
-Vector3.up, hit, rayDistance))
```



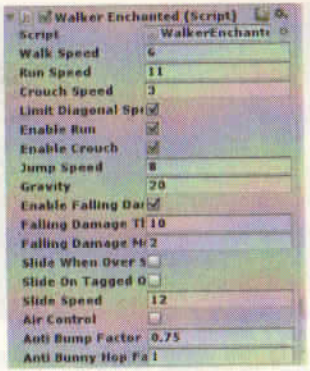
```
                {
                        if (Vector3.Angle(hit.normal,
Vector3.up) > slideLimit)
                                sliding = true;
                }
                else
                {
                        Physics.Raycast(contactPoint +
Vector3.up, -Vector3.up, hit);
                        if (Vector3.Angle(hit.normal,
Vector3.up) > slideLimit)
                                sliding = true;
                }

                if (falling)
                {
                        falling = false;
                        if (myTransform.position.y <
fallStartLevel - fallingDamageThreshold &&
enableFallingDamage == true)
                                ApplyFallingDamage
(fallStartLevel - myTransform.position.y);
                }

                if (Input.GetKey(KeyCode.LeftShift) &&
enableRun == true)
                {
                        speed = runSpeed;
                }
                else

                if (Input.GetKey("c") && enableCrouch ==
true)
                {
                        speed = crouchSpeed;
                }

                else
                {
                        speed = walkSpeed;
                }

                if ( (sliding && slideWhenOverSlopeLimit)
|| (slideOnTaggedObjects && hit.collider.tag == "Slide")
) {
                        var hitNormal = hit.normal;
                        moveDirection =
Vector3(hitNormal.x, -hitNormal.y, hitNormal.z);
                        Vector3.OrthoNormalize (hitNormal,
moveDirection);
                        moveDirection *= slideSpeed;
                        playerControl = false;
                }

                else
                {
                        moveDirection = Vector3(inputX *
inputModifyFactor, -antiBumpFactor, inputY *
inputModifyFactor);
                        moveDirection =
myTransform.TransformDirection(moveDirection) * speed;
                        playerControl = true;
```

```
                }

                if (!Input.GetButton("Jump"))
                        jumpTimer++;
                else if (jumpTimer >= antiBunnyHopFactor)
{
                        moveDirection.y = jumpSpeed;
                        jumpTimer = 0;
                }
        }
        else
        {
                if (!falling)
                {
                        falling = true;
                        fallStartLevel =
myTransform.position.y;
                }

                if (airControl && playerControl)
                {
                        moveDirection.x = inputX * speed *
inputModifyFactor;
                        moveDirection.z = inputY * speed *
inputModifyFactor;
                        moveDirection =
myTransform.TransformDirection(moveDirection);
                }
        }

        moveDirection.y -= gravity * Time.deltaTime;
        grounded = (controller.Move(moveDirection *
Time.deltaTime) & CollisionFlags.Below) != 0;
}
```

10. Selanjutnya, tulis skrip berikut.

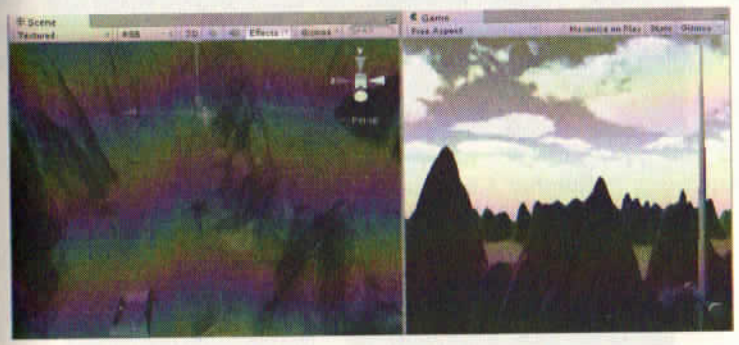
```
function Update ()
{
}

function OnControllerColliderHit (hit :
ControllerColliderHit)
{
        contactPoint = hit.point;
}

function ApplyFallingDamage (fallDistance : float)
{
        gameObject.SendMessage("ApplyDammage",
fallDistance*fallingDamageMultiplier);
        Debug.Log ("Anda terjatuh" + fallDistance + "
units!");
}
```

11. Jika Anda lihat komponen Walker Enchanted, Anda akan melihat banyak variabel yang mempengaruhi keadaan saat karakter terjatuh.



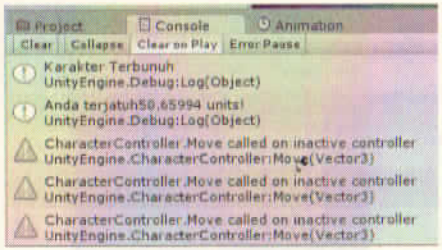
Gambar 4.65 Variabel yang mempengaruhi kejatuhan karakter

12. Jika Anda letakkan karakter pada daerah yang tinggi sehingga karakter terjatuh, maka karakter akan mendapat luka sesuai dengan ketentuan pada fungsi ApplyFallingDammage.



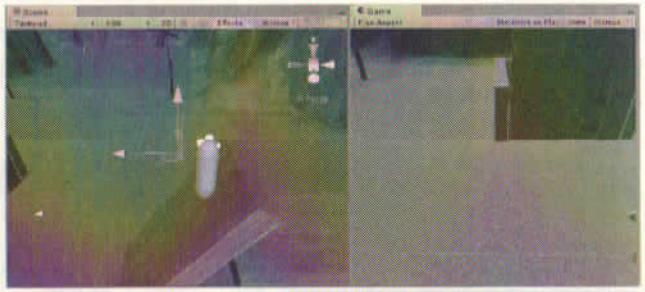
Gambar 4.66 Karakter berada di tempat tinggi

13. Jika Anda meletakkan karakter terlalu tinggi, nyawa karakter akan habis.



Gambar 4.67 Pesan karakter terbunuh karena terjatuh

14. Saat meluncur dari lantai yang landai, karakter akan bergerak lebih cepat.

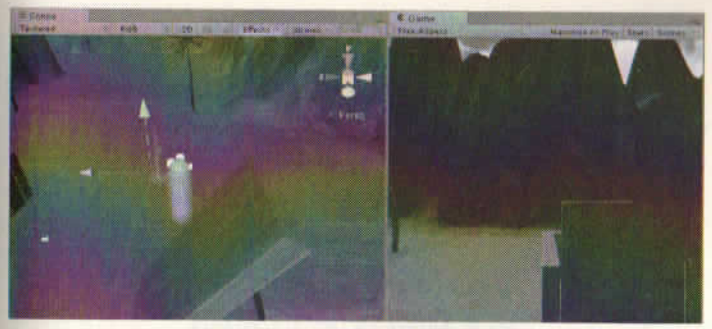


Gambar 4.68 Karakter sedang meluncur

15. Jika lantai terlalu landai, karakter akan tergelincir.

Gambar 4.69 Karkater tergelincir saat menaiki permukaan lantai landai

16. Karakter juga dapat meloncat saat meluncur.

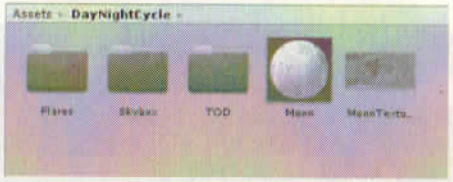


Gambar 4.70 Karakter meloncat saat meluncur

4.9 Membuat Perubahan Siang Malam

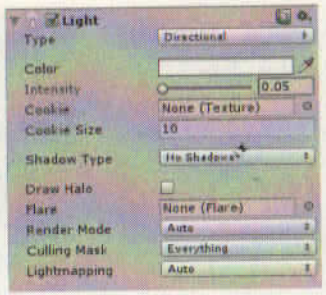
Pada pembahasan ini Anda akan membuat dalam dunia game menjadi waktu siang dan malam. Untuk membuat sistem ini, Anda akan langsung menggunakan paket Unity yang siap digunakan. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Tambahkan paket Unity yang berisi folder siang dan malam dari file yang Anda download pada Bab 3.
2. Jika Anda berhasil menambahkan paket, Anda akan mendapatkan folder baru yang berisi bahan untuk membuat kondisi siang dan malam seperti Gambar 4.71.
3. Pada folder tersebut Anda akan melihat gambar dan material untuk membuat bulan dan matahari.



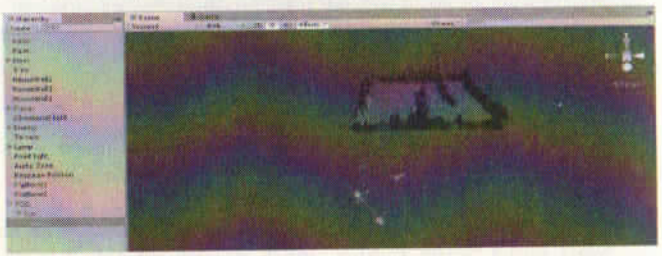
Gambar 4.71 Paket untuk membuat siang dan malam

4. Klik objek **Directional Light** dan buat **Intensity** menjadi "0.05".



Gambar 4.72 Mengurangi intensitas cahaya

5. Saat ini kondisi peta menjadi lebih gelap.
6. Tambahkan prefab TOD ke dalam jendela **Hierarchy**.
7. Anda akan melihat objek berada di peta.

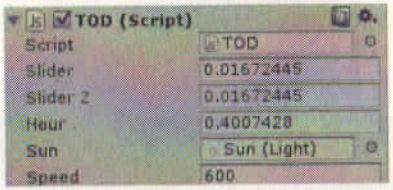


Gambar 4.73 Komponen ditambahkan ke dalam jendela Hierarchy

8. Jika Anda klik tombol **Play** Anda akan melihat pergantian waktu dari siang dan malam.

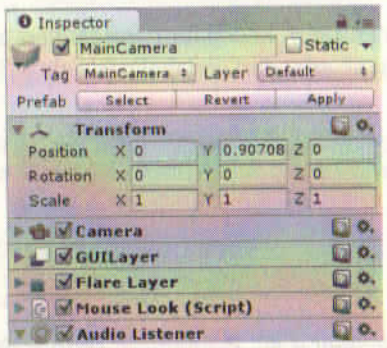
Gambar 4.74 Tampilan siang

Gambar 4.75 Tampilan malam



Gambar 4.76 Petunjuk pergerakan waktu siang dan malam

9. Jika Anda lihat pada Gambar 4.75, yaitu waktu malam, kondisi langit masih terlihat terang meskipun di bawah terlihat gelap.
10. Untuk mengatasi permasalahan pada awan, hapus komponen **Skybox** yang ada di **MainCamera** dari objek **Player**.



Gambar 4.77 Komponen Skybox telah dihapus

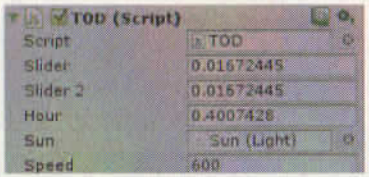
11. Klik **Play** dan Anda akan melihat langit menjadi gelap saat malam.

Gambar 4.78 Tampilan malam dengan langit gelap

12. Selanjutnya, ubah warna **Background** dari komponen **MainCamera** objek **Player** untuk mengubah warna langit. Gunakan warna biru langit.

Gambar 4.79 Mengubah warna langit pada MainCamera

13. Anda dapat mengatur kecepatan pergantian siang dana malam dengan mengganti nilai dari variabel pada skrip **TOD**.



Gambar 4.80 Pengaturan skrip TOD

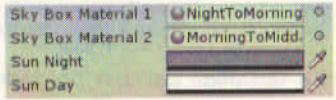
14. Berdasarkan Gambar 4.80 Anda dapat mengatur kecepatan siang dan malam dengan mengubah nilai **Speed**. Anda juga dapat

melihat jam dalam game dengan melihat nilai pada **Hour**. **Sun** merupakan bagian yang mendaftarkan objek matahari. **Slider** untuk melihat pergerakan objek.

15. Anda juga dapat mengatur warna pada bagian bawah dari pengaturan **TOD**.

Gambar 4.81 Pengaturan warna langit saat pergantian waktu

16. Pada bagian paling bawah Anda ada Skybox yang digunakan untuk menampilkan waktu siang dan malam.



Gambar 4.82 Pengaturan Skybox

17. Selanjutnya, akan dibahas mengenai bagian-bagian dari skrip. Pada bagian variabel, Anda akan melihat variabel-variabel berikut.

```
var slider : float;
var slider2 : float;
var Hour : float;
private var Tod: float;

var sun: Light;
var speed = 50;

var NightFogColor : Color;
var DuskFogColor : Color;
var MorningFogColor : Color;
```

```
var MiddayFogColor : Color;

var NightAmbientLight : Color;
var DuskAmbientLight : Color;
var MorningAmbientLight : Color;
var MiddayAmbientLight : Color;

var NightTint : Color;
var DuskTint : Color;
var MorningTint : Color;
var MiddayTint : Color;

var SkyBoxMaterial1 : Material;
var SkyBoxMaterial2 : Material;
var SunNight : Color;
var SunDay : Color;
```

Variabel-variabel di atas menunjukkan pengaturan dari kondisi siang dan malam yang dibahas sebelumnya.

18. Pada bagian fungsi GUI, Anda akan melihat skrip berikut.

```
function OnGUI () {

if(slider >= 1.0)
{
        slider = 0;
}

slider= GUI.HorizontalSlider( Rect(20,20,200,30), slide
0,1.0);
Hour= slider*24;
Tod= slider2*24;
sun.transform.localEulerAngles = Vector3((slider*360)-9
0, 0);
slider = slider +Time.deltaTime/speed;
sun.color = Color.Lerp (SunNight, SunDay, slider*2);
```

Fungsi ini mengatur tampilan gambar pergantian siang dan malam. Bagian slider akan mengatur pergerakan matahari dan bulan. Hour dan TOD menjelaskan perhitungan waktu pergantian siang dan malam.

19. Selanjutnya, Anda akan melihat skrip berikut.

```
if(Tod<4)
{
//Tampilan Malam
RenderSettings.skybox=SkyBoxMaterial1;
RenderSettings.skybox.SetFloat("_Blend", 0);
SkyBoxMaterial1.SetColor ("_Tint", NightTint);
RenderSettings.ambientLight = NightAmbientLight;
RenderSettings.fogColor = NightFogColor;
}
if(Tod>4&&Tod<6)
{
RenderSettings.skybox=SkyBoxMaterial1;
```



```
RenderSettings.skybox.SetFloat("_Blend", 0);
RenderSettings.skybox.SetFloat("_Blend", (Tod/2)-2);
SkyBoxMaterial1.SetColor ("_Tint", Color.Lerp (NightTint,
DuskTint, (Tod/2)-2) );
RenderSettings.ambientLight = Color.Lerp
(NightAmbientLight, DuskAmbientLight, (Tod/2)-2);
RenderSettings.fogColor = Color.Lerp
(NightFogColor,DuskFogColor, (Tod/2)-2);
//Tampilan Senja

}
if(Tod>6&&Tod<8)
{
RenderSettings.skybox=SkyBoxMaterial2;
RenderSettings.skybox.SetFloat("_Blend", 0);
RenderSettings.skybox.SetFloat("_Blend", (Tod/2)-3);
SkyBoxMaterial2.SetColor ("_Tint", Color.Lerp
(DuskTint,MorningTint, (Tod/2)-3) );
RenderSettings.ambientLight = Color.Lerp
(DuskAmbientLight, MorningAmbientLight, (Tod/2)-3);
RenderSettings.fogColor = Color.Lerp
(DuskFogColor,MorningFogColor, (Tod/2)-3);
//Tampilan Pagi

}
if(Tod>8&&Tod<10)
{
RenderSettings.ambientLight = MiddayAmbientLight;
RenderSettings.skybox=SkyBoxMaterial2;
RenderSettings.skybox.SetFloat("_Blend", 1);
SkyBoxMaterial2.SetColor ("_Tint", Color.Lerp
(MorningTint,MiddayTint, (Tod/2)-4) );
RenderSettings.ambientLight = Color.Lerp
(MorningAmbientLight, MiddayAmbientLight, (Tod/2)-4);
RenderSettings.fogColor = Color.Lerp
(MorningFogColor,MiddayFogColor, (Tod/2)-4);

//Tampilan Tengah Hari

}
```

Berdasarkan skrip di atas, pada saat nilai TOD < 4 Anda akan melihat tampilan dunia game dalam kondisi malam hari. Pada saat TOD antara 4 dan 6, Anda akan melihat tampilan senjata. Pada saat TOD 6 dan 8, Anda akan melihat tampilan pagi hari. Pada saat TOD 8 dan 10 Anda akan melihat tampilan tengah hari.

4.10 Memperbaiki Desain Tampilan Peta

Pada pembahasan ini Anda akan melakukan perubahan pada tampilan peta. Anda akan menambahkan beberapa objek dan menambahkan air pada peta. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Untuk menghias peta, tambahkan asset dari file yang Anda download pada Bab 3.
2. Anda akan melihat folder aset untuk menghias peta.

Gambar 4.83 Folder asset

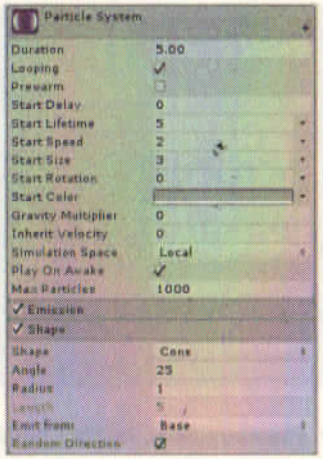
3. Klik folder **CampFire**.
4. Letakkan prefab **CampFire** ke dalam peta.
5. Kemudian berikan tekstur pada prefab sehingga objek api unggun menjadi berwarna seperti Gambar 4.84.
6. Selanjutnya, klik GameObject > Particle System.
7. Anda akan melihat partike kecil yang melayang. Tempatkan posisi partikel ke posisi api unggun.

Gambar 4.84 Tampilan objek api unggun

Gambar 4.85 Menempatkan partikel pada api unggun

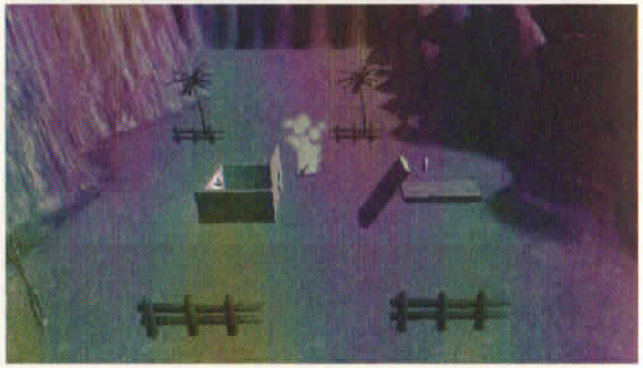
8. Ubah warna partikel menjadi jingga atau merah supaya warnanya terlihat seperti api.
9. Ubah **Start Speed** menjadi "2".
10. Ubah **Start Size** menjadi "3".
11. Klik pada bagian **Shape** dan pilih bentuk **Cone** supaya api unggun menyala ke atas. Ubah ketinggian cone sesuai dengan yang Anda inginkan.
12. Beri tanda pada **Random Direction** pada pengaturan **Shape**.

Gambar 4.86 Membuat api



Gambar 4.87 Pengaturan particle system

13. Klik folder **Fence**.
14. Masukkan prefab **Fance** ke dalam jendela **Scene**. Jika terlalu kecil gunakan **Scale Tool** untuk memperbesar objek.
15. Jika Anda ingin memperbanyak pagar, klik kanan pada objek Fance dan pilih **Duplicate**.
16. Beri tekstur pada setiap pagar.
17. Tambahkan **Mesh Collider** pada setiap **Fance**.

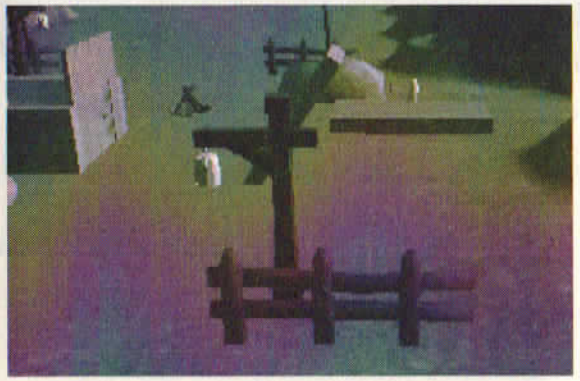


Gambar 4.88 Menempatkan pagar pada peta

18. Klik folder **Stone** dan tambahkan batu pada peta.
19. Masukkan prefab **Stone** pada jendela **Scene**.
20. Tambahkan tekstur pada objek batu.

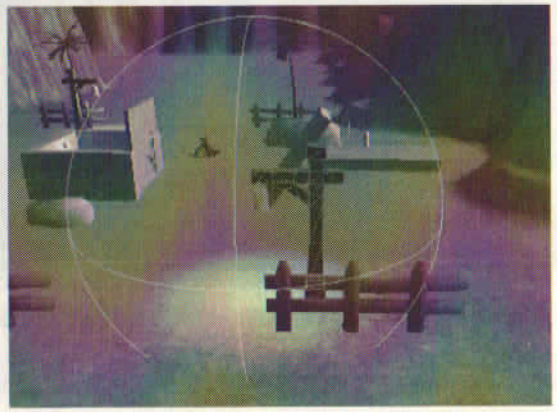
Gambar 4.89 Menambahkan objek batu pada peta

21. Klik folder **StreetLamp** dan tambahkan lampu pada peta.
22. Masukkan prefab **StreetLamp** pada jendela **Scene**.
23. Tambahkan tekstur pada lampu.



Gambar 4.90 Menambahkan objek lampu

24. Duplikasikan objek jika Anda ingin memberikan lebih dari satu lampu.
25. Klik pada bagian lampu. Nama default objek lampu adalah **Cube_003**.
26. Klik kanan dan pilih **Light > Spotlight**.
27. Pada bagian **Range** isi dengan "10".
28. Pada bagian **Intensity** isi dengan"4".



Gambar 4.91 Meletakkan objek lampu pada peta

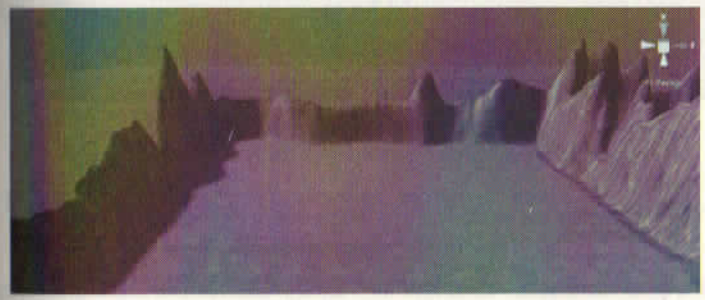
29. Selanjutnya, Anda akan menambahkan air pada peta. Klik kanan pada jendela **Project** dan klik **Import Asset > Water (Basic)**.
30. Klik **Import** pada kotak dialog yang muncul.
31. Anda akan melihat folder untuk membuat air.

Gambar 4.92 Asset untuk membuat air

32. Drag prefab Daylight Simple Water ke jendela Scene.
33. Buat supaya objek air berada di atas tanah sehingga air terlihat di peta.

Gambar 4.93 Menambahkan objek air

34. Tambahkan tekstur pada objek air dari folder **Material**.



Gambar 4.94 Menambahkan tekstur pada air

35. Tinggikan daerah sekitar air dengan klik **Terrain** dan gunakan tool **Paint Height**.
36. Haluskan permukaan tanah dengan **Smooth Height**.

Gambar 4.95 Meninggikan daerah sekitar air

37. Buka kembali skrip **TOD** yang ada di objek **TOD**.
38. Tambahkan variabel seperti berikut.

```
var Water : GameObject;
var IncludeWater = false;
var WaterNight : Color;
var WaterDay : Color;
```

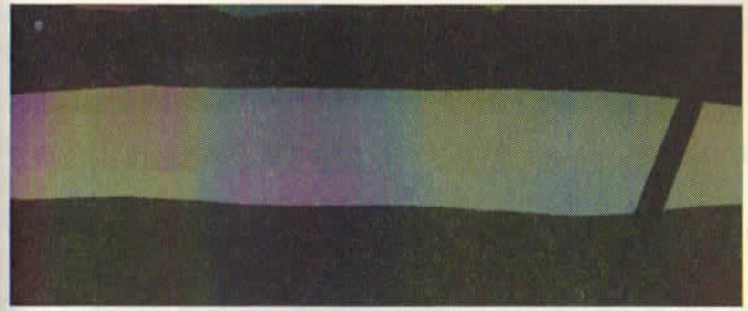
Variabel di atas untuk mendaftarkan objek air di skrip TOD dan juga mengatur pengaruh air pada saat siang dan malam.

39. Selanjutnya, tambahkan skrip berikut.

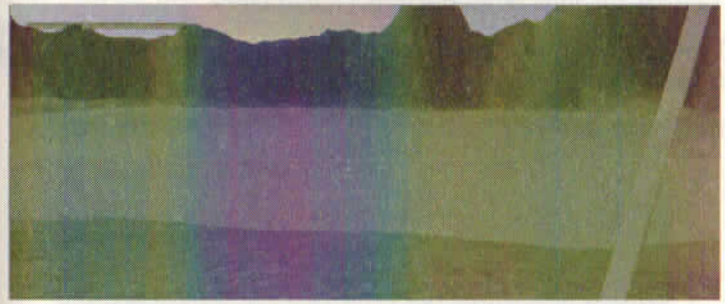
```
if (IncludeWater == true)
{
        Water.renderer.material.SetColor("_horizonColor",
Color.Lerp (WaterNight, WaterDay, slider2*2-0.2));
}
```

Skrip di atas akan memberikan efek pada air saat kondisi siang dan malam.

40. Klik tombol **Play** untuk menguji game.

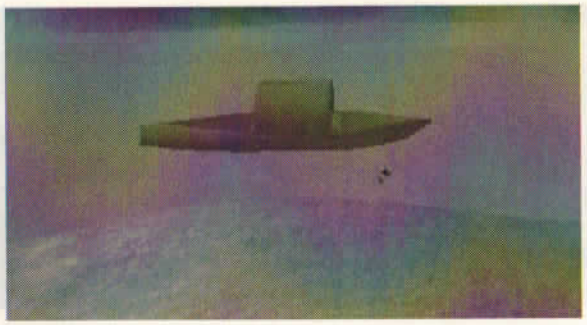


Gambar 4.96 Tampilan air pada malam hari



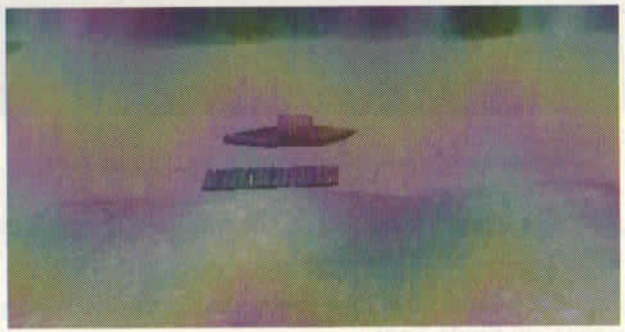
Gambar 4.97 Tampilan air pada saat siang hari

41. Anda dapat menambahkan perahu di objek air. Gunakan objek dari folder **start area** > **Chinese Boat**.
42. Drag prefab **Chinese Boat** ke jendela **Scene** dan perbesar menggunakan **Scale Tool**.
43. Berikan tekstur pada objek.



Gambar 4.98 Menambahkan objek perahu

44. Tambahkan juga objek dock dari folder **Dock**.
45. Drag prefab **Dock** je jendela **Scene** dan perbesar menggunakan **Scale Tool**.
46. Berikan tekstur pada objek.



Gambar 4.99 Tampilan air lengkap dengan perahu dan dermaga

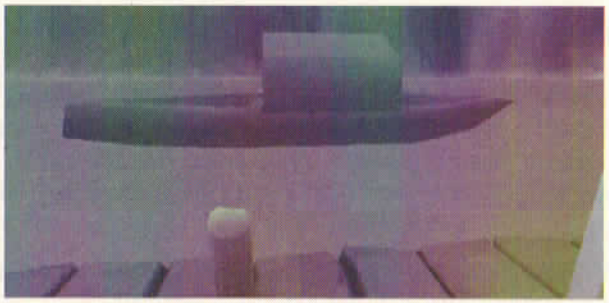
4.11 Perbaikan Objek pada Peta

Jika Anda uji game, Anda akan menemukan benda yang baru saja ditambahkan dapat ditembus oleh karakter. Untuk mengatasinya, Anda perlu menambahkan komponen collider pada bagian benda yang seharusnya tidak dapat dilewati oleh karakter. Untuk lebih jelas, ikuti langkah-langkah berikut:

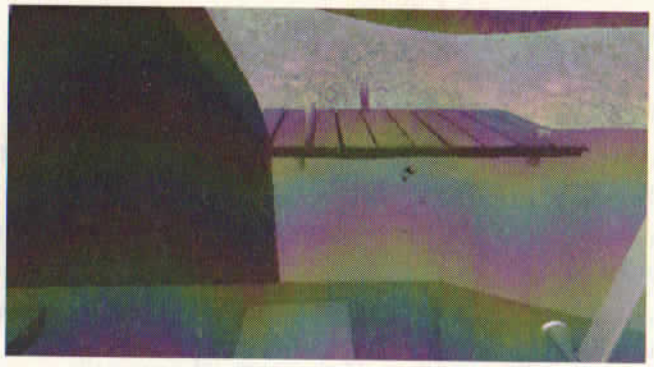
- Pada api unggun, tambahkan **Mesh Collider** pada bagian tempat kayu dan kayunya.
- Pada objek batu, tambahkan **Mesh Collider** pada objek batu.
- Pada lampu jalan, tambahkan **Mesh Collider** pada bagian tiang lampu.
- Pada dermaga, tambahkan **Mesh Collider** pada setiap child dari objek **Dock**.
- Pada perahu, tambahkan **Mesh Collider** pada setiap child dari objek **Chinese Boat**.

Jika Anda lakukan hal-hal di atas, objek-objek tersebut tidak dapat ditembus oleh karakter. Untuk beberapa objek seperti dermaga dan batu justru malah dapat dijadikan pijakan karena objek dapat dilompati oleh karakter.

Gambar 4.100 Karakter berdiri di atas batu



Gambar 4.101 Karakter berdiri di atas dermaga



Gambar 4.102 Karakter berdiri di atas perahu

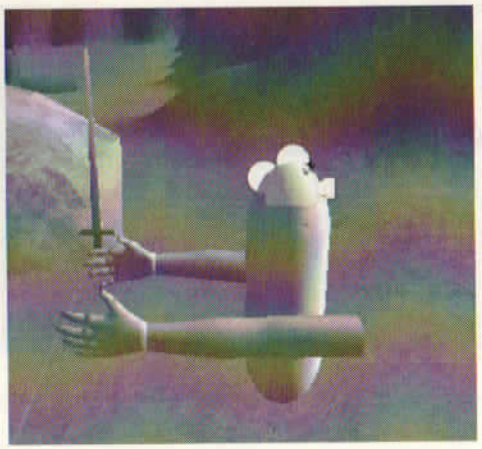
5 FINISHING

Sampai saat ini Anda telah mendapatkan game yang dapat dijalankan. Akan tetapi, Anda dapat memberikan sentuhan akhir supaya game menjadi lebih menarik. Ada bab ini Anda akan dijelaskan beberapa sentuhan akhir seperti menambahkan tangan, memperbanyak pilihan senjata, dan sebagainya. Selain itu, Anda juga akan dijelaskan cara supaya game dapat dijalankan secara *stand alone*.

5.1 Menambahkan Tangan untuk Karakter

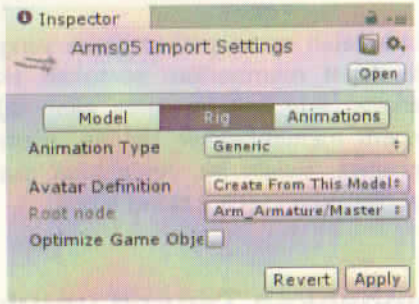
Saat ini senjata yang digunakan karakter seperti melayang. Pada pembahasan ini Anda akan menambahkan tangan untuk memegang senjata. Tidak hanya memegang senjata, tetapi tangan ini dapat digunakan utuk menyerang juga. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Tambahkan folder **Arms** dari file yang Anda download pada Bab 3.
2. Tambahkan file **Arms05.fbx** ke dalam jendela **Scene**.
3. Perbesar ukuran objek tangan dengan **Scale Tool**.



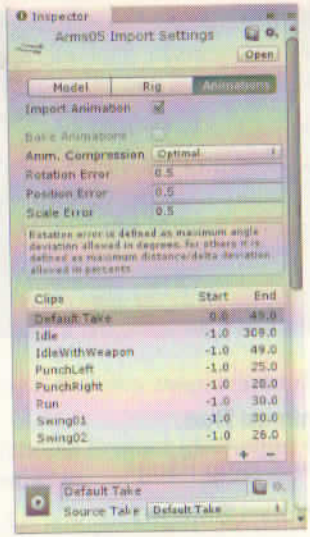
Gambar 5.1 Menambahkan objek tangan pada Scene

4. Klik gambar **Arms05.fbx** di jendela **Project**.
5. Pada jendela **Inspector** akan terlihat seperti Gambar 5.2.



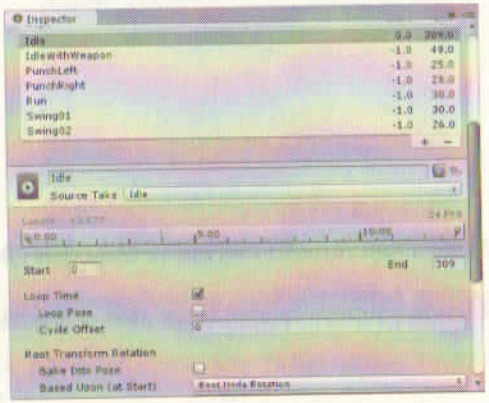
Gambar 5.2 Pengaturan Inspector

6. Pada tab Generic pilih Animation Type menjadi Generic.
7. Pada Root nod pilih Arm_Armature > Master.
8. Selanjutnya, klik **Apply**.
9. Klik tab **Animation**. Anda akan melihat semua animasi gerak tangan.



Gambar 5.3 Daftar animasi gerakan tangan

10. Hapus animasi **Default Take** dengan cara klik nama animasi dan klik tanda -.
11. Klik animasi **Idle** dan scroll untuk menunjukkan pengaturan animasi.
12. Beri tanda pada **Loop Time** dan klik **Clamp Range**.

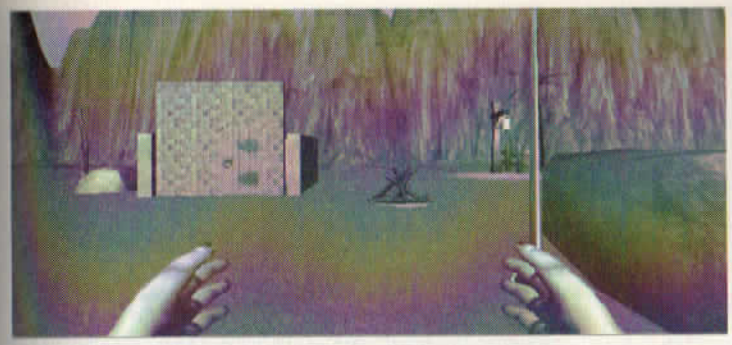


Gambar 5.4 Mengatur animasi Idle

13. Pilhan pada langkah 12 akan membuat animasi berjalan teru setelah waktu animasi selesai. Hal ini dapat dilihat dengan li tombol **Play** pada gambar animasi bagian bawah jendela **Inspector**
14. Selanjutnya, klik animasi **IdleWithWeapon**.
15. Klik Clamp Range.
16. Selanjutnya, klik animasi **PunchLeft**.
17. Klik Clamp Range.
18. Selanjutnya, klik animasi **PunchRight**.
19. Klik Clamp Range.
20. Selanjutnya, klik animasi **Run**.
21. Klik Loop Pause dan Clamp Range.
22. Selanjutnya, klik animasi **Swing01**.
23. Klik Clamp Range.
24. Selanjutnya, klik animasi **Swing02**.
25. Klik Clamp Range.
26. Klik **Apply** pada bagian bawah jendela **Inspector**.
27. Jadikan objek **Arms05** sebagai child dari **MainCamera** yang ada di objek **Player**.
28. Jadikan juga **Melee** menjadi child dari **MainCamera**.

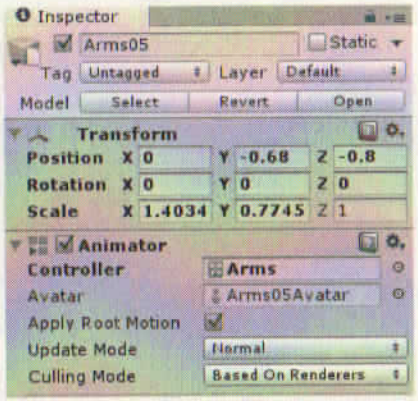
Gambar 5.5 Membuat Arms05 dan melee child dari MainCamera

29. Buat posisi tangan supaya terlihat tepat saat di jendela **Game**.



Gambar 5.6 Posisi tangan saat di game

30. Selanjutya, pada jendela **Project** klik kanan dan pilih **Create Object > Animation Controller**.
31. Beri nama controller dengan nama "Arms".
32. Drag controller **Arms** ke jendela **Inspector Arms05** bagian **Animator**.
33. Masukkan pada bagian **Controller**.

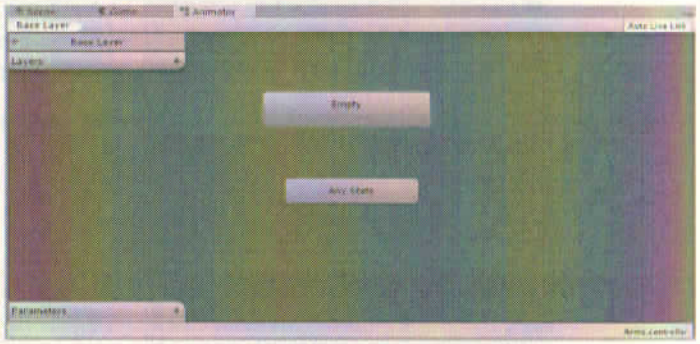


Gambar 5.7 Menambahkan controller untuk animasi tangan

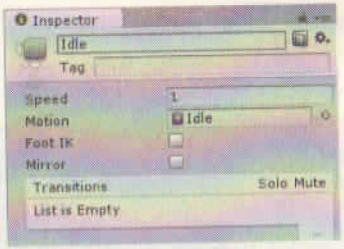
34. Klik ganda pada file controller **Arms** untuk menampilkan jendela **Animator**.

Gambar 5.8 Jendela Animator

35. Klik kanan pada jendela dan pilih **Create State** > **Empty** seperti Gambar 5.9.
36. Pada jendela **Inspector** ganti nama "Empty" dengan "Idle".
37. Tambahkan animasi **Idle** dari **Arms05.fbx**.
38. Drag animasi ke bagian **Motion** seperti Gambar 5.10.



Gambar 5.9 Menambahkan state baru

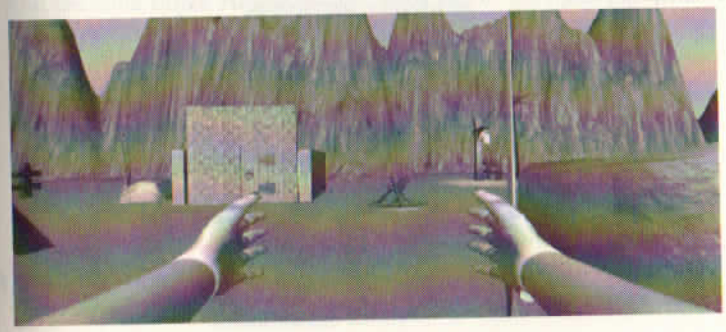


Gambar 5.10 Mengatur state

39. Jika Anda melakukannya dengan benar, saat game Anda jalankan, tangan akan bergerak sesuai dengan animasi Idle.

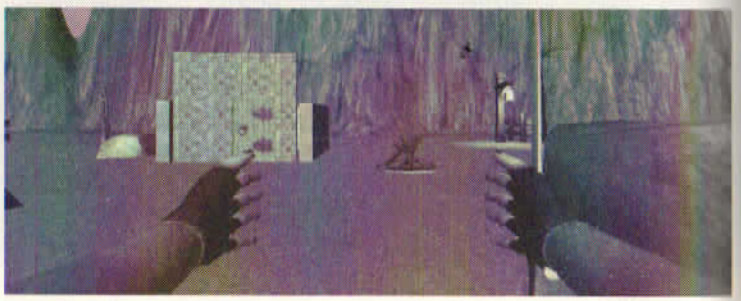
Gambar 5.11 Tangan bergerak menggunakan animasi Idle

40. Jika posisi tangan kurang pas di layar game, geser posisi tangan sehingga terlihat jelas pada layar game.



Gambar 5.12 Memperbaiki posisi tangan

41. Tambahkan tekstur pada tangan. Berikan warna tekstur pada **Arm_Mesh**.

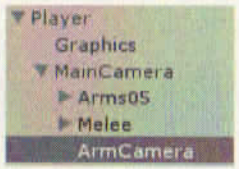


Gambar 5.13 Memberi tekstur pada tangan

5.1.1 Membuat Tangan Tidak Dapat Menembus Objek

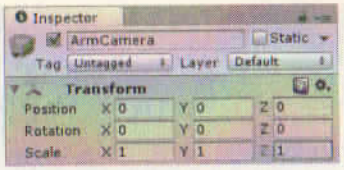
Jika Anda perhatikan, saat karakter menabrak objek, tangan akan masuk ke dalam objek tersebut. Pada pembahasan ini akan menjelaskan cara mencegah hal itu terjadi. Untuk lebih jelas, ikuti langkah-langkah berikut

1. Buat kamera baru dengan klik **Game Object > Camera**.
2. Beri nama kamera dengan nama "ArmCamera".
3. Jadikan kamera kedua menjadi child dari **MainCamera**.



Gambar 5.14 Membuat kamera baru

4. Isi bagian **Rotaion** dan **Position** dengan "0", sedangkan pada **Scale** isi dengan "1".

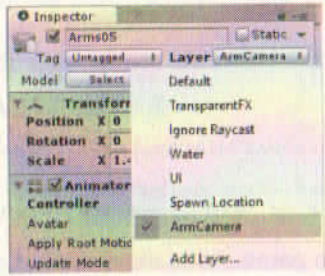


Gambar 5.15 Menempatkan kamera di tengah

5. Pada bagian **Camera**, buat **Clear Flag** menjadi **Depth Only**.
6. Klik objek **Arms05** dan klik **Add Layer** pada **Layer**.
7. Ketikkan nama layer baru dengan "ArmCamera".

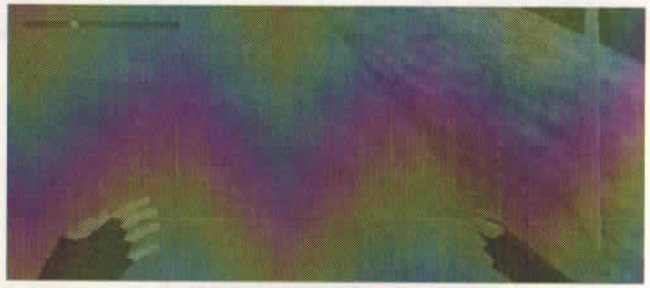
Gambar 5.16 Membuat layer baru

8. Klik kembali objek **Arms05** dan ganti **Layer** menjadi **ArmCamera**.



Gambar 5.17 Mengganti layer menjadi ArmCamera

9. Klik **ArmCamera** dan klik **Culling Mask** pada bagian Camera menjadi **Nothing**.
10. Selanjutnya, klik kembali **Culling Mask** dan klik**ArmCamera**.
11. Pada bagian **Depth** isi dengan "1".
12. Hapus komponen Flare Layer, GUILayer, dan Audio Listener dari ArmCamera.
13. Jika Anda jalankan game, Anda akan melihat sekarang tangan tidak lagi menembus objek.



Gambar 5.18 Tangan tidak menembus objek

14. Anda dapat mengubah layer dari senjata dengan **ArmCamera** sehingga senjata juga tidak dapat menembus objek lain.
15. Jika Anda perhatikan, pada bagian kiri layar terdapat garis yang menandakan waktu yang sedang berjalan. Anda dapat menghilangkannya dengan membuka skrip **TOD**.
16. Beri tanda // pada bagian slider= GUI.HorizontalSlider(Rect(20,20,200,30), slider, 0,1.0);.

```
//slider= GUI.HorizontalSlider( Rect(20,20,200,30),
slider, 0,1.0);
Hour= slider*24;
Tod= slider2*24;
sun.transform.localEulerAngles = Vector3((slider*360)-90,
0, 0);
slider = slider +Time.deltaTime/speed;
sun.color = Color.Lerp (SunNight, SunDay, slider*2);
```

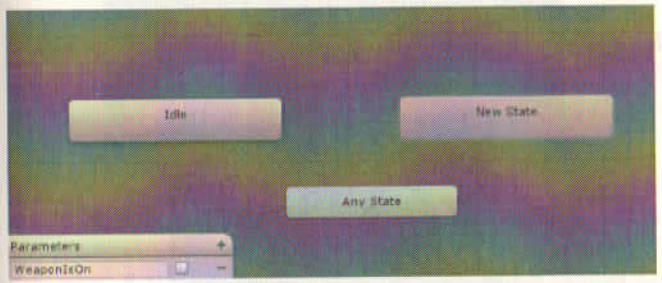
17. Jika Anda jalankan game, Anda akan melihat garis telah hilang.

Gambar 5.19 Garis pada kiri atas telah hilang

5.1.2 Menambahkan Animasi Gerakan Tangan

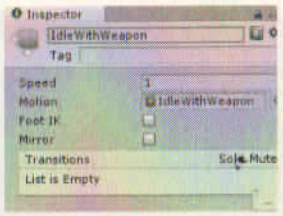
Pada pembahasan ini Anda akan menambah berbagai gerakan tangan. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Buka jendela **Animator** dan buat state baru dengan klik kanan pilih **Create State > Empty**.



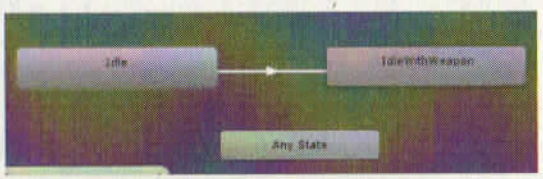
Gambar 5.20 Membuat state baru

2. Beri nama state dengan nama "IdleWithWeapon".
3. Pilih animasi IdleWithWeapon.fbx.



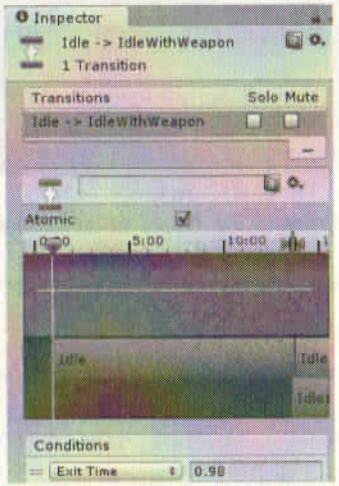
Gambar 5.21 Membuat state baru

4. Klik kanan pada state **Idle** dan pilih **Make Tansition**.
5. Arahkan panah pada state **IdleWithWeapon**.



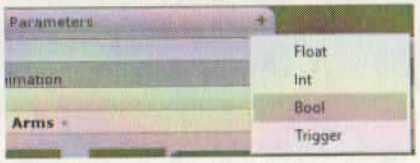
Gambar 5.22 Membuat transisi

6. Klik garis transisi untuk menampilkan pengaturan transisi.



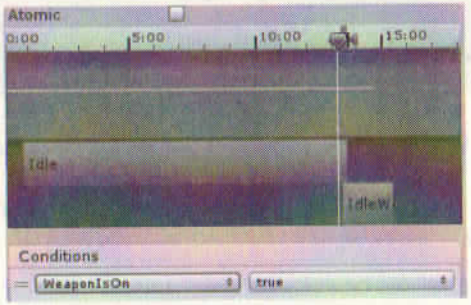
Gambar 5.23 Pengaturan garis transisi

7. Hilangkan tanda pada **Atomic**.
8. Klik **Parameters** pada jendela **Animator** dan pilih **Bool**.



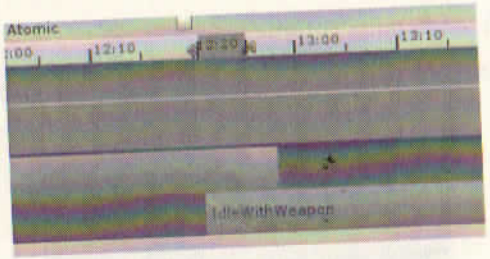
Gambar 5.24 Menambah parameter

9. Beri nama parameter dengan nama "WeaponIsOn".
10. Kembali pada pengaturan transisi. Pada bagian **Condition** pilih **WeaponIsOn** dan beri nilai **True**.



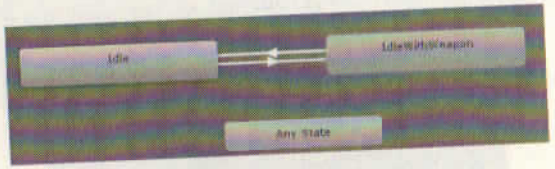
Gambar 5.25 Menambahkan parameter baru

11. Geser garis di bawah **Atomic** dengan cara scroll mouse ke depan. Anda akan melihat jarak antar titik menjadi lebih panjang.
12. Pada garis yang memblok biru, perlebar ke kiri supaya transisi pergerakan menjadi lebih halus.



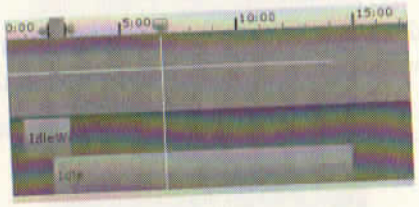
Gambar 5.26 Memperhalus transisi

13. Selanjutnya buat garis transisi kebalikan dengan cara klik kanan state **IdleWithWeapon > Make Transision**.
14. Arahkan garis ke state **Idle**.



Gambar 5.27 Membuat garis transisi baru

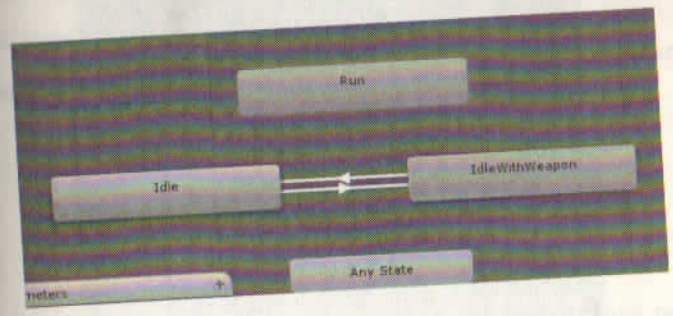
15. Klik garis transisi. Pada bagian **Condition** pilih **WeaponIsOn** dengan nilai **False**.
16. Perbaiki garis di bawah **Atomic** untuk memperhalus gerakan transisi.



Gambar 5.28 Mengatur pergantian transisi

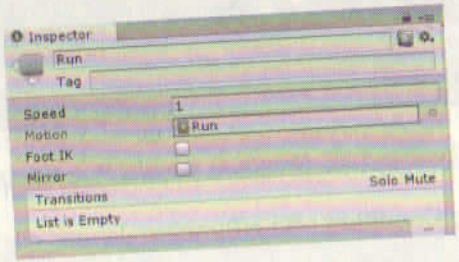
17. Klik **Parameters** pada jendela **Animator**.
18. Tambahkan parameter baru dengan klik **Bool** dan beri nama "IsRunning".

19. Klik kanan pada jendela **Animator** dan pilih **Create State > Empty**.



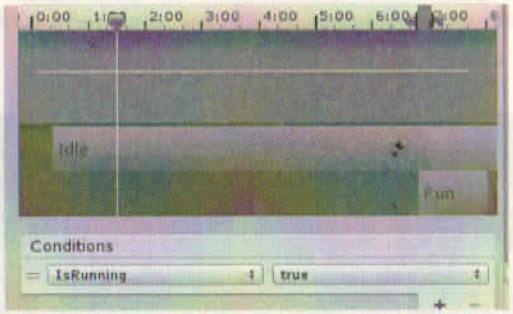
Gambar 5.29 Membuat state baru

20. Beri nama sate dengan nama "Run".
21. Tambahkan animasi **Run.fbx**.



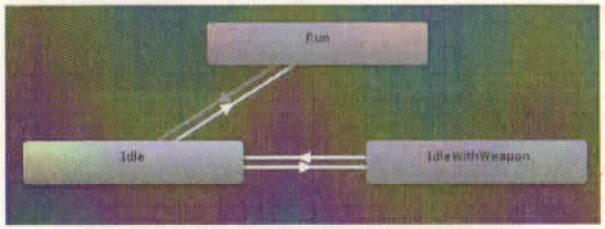
Gambar 5.30 Pengaturan state Run

22. Buat garis transisi dari **Idle** ke **Run**.
23. Klik garis tersebut untuk menampilkan pengaturan transisi.
24. Pada bagian **Condition** pilih **IsRunning** dengan nilai **True**.



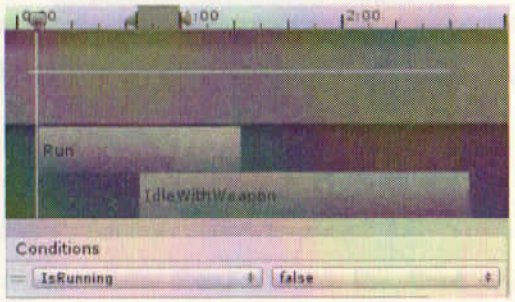
Gambar 5.31 Pengaturan transisi

25. Buat garis transisi dari **Run** ke **Idle**.



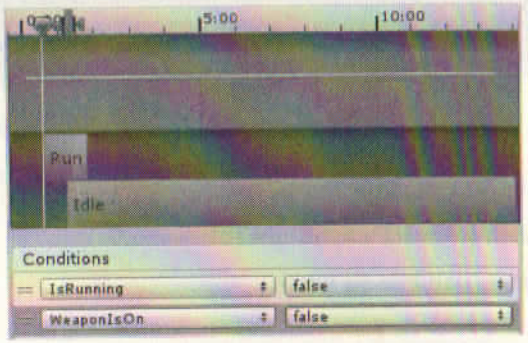
Gambar 5.32 Membuat transisi Run ke Idle

26. Pada bagian **Condition** pilih **IsRunning** dengan nilai **False**.
27. Buat garis transisi dari **Run** ke **IdleWithWeapon**.
28. Pada bagian **Condition** tambahkan **IsRunning** dan beri nilai **False**.



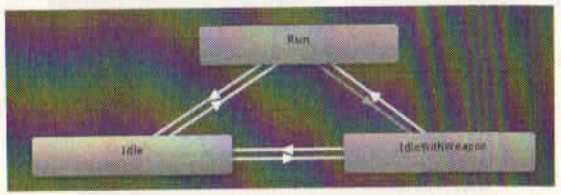
Gambar 5.33 Mengatur transisi gerakan Run ke IdleWithWeapon

29. Buat garis transisi dari **IdleWithWeapon** ke **Run**.
30. Pada bagian **Condition** tambahkan **IsRunning** dan beri nilai **True**.
31. Klik garis transisi dari **Run** ke **Idle**.
32. Pada **Condition** tambahkan **WeaponIsOn** dan beri nilai **False**.



Gambar 5.34 Mengatur transisi Run ke Idle

33. Klik garis transisi **Run** ke **IdleWithWeapon**.
34. Pada bagian Condition tambahkanWeaponIsOn dengan nilai True.
35. Hasil hubungan ketiga animasi akan seperti Gambar 5.35.

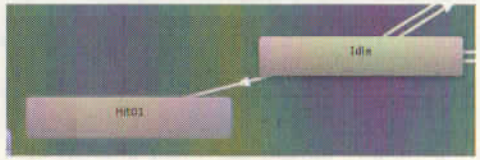


Gambar 5.35 Transisi antar animasi

36. Klik objek **Arms05** dan beri tanda pada **Apply Root**.
37. Klik tombol **Play** untuk menguji game.
38. Pada jendela **Animator** beri tanda pada **IsRunning** dan **WeaponIsOn** untuk melihat perubahan pergerakan tangan.

Gambar 5.36 Pergerakan tangan saat berlari dan membawa senjata

39. Buat state baru dan beri nama "Hit01".
40. Tambahkan animasi **PunchRight.fbx**.
41. Buat Transisi dari **Idle** ke **hit01**.

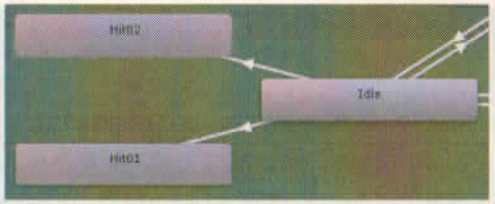


Gambar 5.37 Membuat transisi dari idle ke Hit01

42. Buat **Parameter** baru dengan klik **Bool** dan beri nama "Hit01".
43. Klik garis transisi **Idle** ke **Hit01**.
44. Pada bagian **Condition** tambahkan **Hit01** dengan nilai **True**.

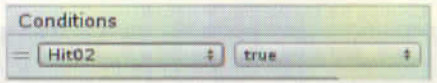
Gambar 5.38 Kondisi transisi Idle ke Hit01

45. Buat state baru dengan nama "Hit02".
46. Tambahkan animasi **PunchLeft.fbx**.
47. Buat transisi dari **Idle** ke **Hit02**.



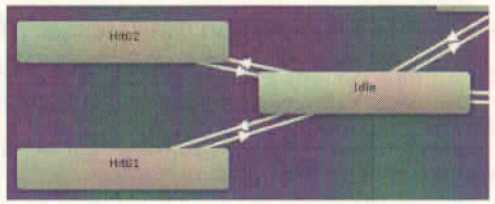
Gambar 5.39 Membuat transisi dari Idle ke Hit02

48. Buat **Parameter** baru dengan klik **Bool** dan beri nama "Hit02".
49. Klik garis transisi **Idle** ke **Hit02**.
50. Pada bagian **Condition** tambahkan **Hit02** dengan nilai **True**.



Gambar 5.40 Kondisi transisi dari Idle ke Hit02

51. Selanjutnya buat transisi dari **Hit01** ke **Idle** dan **Hit02** ke **Idle**. Gunakan **Condition** default.

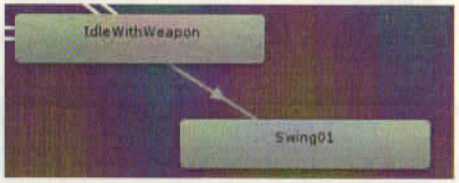


Gambar 5.41 Transisi Idle ke Hit01 dan Hit02

52. Klik tombol **Play** untuk menguji game. Jika Anda melakukannya dengan benar saat Anda menekan **Hit01** atau **Hit02**, setelah animasi dilakukan tangan akan kembali ke posisi **Idle**.

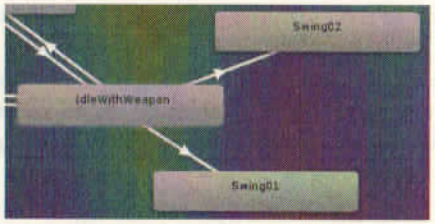
Gambar 5.42 Animasi pergerakan tangan saat memukul

53. Buat state baru dengan nama "Swing01".
54. Tambahkan animasi **Swing01.fbx**.
55. Buat transisi dari **Idle** ke **Swing01**.



Gambar 5.43 Membuat transisi IdleWithWeapon ke Swing01

56. Pada bagian **Condition** tambahkan **Hit01** dengan nilai **True**.
57. Buat state baru dengan nama "Swing02".
58. Tambahkan animasi **Swing02.fbx**.
59. Buat transisi dari **Idle** ke **Swing02**.



Gambar 5.44 Membuat transisi IdleWithWeapon ke Swing02

60. Pada bagian **Condition** tambahkan **Hit02** dengan nilai **True**.

61. Selanjutnya buat transisi dari **Swing01** ke **Idle** dan **Swing02** ke **Idle**. Gunakan **Condition** default.
62. Klik tombol **Play** untuk menguji game. Jika Anda melakukannya dengan benar saat Anda menekan **Swing01** atau **Swing02**, setelah animasi dilakukan tangan akan kembali ke posisi **IdleWithWeapon**.

Gambar 5.45 Gerakan Swing01

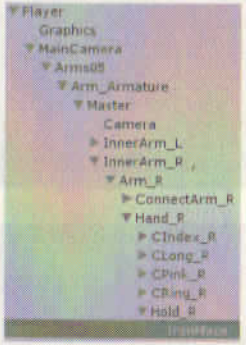
5.1.3 Menambahkan Senjata pada Tangan

Pada pembahasan ini Anda akan membuat senjata dapat bergerak sesuai dengan animasi tangan. Senjata yang akan digunakan merupakan senjata baru yang dapat Anda peroleh dari file yang Anda download dari Bab 3. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Tambahkan senjata ke dalam Unity. Pada contoh ini akan ditambahkan **IronMace.fbx**.
2. Hapus komponen **Animator** dari objek **IronMace**.

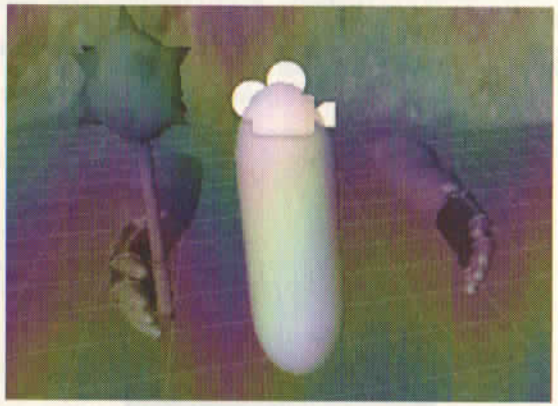
Gambar 5.46 Memberi tekstur pada IronMace

3. Klik **IronMace.fbx** di jendela **Project**. Pada tab **Animation** hilangkan tanda pada **Import Animation** dan pada tab **Rig** buat **Animation Type** menjadi **None**.
4. Beri tekstur pada objek **IronMace** seperti Gambar 5.46.
5. Selanjutnya, buat objek **IronMace** menjadi child dari tangan kanan seperti Gambar 5.47.



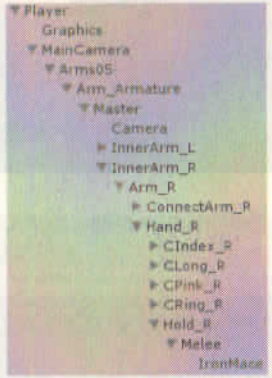
Gambar 5.47 IronMace menjadi child dari Hold_R

6. Semua senjata nantinya akan diletakkan pada Hold_R.
7. Buat posisi **IronMace** tepat di tangan kanan.



Gambar 5.48 Senjata diletakkan tepat di tangan kanan

8. Anda dapat memindahkan **Melee** menjadi child dari **Hold_R** dan **IronMace** menjadi child dari **Melee**. Akan tetapi non aktifkan semua skrip yang ada di **Melee** dan juga non aktifkan **Sword** dan **Mace** karena Anda tidak menggunakan senjata itu lagi.



Gambar 5.49 Susunan child senjata

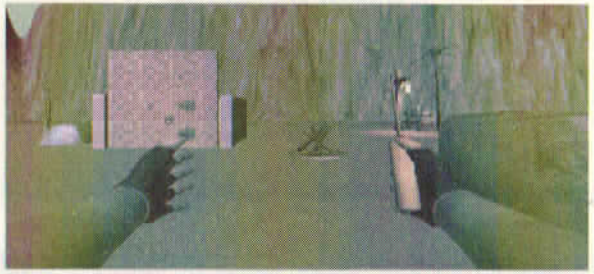
9. Buat posisi **Melee** dan **IronMace** tepat di tangan kanan.

Gambar 5.50 Posisi senjata tepat di tangan kanan saat tampilan game

10. Lakukan langkah yang sama dari penghilangan komponen **Animator**, pemberian tekstur dan peletakan tangan kanan untuk senjata lain seperti pisau dan pedang yang ada dalam paket file yang di-download.

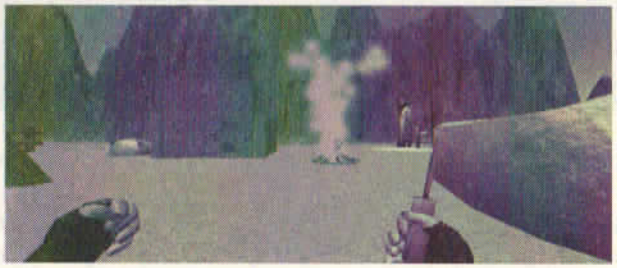
11. Proses peletakan senjata akan seperti Gambar 5.51.

Gambar 5.51 Meletakkan pisau pada tangan kanan



Gambar 5.52 Tampilan dalam game

12. Jika Anda coba menjalankan animasi seperti pada pembahasan menambahkan animasi gerakan tangan, maka pergerakan senjata akan sesuai dengan pergerakan tangan.



Gambar 5.53 Pergerakan senjata sesuai dengan pergerakan tangan

5.2 Memperbaiki Sistem Senjata

Pada pembahasan sebelumnya Anda dapat melukai musuh dengan senjata. Akan tetapi, senjata tersebut tidak dapat bergerak bersamaan dengan animasi tangan. Pada pembahasan ini Anda akan menggunakan senjata yang baru saja ditambahkan supaya dapat melukai musuh dengan pergerakan sesuai dengan tangan. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Pada objek **Melee** yang berada di **Hold_R** buat **Javascript** dengan nama "NewWeaponSwitch".

Gambar 5.54 Membuat skrip NewWeaponSwitch

2. Tulisakan variabel seperti berikut.

```
#pragma strict
var currentWeapon =0;
var maxWeapons = 2;
var theAnimator : Animator;
```

Variabel currentWeapon merupakan variabel untuk menentukan senjata yang sedang digunakan. Variabel maxWeapon untuk membatasi jumlah senjata yang dapat digunakan. Nilai 2 pada variabel berarti membatasi senjata yang dapat digunakan 3 (IronMace dan Knife). Variabel theAnimator untuk mengatur animasi gerakan.

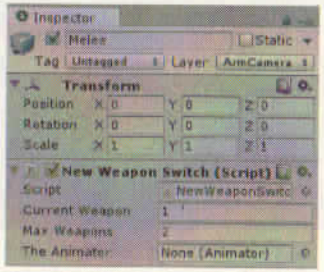
3. Selanjutnya, ketikkan skrip berikut.

```
function Update ()
{
        if(Input.GetAxis("Mouse ScrollWheel") <0)
        {
                if(currentWeapon +1 <= maxWeapons)
                {
                        currentWeapon++;
                }
```

```
                else
                {
                        currentWeapon =0;
                }
                SelectWeapon(currentWeapon);
        }
        else
        if(Input.GetAxis("Mouse ScrollWheel") >0)
        {
                if(currentWeapon - 1 >= 0)
                {
                        currentWeapon--;
                }
                else
                {
                        currentWeapon = maxWeapons;
                }
                SelectWeapon(currentWeapon);
        }
        if(currentWeapon == maxWeapons + 1)
        {
                currentWeapon =0;
        }
        if(currentWeapon == -1)
        {
                currentWeapon = maxWeapons;
        }
}

function SelectWeapon (index : int)
{

}
```

Pada fungsi Update, Anda memberikan pengaturan untuk pergantian senjata. Cara untuk mengganti senjata dengan scroll mouse. Pada fungsi tersebut Anda dapat memindahkan senjata dari senjata utama ke senjata satu atau dua.

4. Jika Anda klik tombol **Play**, Anda akan melihat variabel baru pada jendela **Inspector** dari **Melee**. Scroll mouse untuk mengubah angka. Angka ini mewakili pergantian senjata.



Gambar 5.55 Perubahan angka sebagai pergantian senjata

5. Anda juga dapat membuat pergantian senjata menggunakan tombol keyboard dengan menambahkan skrip berikut pada bagian bawah fungsi Update agar Anda dapat mengganti senjata dengan menekan tombol **1-2** pada keyboard.

```
if(Input.GetKeyDown(KeyCode.Alpha1))
        {
                currentWeapon = 0;
                SelectWeapon(currentWeapon);
        }
        if(Input.GetKeyDown(KeyCode.Alpha2))
        {
                currentWeapon = 1;
                SelectWeapon(currentWeapon);
        }
        if(Input.GetKeyDown(KeyCode.Alpha3))
        {
                currentWeapon = 2;
                SelectWeapon(currentWeapon);
        }
```

6. Ketikkan skrip berikut pada fungsi Awake.

```
function Awake ()
{
        SelectWeapon(0);
}
```

7. Pada fungsi SelectWeapon ketikkan skrip berikut untuk membuat animasi dari gerakan tangan.

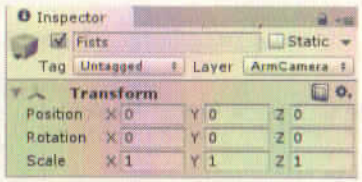
```
function SelectWeapon (index : int)
{
        for (var i = 0; i < transform.childCount; i++)
        {
                if (i == index)
                {
                        if (transform.GetChild(i).name ==
"Fists")
                        {
        theAnimator.SetBool("WeaponIsOn", false);
                        }
                        else
                        {
        theAnimator.SetBool("WeaponIsOn", true);
                        }
        transform.GetChild(i).gameObject.SetActive(true);
                }
                else
                {
        transform.GetChild(i).gameObject.SetActive(false)
;
```

}
}

Animasi akan dibagi dua, yaitu saat memegang senjata (IronMace dan Knife) dan tanpa senjata (Fists). WeaponIsOn merupakan kondisi yang menentukan apakah tangan sedang memegang senjata atau tidak.

8. Selanjutnya, klik kanan pada **Melee** yang ada di **Hold_R**. Kemudian pilih **Create Empty**.

9. Beri nama "Fists". Objek ini akan mewakili tangan saat tidak memegang senjata. Buat semua rotasi dan posisi menjadi 0.

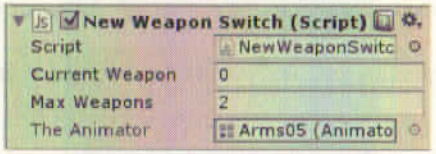


Gambar 5.56 Posisi dan rotasi objek Fists

Gambar 5.57 Senjata yang digunakan karakter

10. Selanjutnya, klik **Melee** dan lihat pada jendela **Inspector** terdapat **The Animator** yang masih kosong.

11. Drag objek **Arms05** ke dalam **The Animator**.



Gambar 5.58 Menambahkan The Animator

12. Klik tombol **Play** untuk menguji game.

13. Anda dapat mengganti senjata dengan scroll mouse atau menekan tombol 1-3. Pada saat memegang senjata, tangan akan menggenggam, sedangkan tanpa senjata tangan akan terbuka.

Gambar 5.59 Posisi tangan saat senjata Fists (tidak bersenjata)

Gambar 5.60 Posisi tangan saat memegang senjata

14. Selanjutnya, Anda akan membuat skrip supaya serangan karakter dapat melukai musuh. Buat **Javascript** di **Melee** dengan nama "MeleeSystem" atau nama yang Anda inginkan.

15. Ketik variabel seperti berikut.

```
#pragma strict

var TheDammage : int = 50;
private var Distance : float;
```

```
var MaxDistance : float = 1.5;
var TheAnimator : Animator;
var DammageDelay : float = 0.6;
```

Secara umum variabel ini mirip dengan MeleeSystem yang lama. Hal ini karena jarak serangan akan dibatasi variabel MaxDistance dan nilai serangan adalah 50.

16. Tambahkan variabel lain seperti berikut.

```
private var Hit01Streak = 0;
private var Hit02Streak = 0;
```

Variabel di atas digunakan untuk menentukan animasi serangan karakter.

17. Buat fungsi Update seperti berikut.

```
function Update ()
{
        if (Input.GetButtonDown("Fire1"))
        {
                AttackDammage();
        }
}
```

Fungsi ini akan memberikan tombol untuk perintah menyerang, yaitu klik kiri.

18. Selanjutnya, buat fungsi AttackDamage seperti berikut.

```
function AttackDammage ()
{
        if (Random.value >= 0.5 && Hit01Streak <= 2)
        {
                TheAnimator.SetBool("Hit01", true);
                Hit01Streak += 1;
                Hit02Streak = 0;
        }
        else
        {
                if (Hit02Streak <= 2)
                {
                        TheAnimator.SetBool("Hit02",
true);
                        Hit01Streak = 0;
                        Hit02Streak += 1;
                }
                else
                {
                        TheAnimator.SetBool("Hit01",
true);
                        Hit01Streak += 1;
                        Hit02Streak = 0;
                }
        }
```



```
        yield WaitForSeconds(DammageDelay);

        var hit : RaycastHit;
        var ray =
Camera.main.ScreenPointToRay(Vector3(Screen.width/2,
Screen.height/2, 0));
        if (Physics.Raycast (ray, hit))
        {
                Distance = hit.distance;
                if (Distance < MaxDistance)
                {

        hit.transform.SendMessage("ApplyDammage",
TheDammage, SendMessageOptions.DontRequireReceiver);
                }
        }

        TheAnimator.SetBool("Hit01", false);
        TheAnimator.SetBool("Hit02", false);
}
```

Pada bagian atas pernyataan if, menjelaskan pergerakan Hit01 dan Hit02 secara bergantian. Jika karakter menggunakan senjata, maka yang dijalankan Swing01 dan Swing02. Kondisi ini aktif secara bergantian.

Pernyataan if di bawah variabel hit dan Ray menjelaskan formula supaya senjata dapat melukai musuh. Pada bagian TheAnimator mengatur pergerakan animasi serangan.

19. Klik objek **Melee** untuk membuka jendela **Inspector**.

Gambar 5.61 Skrip pada Melee

20. Pada bagian **The Animator** tambahkan **Arms05**.

21. Klik tombol **Play** untuk menguji game.
22. Sekarang Anda dapat menyerang musuh dan melukai musuh.

Gambar 5.62 Mendekati musuh dengan tangan kosong

Gambar 5.63 Menyerang musuh dengan IronMace

23. Setelah nyawa musuh 0, maka objek musuh akan menghilang.

Gambar 5.64 Musuh berhasil dikalahkan

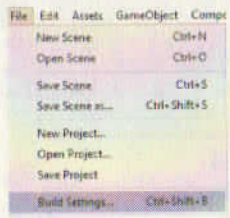
5.3 Membuat Game Menjadi Stand Alone

Sampai saat ini game telah berjalan dengan baik. Anda dapat menggunakan senjata untuk melawan musuh. Anda juga dapat bergerak bebas mengelilingi peta game. Bagian terakhir yang perlu Anda lakukan adalah membuat game dapat berjalan tanpa membuka Unity.

Sebelum membangun game, Anda dapat menambahkan jumlah musuh dengan cara menduplikasi jumlah musuh dan mengubah variabel musuh seperti nyawa dan serangannya.

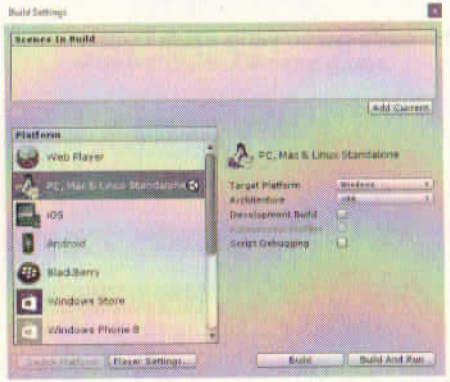
Untuk membuat game menjadi stand alone, ikuti langkah-langkah berikut:

1. Klik File > Build Settings.



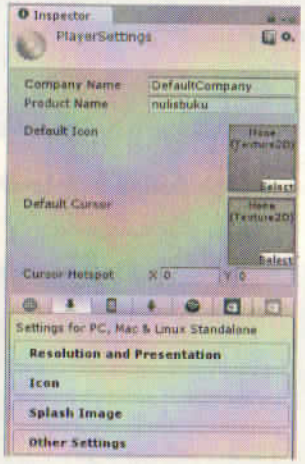
Gambar 5.65 Klik File > Build Settings

2. Selanjutnya, akan muncul kotak dialog **Build Settings**.



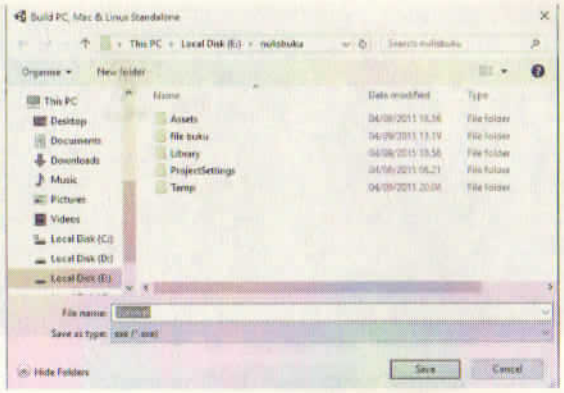
Gambar 5.66 Kotak dialog Build Settings

3. Tambahkan scene game dengan klik **Add Current.**
4. Selanjutnya, pilih platform yang akan digunakan untuk menjalankan game. Pada contoh ini tujuan platform adalah sistem operasi Windows 32 bit.
5. Klik **Player Setting** jika ingin menambahkan keterangan lain pada game.



Gambar 5.67 Player Settings

6. Klik **Build** untuk memulai proses pembuatan game menjadi stand alone.
7. Berikan nama pada game yang Anda buat.



Gambar 5.68 Memberi nama game

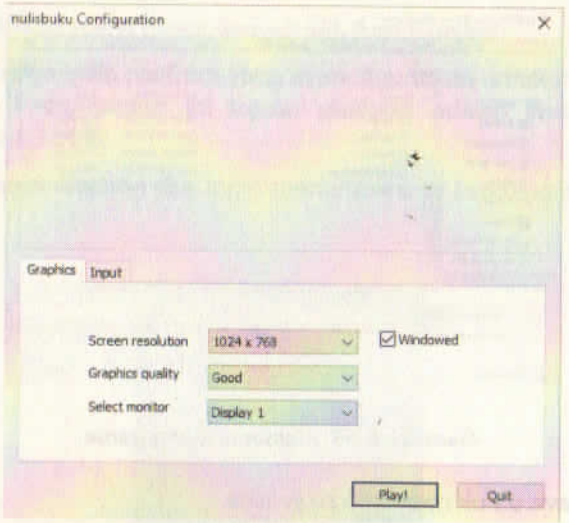
8. Klik **Save** untuk melanjutkan proses.

Gambar 5.69 Proses memacetkan game

9. Setelah proses selesai, Anda akan mendapatkan file **exe**.

Gambar 5.70 File exe game

10. Klik file **exe** untuk menjalankan game.



Gambar 5.71 Menjalankan file exe

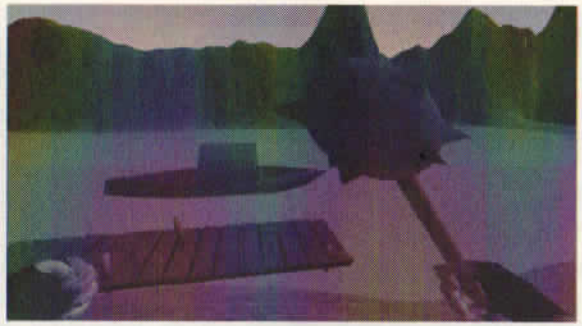
11. Anda dapat mengatur tampilan game pada tab **Graphic**.
12. Anda dapat melihat tombol yang digunakan dalam game dengan klik tab **Input**.
13. Klik **Play** untuk menjalankan game.

Gambar 5.72 Game berhasil dijalankan

Gambar 5.73 Mengganti senjata

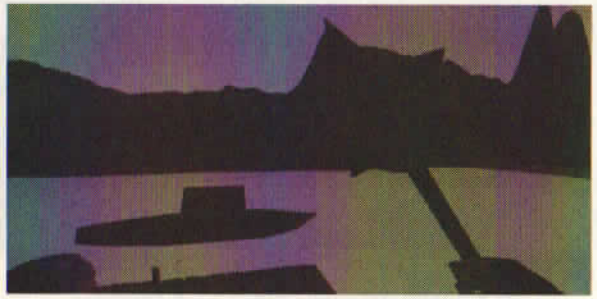
Gambar 5.74 Musuh siap menyerang

Gambar 5.75 Bertarung dengan musuh



Gambar 5.76 Kondisi danau siang hari

Gambar 5.77 Matahari mulai tenggelam



Gambar 5.78 Kondisi malam hari

14. Klik tanda silang untuk menutup game.

5.4 Kompilasi Game untuk Android

Pada pembahasan ini Anda akan meng-compile game khusus untuk platform Android. Secara umum game yang telah dibuat dapat di-compile pada platform Android, tetapi untuk menjalankan karakter, Anda perlu mengubah skrip yang mengatur pergerakan karakter menjadi yang mendukung touchscreen. Jika Anda menemui error pada script saat compile, Anda dapat menghapus script tersebut.

Untuk membuat game yang Anda buat dapat dimainkan menggunakan perangkat Android, Anda perlu menginstal beberapa hal berikut:

- JDK
- SDK
- Perangkat Android (optional)

5.4.1 Menginstal JDK

JDK (*Java Development Kit*) merupakan persyaratan utama supaya Anda dapat meng-install SDK. JDK dapat Anda download dari situs:

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Pada halaman tersebut, pilih versi JDK yang sesuai dengan sistem operasi yang Anda gunakan.

Java SE Development Kit 8u60

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ **Accept License Agreement** ◉ **Decline License Agreement**

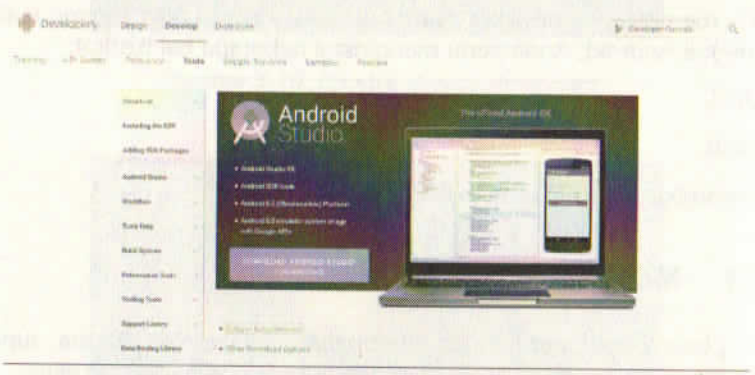
Product / File Description	File Size	Download
Linux ARM v6/v7 Hard Float ABI	77.69 MB	jdk-8u60-linux-arm32-vfp-hflt.tar.gz
Linux ARM v8 Hard Float ABI	74.64 MB	jdk-8u60-linux-arm64-vfp-hflt.tar.gz
Linux x86	154.66 MB	jdk-8u60-linux-i586.rpm
Linux x86	174.83 MB	jdk-8u60-linux-i586.tar.gz
Linux x64	152.67 MB	jdk-8u60-linux-x64.rpm
Linux x64	172.84 MB	jdk-8u60-linux-x64.tar.gz
Mac OS X x64	227.07 MB	jdk-8u60-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.67 MB	jdk-8u60-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.02 MB	jdk-8u60-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	140.18 MB	jdk-8u60-solaris-x64.tar.Z
Solaris x64	96.71 MB	jdk-8u60-solaris-x64.tar.gz
Windows x86	180.82 MB	jdk-8u60-windows-i586.exe
Windows x64	186.16 MB	jdk-8u60-windows-x64.exe

Gambar 5.79 Pilihan JDK berbagai sistem operasi

5.4.2 Menginstal SDK

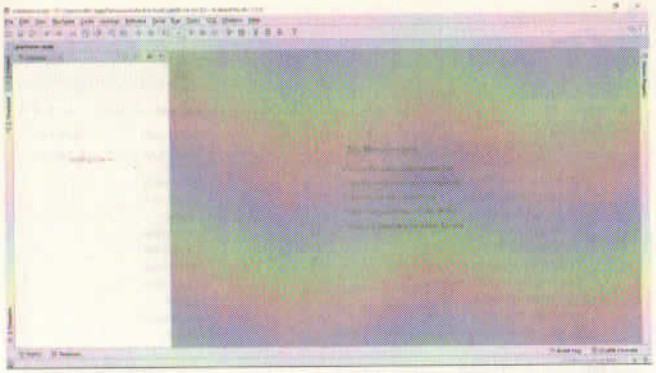
Untuk meng-compile game menjadi platform Android, Anda perlu menggunakan root dari SDK sekaligus perangkat virtual Android. Untuk menginstal SDK, ikuti langkah-langkah berikut:

1. Anda dapat download SDK dari https://developer.android.com/sdk/index.html.

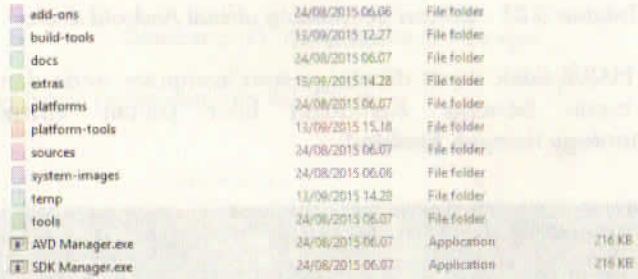


Gambar 5.80 Situs untuk download SDK

2. Setelah berhasil menginstal SDK, Anda akan melihat Android Studio terpasang di komputer. Jika Anda perhatikan, Anda akan menemui folder Android di dalam drive tempat Anda menginstal atau di dalam folder **user/nama_user/AppData/Local**. Pada folder tersebut, Anda akan menemui SDK Manager.exe dan AVD Manager.exe.

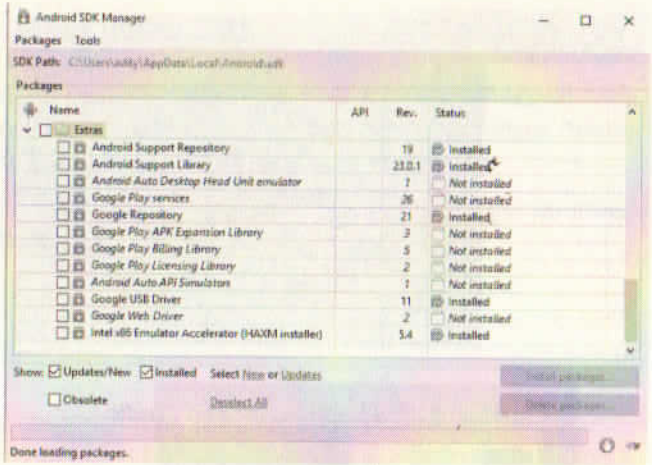


Gambar 5.81 Tampilan Android Studio



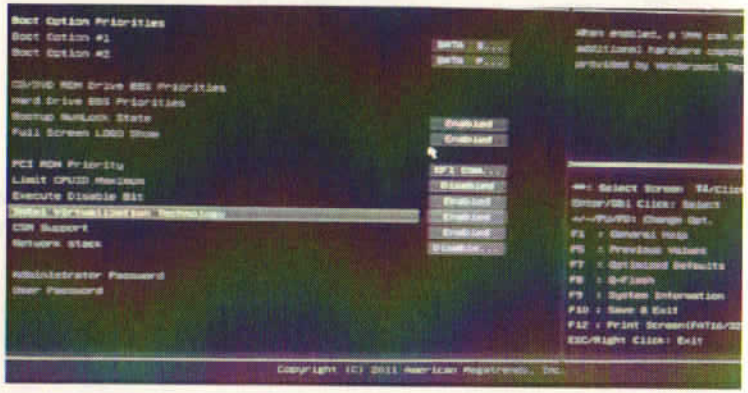
Gambar 5.82 Isi dari folder Android

3. Klik **SDK Manager** dan install paket yang dibutuhkan, terutama HAXM Installer jika HAXM gagal diinstal saat Anda menginstal Android Studio.



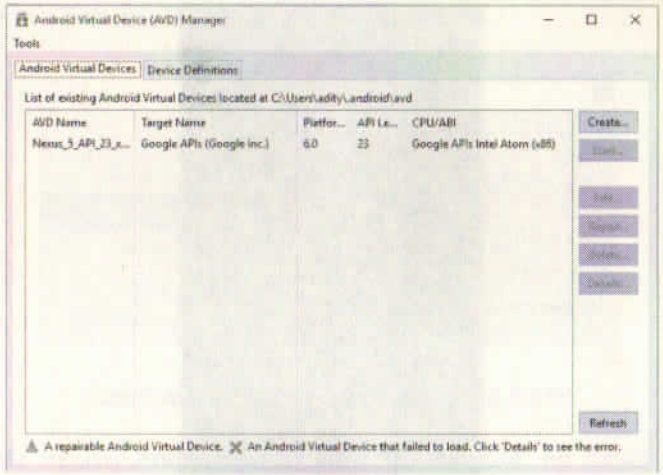
Gambar 5.83 Aplikasi pendukung virtual Android device

4. Jika HAXM tidak dapat diinstal, restart komputer Anda dan masuk ke menu booting. Kemudian buat pilihan **Virtualization Technology** menjadi **Enabled**.



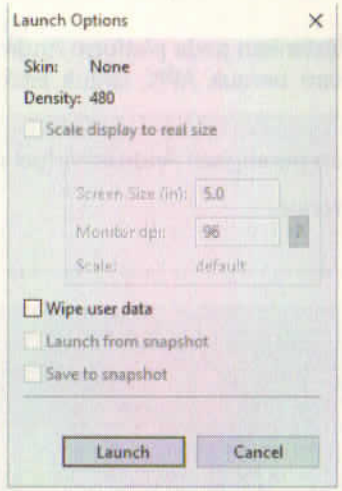
Gambar 5.84 Membuat Virtualization Technology menjadi Enabled

5. Selanjutnya, Anda dapat kembali menginstal HAXM.
6. Buka AVD Manager dan ikuti proses pemasangan sehingga Anda dapat melihat perangkat virtual Android.



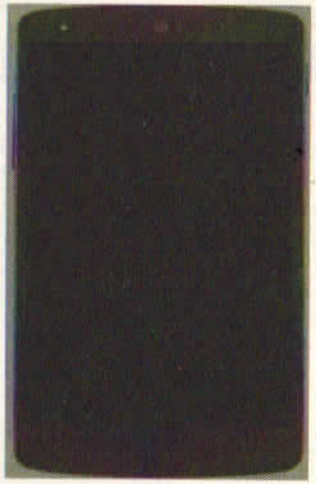
Gambar 5.85 Tampilan AVD Manager

7. Klik nama perangkat dan klik **Start**.



Gambar 5.86 Pengaturan tampilan perangkat Android

8. Anda akan melihat tampilan smartphone Android di monitor.

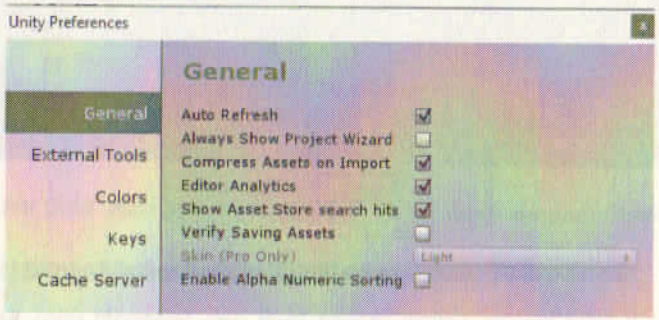


Gambar 5.87 Tampilan perangkat Android virtual

5.4.3 Compile Game Menjadi APK

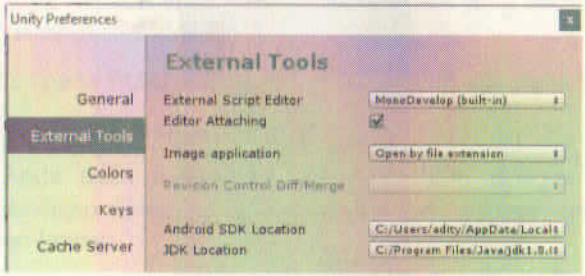
Supaya game dapat dijalankan pada platform Android, Anda akan meng-compile game ke dalam bentuk APK. Untuk lebih jelas, ikuti langkah-langkah berikut:

1. Buka Unity dengan perangkan Android virtual masih aktif.
2. Klik Edit > Preference.



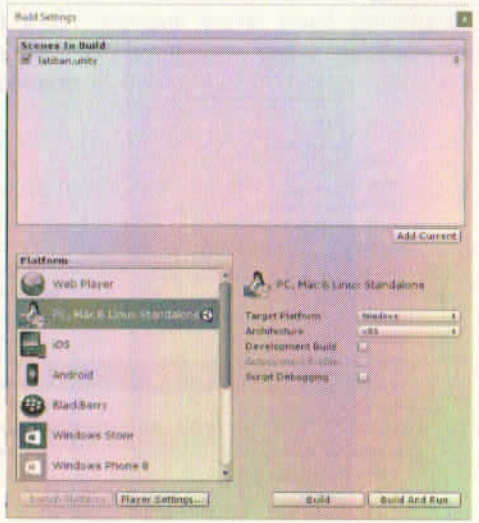
Gambar 5.88 Tampilan Unity Preference

3. Klik tab External Tools.



Gambar 5.89 Tampilan tab External Tools

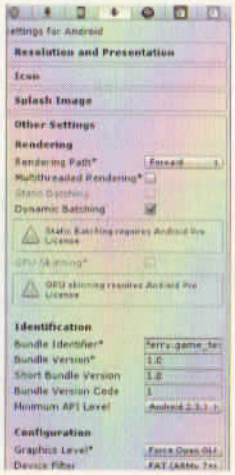
4. Pada bagian **Android SDK Location** arahkan ke folder tempat Anda menyimpan file SDK. File ini berada di dalam folder Android.
5. Pada bagian **JDK Location** arahkan ke folder tempat Anda meng-install JDK. Biasanya berada di dalam **Program File**.
6. Tutup External Tools dan klik File > Build Settings.
7. Klik Player Settings.



Gambar 5.90 Tampilan Build Setting

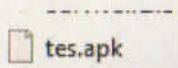
Gambar 5.91 Tampilan Player Setting

8. Pada bagian **Company Name** dan **Product Name** isi dengan nama yang Anda inginkan. Pastikan tidak menggunakan spasi.
9. Pada deretan ikon, pilih ikon **Android** (keempat dari kiri).



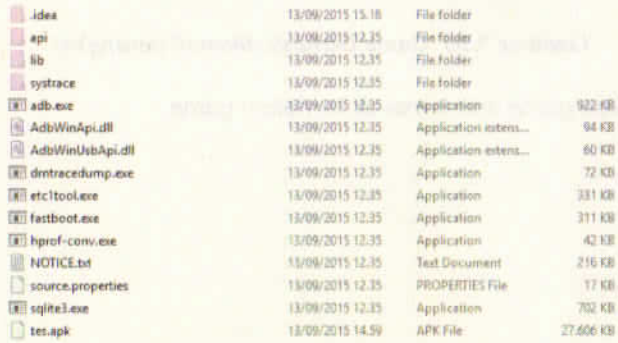
Gambar 5.92 Tampilan Player Setting

10. Pada bagian **Bundle Identifier** isi dengan dengan format Company_Name.Product_Name.
11. Anda juga dapat memilih minimum level API yang dapat digunakan pada perangkat Android.
12. Kembali pada **Build Setting**. Pastikan pilihan ada pada Android.
13. Selanjutnya, klik **Build**.
14. Jika Anda telah menginstal JDK dan SDK dengan benar dan mengarahkan Unity ke folder yang tepat, proses compile akan berjalan lancar.
15. Setelah proses selesai, Anda dapat menemukan file APK di dalam folder yang Anda pilih untuk menyimpan file saat proses compile.



Gambar 5.93 File APK game Android

16. File ini dapat Anda pindahkan langsung ke perangkat Android supaya dapat dimainkan atau Anda dapat mencobanya pada perangkat virtual Android.
17. Jika Anda ingin menguji game pada perangkat virtual Android, pindahkan file APK ke dalam folder **platform-tools** yang ada di dalam fodler **sdk**.



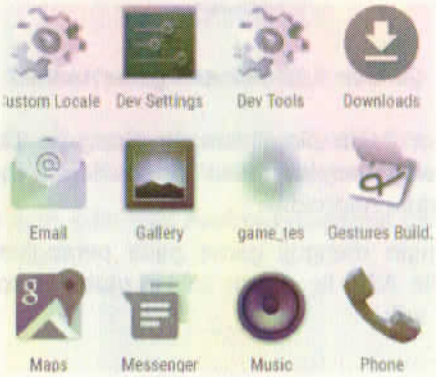
Gambar 5.94 Memindahkan file APK ke dalam platform-tool

18. Selanjutnya, buka command prompt dan arahkan ke dalam folder **platform-tools**. Gunakan perintah "cd" untuk pindah folder.

```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\adity>cd AppData\Local\Android\sdk\platform-tools

C:\Users\adity\AppData\Local\Android\sdk\platform-tools>_
```

Gambar 5.95 Pindah ke folder platform-tools

19. Selanjutnya, ketikkan **/adb install nama_APK**.
20. Jika berhasil, pada perangkat virtual Android akan ditampilkan game yang telah Anda buat.



Gambar 5.96 Game berhasil dikenali perangkat

21. Klik ikon game untuk masuk ke dalam game.

Gambar 5.97 Tampilan game dalam perangkat Android virtual

22. Untuk keluar game Anda harus menutup aplikasi secara manual karena Anda tidak menambahkan menu untuk menutup game.

Membuat Game **Android** dengan **Unity 3D**

Berkat Android, kini ponsel tidak hanya untuk bertelepon, tetapi juga bisa digunakan untuk bermain game. Penggunaan Android untuk bermain game pun makin merata, terutama di kalangan anak-anak dan remaja.

Unity 3D sendiri adalah game engine yang cukup populer. Game engine merupakan sebuah software yang digunakan untuk membuat atau mengembangkan game. Unity 3D banyak digunakan oleh kalangan pembuat game indie maupun orang yang baru belajar membuat game.

Buku ini menjelaskan cara menggunakan Unity 3D untuk membuat game. Anda aka buat game. Dari nol hingg me erangkat A

Kelompok
Pemrograman
Keterampilan
☑ Tingkat Pemula
☑ Tingkat Menengah
☐ Tingkat Mahir
Jenis Buku
☑ Referensi
☑ Tutorial
☐ Latihan

PT ELEX MEDIA KOMPUTINDO
Kompas Gramedia Building
Jl. Palmerah Barat 29-37, Jakarta 10270
Telp. (021) 53650110-53650111, Ext. 3214
Webpage: www.elexmedia.id

9 786020 277189
715052258