**RANCANG BANGUN MINI GAME RUN FORREST RUN BERBASIS UNITY-GAME ENGINE**

**Laporan ini disusun untuk mata kuliah: Game Programming II**

**Dosen pengajar: Kurnia Jaya Eliazar, S.T., M.T.**

******

Disusun Oleh:

Niko Parulian (061303036)

Arvy D.Andhika (061303021)

**SK.Ketua Badan Akreditasi Nasional Perguruan Tinggi (BAN-PT)**

**Nomor : 041/BAN-PT/AK-XIV/S1/XII/2011**

**TEKNIK INFORMATIKA – FAKULTAS TEKNIK**

**UNIVERSITAS WIDYATAMA**

**BANDUNG**

**2017**

# ABSTRAK

“Buatlah sebuah game sederhana namun menarik untuk dimainkan”, hal itulah yang diminta Bp. Kurnia untuk kami kerjakan sebagai tugas besar. Akhirnya kami memutuskan untuk merancang bangun sebuah mini game dengan judul “RUN FORREST RUN”. Idenya sendiri datang dari sebuah film yang berjudul “Forrest Gump”. Pada film tersebut, terdapat scene yang menggambarkan Forrest (pemeran utama) diganggu oleh para berandal. Para berandal itu menaiki sebuah mobil dan mengganggu Forrest dengan menggejar serta berusaha menabraknya. Ketika itu Jenny (teman Forrest semenjak kecil) meminta Forrest untuk melarikan diri dan berteriak “RUN FORREST RUN”. Konsep game “RUN FORREST RUN” sendiri sangat sederhana, player diharuskan berlari dari kejaran sebuah mobil yang berusaha menabraknya. Semakin lama player bertahan, semakin tinggi skor yang didapatkan. Namun pada prosesnya tidaklah mudah, karena terdapat beberapa tantangan pada gameplay. Seperti terdapat rintangan pada jalan yang harus player hindari, serta semakin meningkatnya kecepatan berlari Forrest, yang membuatnya semakin sulit dikendalikan. Pada proses rancang bangun mini game “RUN FORREST RUN” ini, kami menggunakan Unity-Game Engine.

# KATA PENGANTAR

Puji syukur kehadirat ALLAH SWT, karena atas perkenan-Nya lah laporan tugas besar dengan judul “Rancang Bangun Mini Game Run Forrest Run Berbasis Unity-Game Engine” ini dapat selesai dengan baik sesuai pada waktu yang telah ditentukan.

Tugas besar ini memiliki bobot penilaian yang besar pada matakuliah Game Programming II di Universitas Widyatama Bandung. Laporan tugas besar ini disusun sebagai pelengkap tugas besar yang telah mulai dikerjakan dari awal perkuliahan semester 8.

Selesainya tugas besar ini tidak terlepas dari bantuan beberapa pihak yang telah memberikan bantuan serta saran-saran konstruktif kepada kami. Untuk itu kami mengucapkan banyak terimakasih kepada:

* Febri Adenugraha, selaku teman kami yang menguasai seluk beluk Unity-Game Engine. Telah banyak membantu kami menyelesaikan masalah-masalah yang berhubungan dengan Unity-Game Engine.
* Yusuf Nugraha, selaku teman kami yang menguasai seluk beluk pemrograman. Telah banyak membantu kami dalam hal scripting dan algortima-algoritma sederhana yang powerfull.

Penulis menyadari bahwa masih banyak kekurangan pada laporan ini, baik dari materi maupun teknik penyajiannya. Oleh karena itu, kritik dan saran yang membangun sangat kami harapkan.

Akhir kata semoga laporan tugas besar ini dapat memberikan banyak manfaat bagi kita semua.

Bandung, Juni 2017

# DAFTAR ISI

ABSTRAK i

KATA PENGANTAR ii

DAFTAR ISI iii

DAFTAR GAMBAR iv

BAB I PENDAHULUAN 1

1.1. Latar Belakang Masalah 1

1.2. Tujuan 2

1.3. Batasan Masalah 2

1.4. Sistematika Penulisan 2

BAB II ANALISIS KEBUTUHAN DAN PERANCANGAN SISTEM 3

2.1. Analisis Kebutuhan 3

2.1.1. Kebutuhan Perangkat Keras 3

2.1.2. Kebutuhan Perangkat Lunak 3

2.2. Perancangan Sistem 3

2.2.1. Download Asset dari Asset Store 4

2.2.2. Membuat Game Object dan Scripting 5

2.2.3. Membuat Game Object Particle 19

BAB III IMPLEMENTASI DAN TAMPILAN AKHIR 25

3.1. Implementasi 25

3.2. Tampilan Akhir 25

BAB IV PENUTUP 26

4.1. Kesimpulan 26

DAFTAR PUSTAKA xxvii

# DAFTAR GAMBAR

[Gambar 221.1. Supercyan Character Male 4](#_Toc484570540)

[Gambar 221.2. Grass Road Race 5](#_Toc484570541)

[Gambar 221.3. 3D Models Vehicles Land 5](#_Toc484570542)

[Gambar 222.1. Object dan Script in Game 6](#_Toc484570543)

[Gambar 222.2. Directional Light 6](#_Toc484570544)

[Gambar 222.3. Plane 7](#_Toc484570545)

[Gambar 222.4. Main Camera 7](#_Toc484570546)

[Gambar 222.5. Inspector Main Camera 8](#_Toc484570547)

[Gambar 222.6. Inspector Tile Manager 9](#_Toc484570548)

[Gambar 222.7. Inspector Canvas 12](#_Toc484570549)

[Gambar 222.8. Inspector Player 13](#_Toc484570550)

[Gambar 222.9. Inspector Style 17](#_Toc484570551)

[Gambar 223.1. Particle Dust 20](#_Toc484570552)

[Gambar 223.2. Inspector Particle Dust 21](#_Toc484570553)

[Gambar 223.3. Particle Doink 21](#_Toc484570554)

[Gambar 223.4. Inspector Particle Doink 22](#_Toc484570555)

[Gambar 223.5. Particle Blood 23](#_Toc484570556)

[Gambar 223.6. Inspector Particle Blood 24](#_Toc484570557)

# BAB I PENDAHULUAN

## 1.1. Latar Belakang Masalah

Metode pembelajaran yang dinilai sangat efektif semenjak jaman penjajahan Jepang adalah belajar sambil mempraktekan apa yang sedang kita pelajari. Dengan cara ini kita akan dapat memahami, mengerti, dan mengingat lebih lama hal-hal sudah kita pelajari tersebut. Oleh karena hal itu, pada matakuliah Game Programming II ini, Bp. Kurnia selaku dosen matakuliah bersangkutan meminta kami untuk merancang bangun sebuah game sebagai tugas besar. Beberapa hal yang diminta Bp. Kurnia adalah: Game sederhana namun menarik, Ilmu-ilmu yang dipelajari selama perkuliahan Game Programming II harus diterapkan di dalam game tersebut.

Pada perkuliahan Game Programming II ini, tools yang digunakan adalah Unity-Game Engine. Game untuk tugas besar ini berbasis Unity.

Game ini akan menggambarkan adegan ketika Forrest dikejar oleh para berandal menggunakan mobil (seperti yang sudah di bahas pada abstract). Tujuan dari game ini adalah mengumpulkan point sebesar-besarnya, dan point itu sendiri akan bertambah selama player berhasil melarikan diri dari kejaran mobil para berandal. Game akan berakhir ketika player ditabrak oleh mobil tersebut. Mobil para berandal akan digerakan oleh ai, dan akan terus menerus mengejar player. Selain itu, mobil tersebut akan terus meningkatkan kecepatannya, sehingga player diharuskan berlari lebih cepat lagi. Terdapat juga beberapa rintangan yang harus player hindari dengan cara melompat atau bergerak ke arah kiri atau kanan. Bila player gagal menghindari rintangan tersebut, player akan kehilangan 20% (dua puluh persen) dari total kecepatan berlari player. Semakin lama player bertahan, maka kecepatan berlari player pun akan semakin bertambah, sehingga player akan lebih sulit dikendalikan.

## 1.2. Tujuan

Tujuan yang ingin dicapai dari Rancang Bangun Mini Game Run Forrest Run Berbasis Unity-Game Engine ini adalah sebagai berikut:

* Merancang bangun sebuah game sederhana dan menarik berbasis Unity.
* Menerapkan ilmu-ilmu yang telah dipelajari pada matakuliah Game Programming II di dalam game.

## 1.3. Batasan Masalah

Dalam penyusunan laporan tugas besar ini, kami mencoba membuat batasan-batasan agar tidak terjadi penyimpangan dari tujuan yang diharapkan. Batasan-batasan tersebut adalah sebagai berikut:

* Game sebatas akan dimainkan melalui Unity-Game Engine.
* Audio pada game tidak akan kami prioritaskan.

## 1.4. Sistematika Penulisan

*Bab I Pendahuluan*. Bagian ini berisi tentang penjelasan latar belakang masalah yang ditemukan saat kerja praktek, tujuan masalah, batasan masalah, dan sistematika penulisan laporan.

*Bab II Analisis Kebutuhan dan Perancangan Sistem*. Bagian ini berisi tentang langkah-langkah prosedural yang ditempuh dalam membangun mini game “Run Forrest Run”.

*Bab III Implementasi dan Tampilan Akhir.* Bagian ini berisi tentang manual book dan tampilan akhir dari mini game “Run Forrest Run”.

*Bab IV Penutup*. Bagian ini berisi kesimpulan dari hasil tugas besar.

# BAB II ANALISIS KEBUTUHAN DAN PERANCANGAN SISTEM

## 2.1. Analisis Kebutuhan

Dalam pembuatan game ini digunakan perangkat yang dibagi menjadi dua perangkat, yaitu perangkat keras (hardware) dan perangkat lunak (software).

### 2.1.1. Kebutuhan Perangkat Keras

Perangkat keras yang digunakan oleh kami untuk merancang bangun game ini adalah laptop, dengan spesifikasi sebagai berikut:

* Processor Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz (4 CPUs),~2.3GHz.
* Memory 4096MB RAM.
* VGA NVIDIA GeForce 720M version 376.33.

### 2.1.2. Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan oleh kami untuk merancang bangun game ini adalah sebagai berikut:

* Unity-Game Engine v5.6.0f3 Personal (64bit).
* Adobe Photoshop CS3 PORTABLE version.

## 2.2. Perancangan Sistem

Perancangan sistem pada tugas besar ini akan mencakup beberapa aspek, yaitu:

* Download asset dari asset store.
* Membuat game object dan scripting.
* Membuat game object particle.

### 2.2.1. Download Asset dari Asset Store

Dalam pembuatan game untuk tugas besar ini, kami menggunakan beberapa asset gratis yang ada di asset store Unity. Asset-asset gratis yang kami gunakan adalah:

* **Supercyan Character Pack Free Sampe**

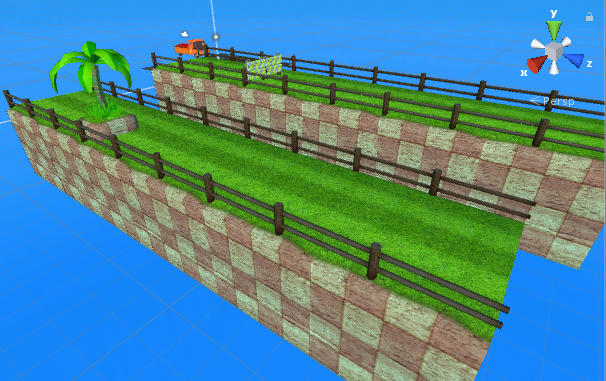
Asset ini menyediakan object textures character male cartoon. Kami menggunakan asset ini untuk tampilan pada object “Player” di dalam game (gambar 221.1).



Gambar 221.1. Supercyan Character Male

* **Grass Road Race**

Asset ini menyediakan object textures tanaman, ubin, jembatan, dan masih banyak lagi. Kami menggunakan asset ini untuk tampilan terrain did dalam game (gambar 221.2).



Gambar 221.2. Grass Road Race

* **3D Models Vehicles Land**

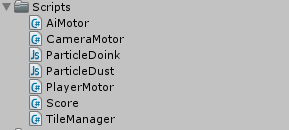
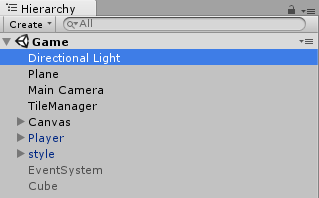
Asset ini menyediakan object textures mobil. Kami menggunakan asset ini untuk tampilan pada object “Ai” di dalam game (gambar 221.3).



Gambar 221.3. 3D Models Vehicles Land

### 2.2.2. Membuat Game Object dan Scripting

Pembuatan game menggunakan Unity-Game Engine, tentunya dibutuhkan object dan scripting. Disini kami akan menjelaskan object dan script apa saja yang digunakan di dalam game “RUN FOREST RUN” (gambar 222.1).



Gambar 222.1. Object dan Script in Game

* **Directional Light**

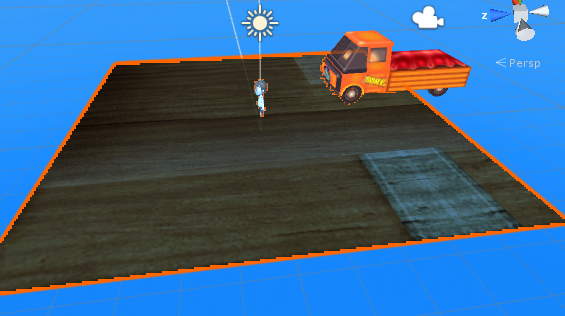
Untuk pencahayaan pada game. Directional Light ini akan berada tepat diatas kepala “Player” dan akan mengikuti kemanapun “Player” berpindah tempat (gambar 222.2).



Gambar 222.2. Directional Light

* **Plane**

Untuk pijakan awal “Player” dan “Ai” ketika game akan dimulai (gambar 222.3).



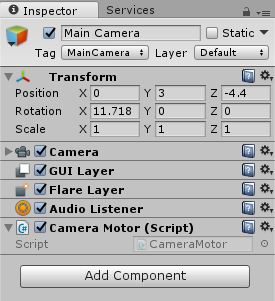
Gambar 222.3. Plane

* **Main Camera**

Untuk mengatur sudut pandang pemain ketika game dimainkan (gambar 222.4). Dan pada object ‘Main Camera’ ini terdapat script ‘CameraMotor’ (gambar 222.5) yang fungsinya agar object ‘Main Camera’ bergerak mengikuti pergerakan object “Player”.



Gambar 222.4. Main Camera



Gambar 222.5. Inspector Main Camera

* + **Script CameraMotor (C#)**

using UnityEngine;

using System.Collections;

public class CameraMotor : MonoBehaviour {

private Transform lookAt;

private Vector3 startOffset;

private Vector3 moveVector;

// Use this for initialization

void Start () {

//Metode untuk mencari posisi gameObject dengan tag “Player”.

lookAt = GameObject.FindGameObjectWithTag ("Player").transform;

//Metode mendapatkan selisih jarak antara Camera dan “Player”.

startOffset = transform.position - lookAt.position;

}

// Update is called once per frame

void Update () {

//moveVector akan bernilai jumlah yang dibutuhkan Camera untuk mencapai posisi “Player”.

moveVector = lookAt.position + startOffset;

//Tidak ada perubahan untuk axis x Camera.

// X

moveVector.x = 0;

//Math.Clamp adalah fungsi yang susah kami jelasikan. Namun intinya coding dibawah ini yang membuat posisi y Camera akan berpindah mengikuti posisi “Player”.

//Y

moveVector.y = Mathf.Clamp(moveVector.y,3,5);

//Eksekusi perubahan posisi axis pada Camera.

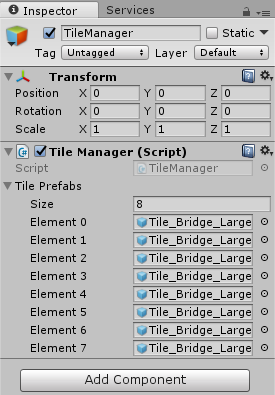
transform.position = moveVector;

}

}

* **TileManager**

Untuk memunculkan terrain yang akan digunakan sebagai track di dalam game. Pada object ini terdapat script “TileManager” yang fungsinya adalah untuk memunculkan terrain (terrain yang sudah di design oleh kami dan diinput kedalam inspector) secara random sebagai track. Dan script ini pun berfungsi untuk men-destroy terrain yang sudah dilewati oleh “Player” (gambar 222.6).



Gambar 222.6. Inspector Tile Manager

* + **Script TileManager (C#)**

using System.Collections;

using UnityEngine;

using System.Collections.Generic;

public class TileManager : MonoBehaviour

{

public GameObject[] tilePrefabs;

private Transform playerTransform;

private float spawnZ = -6.0f;

private float tileLength = 9.0f;

private float safeZone = 15.0f;

private int amnTileOnScreen = 7;

private int lastPrefabIndex = 0;

private List<GameObject> activeTiles;

// Use this for initialization

private void Start () {

activeTiles = new List<GameObject>();

//Metode untuk mencari posisi gameObject dengan tag “Player”.

playerTransform = GameObject.FindGameObjectWithTag ("Player").transform;

//Looping untuk memunculkan terrain sebanyak 7 terrain kedepan dari posisi “Player” berada.

for (int i = 0; i < amnTileOnScreen; i++)

{

SpawnTile ();

}

}

//Metode untuk menghapus terrain bila sudah berada pada jarak yang ditentukan (berada di belakan “Player”), dan untuk memunculkan terrain di depan “Player”. Jadi sistemnya 1 terrain dibelakang “Player” akan hilang, dan 1 terrain didepan “Player” akan muncul secara terus menerus (looping).

private void Update () {

if (playerTransform.position.z - safeZone > (spawnZ - amnTileOnScreen \* tileLength))

{

SpawnTile ();

DeleteTile ();

}

}

//Metode untuk memilih secara random terrain yang akan dimunculkan.

private void SpawnTile(int prefabIndex = -1)

{

GameObject go;

go = Instantiate (tilePrefabs [RandomPrefabIndex()]) as GameObject;

go.transform.SetParent (transform);

go.transform.position = Vector3.forward \* spawnZ;

spawnZ += tileLength;

activeTiles.Add (go);

}

private void DeleteTile()

{

Destroy (activeTiles [0]);

activeTiles.RemoveAt (0);

}

private int RandomPrefabIndex()

{

if (tilePrefabs.Length <= 1)

return 0;

int randomIndex = lastPrefabIndex;

while (randomIndex == lastPrefabIndex)

{

randomIndex = Random.Range (0, tilePrefabs.Length);

}

lastPrefabIndex = randomIndex;

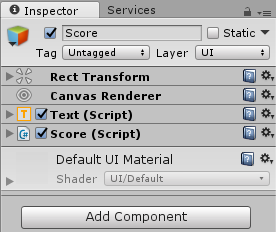
return randomIndex;

}

}

* **Canvas**

Untuk memunculkan score pemain di dalam game. Score sendiri akan bertambah selama “Player” berhasil melarikan diri dari kejaran “Ai”. Pada object ini terdapat script “Score” yang kami gunakan untuk: Membuat score bertambah setiap terjadi update(), dan menambah speed “Player” secara berangsur-angsur (gambar 222.7).



Gambar 222.7. Inspector Canvas

* + **Script Score (C#)**

using System.Collections;

using UnityEngine.UI;

using UnityEngine;

public class Score : MonoBehaviour

{

private float score = 0.0f;

private float score2 =0.0f;

public Text scoreText;

public bool test;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

//Untuk menambah score sebanyak 1 per Update().

score++;

//Untuk menambah score2 sebanyak 1 per score berjumlah 10.

if (score == 10) {

//Boolean yang akan digunakan sebagai penambah speed player.

test = true;

score2++;

score = 0;

} else {

test = false;

}

//Menampilkan jumlah score2 ke canvas sebagai score akhir pemain.

scoreText.text = ((int)score2).ToString ();

}

}

* **Player**

Untuk object “Player” di dalam game. Object ini akan dapat pemain kendalikan, seperti melompat ataupun bergerak kearah kiri atau kanan. Di dalam object “Player” ini terdapat CharacterController dan beberapa script yang membuat object ini memiliki collision, dapat dikendalikan oleh pemain, dan memiliki efek particle ketika pemain menekan tombol tertentu atau object bersentuhan dengan object lainnya (gambar 222.8).



Gambar 222.8. Inspector Player

* + **PlayerMotor (C#)**

using System.Collections;

using UnityEngine;

public class PlayerMotor : MonoBehaviour {

private CharacterController controller;

private Vector3 moveVector;

public float speed = 5.0f;

private float verticalVelocity = 0.0f;

private float gravity = 18.0f;

public bool kondisi;

public int jumpSpeed = 200;

// Use this for initialization

void Start () {

controller = GetComponent<CharacterController> ();

}

// Update is called once per frame

void Update () {

moveVector = Vector3.zero;

//Inisialisasi untuk mengambil nilai dari variable pada Class Score.

GameObject score = GameObject.Find ("Score");

Score budi = score.GetComponent<Score> ();

kondisi = budi.test;

//Untuk menambah speed object ketika bool true (sudah dijelaskan pada script sebelumnya).

if(kondisi == true){

speed += 0.10f;

}

if (controller.isGrounded) {

verticalVelocity = -0.5f;

//Untuk membuat object “Player” loncat ketika button space ditekan.

//untuk jump

if (Input.GetKeyDown (KeyCode.Space)) {

float ground = moveVector.y;

float counter;

for (counter = ground; counter < jumpSpeed; counter++) {

verticalVelocity += 2.00f \* Time.deltaTime;

}

}

//Untuk membuat object “Player” bergerak ke arah kiri dan kanan.

// X - left and Right

moveVector.x = Input.GetAxisRaw("Horizontal") \* speed;

}

else

{

//Untuk memberikan efek gravitasi bila object “Player” tidak menyentuh terrain.

verticalVelocity -= gravity \* Time.deltaTime;

}

// Y - Up and Down

moveVector.y = verticalVelocity;

//Untuk membuat object “Player” bergerak ke arah depan.

// Z - Forward and Backward

moveVector.z = speed;

controller.Move (moveVector \* Time.deltaTime);

}

//Fungsi untuk mengatur apa yang akan terjadi bila collision terjadi.

void OnControllerColliderHit(ControllerColliderHit hit){

float melanin = speed \* 20/100;

//Fungsi untuk mengurangi speed object ketika terjadi collision dengan object lain yang memiliki tag “Enemy”.

if (hit.collider.gameObject.tag == "Enemy") {

if (speed != 0) {

speed -= melanin;

} else if (speed <= 0) {

speed = 0;

}

}

}

}

* + **Script ParticleDoink (js)**

#pragma strict

var doink : Transform;

var blood : Transform;

function OnControllerColliderHit (info : ControllerColliderHit) {

//Untuk memunculkan particle doink ketika object “Player” collision dengan object lain yang memiliki tag “Enemy”.

if (info.gameObject.tag == 'Enemy') {

var dumb = info.gameObject;

Instantiate(doink, transform.position, transform.rotation);

//yield(3);

Destroy (dumb);

}else if (info.gameObject.tag == 'Ai') {

//Untuk memunculkan particle blood ketika object “Player” collision dengan object lain yang memiliki tag “Ai”, dan men-destroy object “Player”.

Instantiate(blood, transform.position, transform.rotation);

Destroy(this.gameObject);

}

}

* + **Script ParticleDust (js)**

#pragma strict

var dust : Transform;

function Update () {

//Untuk memunculkan particle dust ketika pemain menekan button space.

if(Input.GetKeyDown("space"))

{

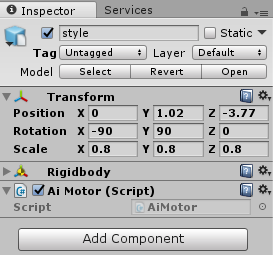
Instantiate(dust, transform.position, transform.rotation);

}

}

* **Style**

Untuk object “Ai” di dalam game. Karena terdapat script “AiMotor”, object ini akan bergerak secara otomatis zig-zag ke arah depan mengejar object “Player”. Object “Player” akan destroy bila collision dengan object ini. Inspector untuk object ini dapat dilihat pada gambar 222.9.



Gambar 222.9. Inspector Style

* + **AiMotor (C#)**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class AiMotor : MonoBehaviour {

private float speed = 0.083f;

private Transform lookAt;

private Vector3 startOffset;

private Vector3 moveVector;

private int speed2 = 0;

private bool kiriKanan = false;

private int counter = 0;

// Use this for initialization

void Start () {

moveVector = new Vector3();

// X

moveVector.x = 1.2f;

//Y

moveVector.y = 0.5f;

moveVector.z = -2;

}

// Update is called once per frame

void Update ()

{

//Untuk mengetahui apakan object sedang berada di sudut kiri atau kanan.

if (counter == 200) {

if (kiriKanan) {

kiriKanan = false;

} else {

kiriKanan = true;

}

counter = 0;

}

//Untuk menggerakan object ke arah kiri dan kanan per 200 counter.

if (kiriKanan) {

moveVector.x += 0.01f;

counter++;

} else {

moveVector.x -= 0.01f;

counter++;

}

//Untuk menambah speed object secara perlahan.

speed2++;

if (speed > 0.250f) {

speed = 0.250f;

} else {

if (speed2 == 70) {

speed2 = 0;

speed += 0.005f;

}

}

moveVector.z += speed;

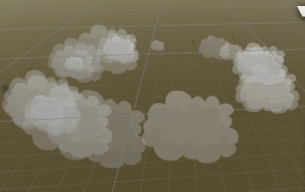
transform.position = moveVector;

}

}

### 2.2.3. Membuat Game Object Particle

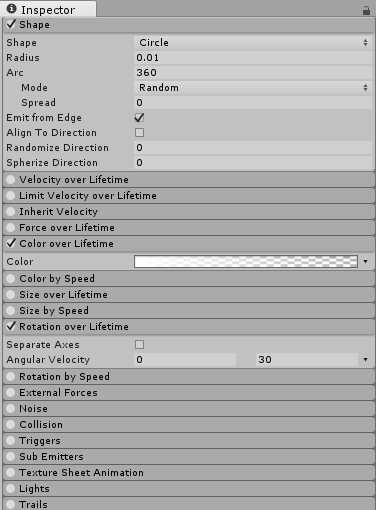
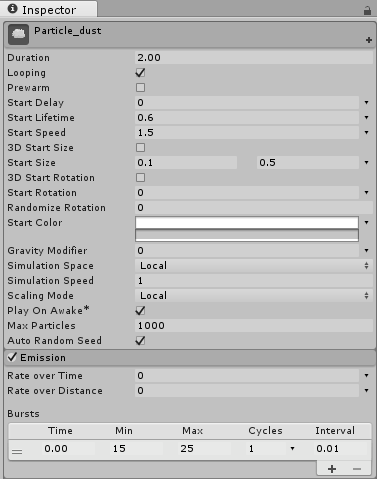
Salah satu ilmu yang dipelajari pada matakuliah Game Programming II adalah object particle effects. Oleh karena itu Bp. Kurnia mengharuskan kami menerapkan object particle effects di dalam game ini. Kami membuat dan menerapkan 3 object particle effects.

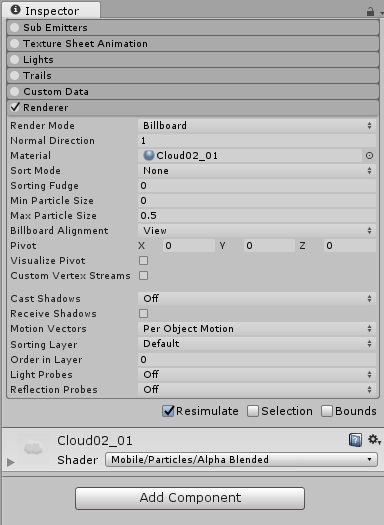


Gambar 223.1. Particle Dust

* **Particle Dust (gambar 223.1)**

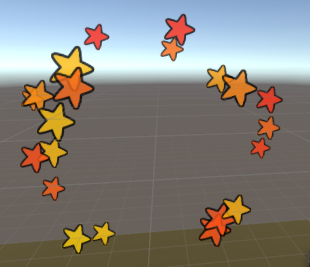
Untuk efek ketika object “Player” meloncat. Object “Player” akan mengeluarkan debu dan tanah dari bagian kakinya. Untuk cara pembuatan particle ini dapat dilihat pada gambar dibawah ini (gambar 223.2).





Gambar 223.2. Inspector Particle Dust

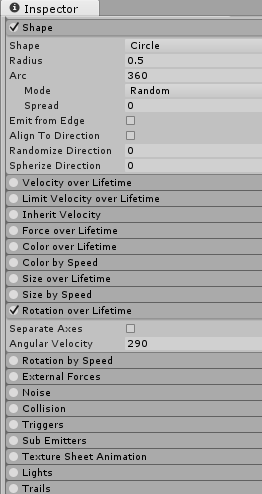
Untuk renderer material kami menggunakan image awan (.png), dengan shader Mobile/Particles/Alpha Blended.

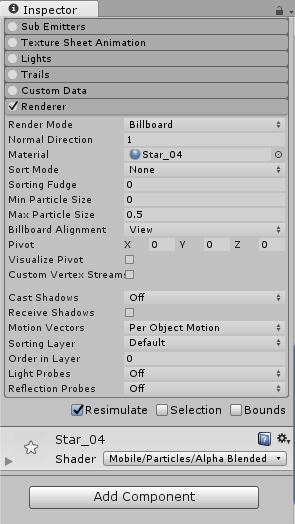


Gambar 223.3. Particle Doink

* **Particle Doink (gambar 223.3)**

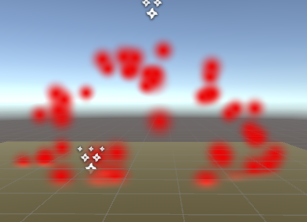
Untuk efek ketika object “Player” menabrak object lain dengan tag “Enemy”. Object “Player” akan mengeluarkan bintang-bintang dari bagian kepalanya. Untuk cara pembuatan particle ini dapat dilihat pada gambar dibawah ini (gambar 223.4).





Gambar 223.4. Inspector Particle Doink

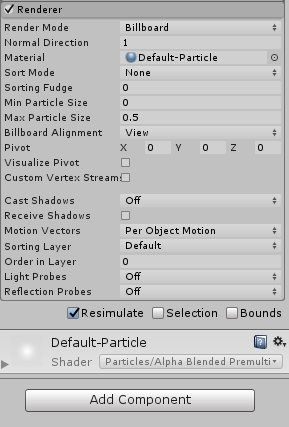
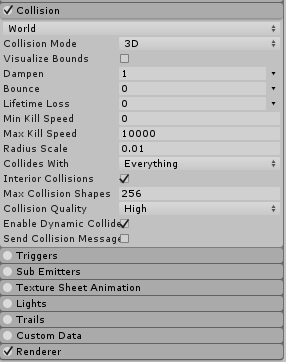
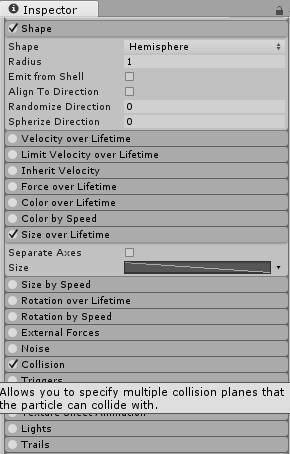
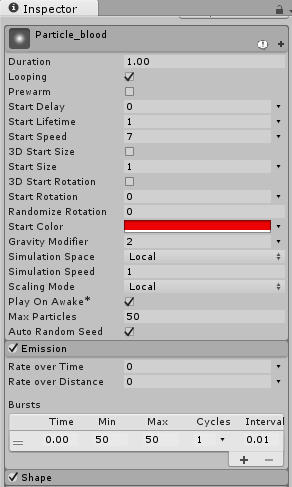
Untuk renderer material kami menggunakan image star (.png), dengan shader Mobile/Particles/Alpha Blended.



Gambar 223.5. Particle Blood

* **Particle Blood (gambar 223.5)**

Untuk efek ketika object “Player” menabrak object lain dengan tag “Ai”. Object “Player” akan mengeluarkan darah dari bagian badannya. Untuk cara pembuatan particle ini dapat dilihat pada gambar dibawah ini (gambar 223.6).



Gambar 223.6. Inspector Particle Blood

Untuk shader Mobile/Particles/Alpha Blended.

# BAB III IMPLEMENTASI DAN TAMPILAN AKHIR

## 3.1. Implementasi

Cara memainkan mini game “RUN FORREST RUN” ini sangat sederhana. Pemain cukup menekan button ‘a’ untuk bergerak ke arah kiri, button ‘d’ untuk bergerak ke arah kanan, dan space untuk meloncat.

## 3.2. Tampilan Akhir

Untuk tampilan akhir mini game “RUN FORREST RUN” dapat dilihat di [youtube](https://youtu.be/6igON5tyQRM): <https://youtu.be/6igON5tyQRM>.

# BAB IV PENUTUP

## 4.1. Kesimpulan

Setelah kami menyelesaikan tugas besar ini, kami dapat menyimpulkan beberapa point, yaitu:

* Unity-Game Engine adalah sebuah tools yang sangat powerfull dalam membangun sebuah game, namun tetap dibutuhkan investasi waktu untuk mempelajari cara penggunaannya.
* Metode pembelajaran dengan cara mempraktekan apa yang kita pelajari ke dalam sebuah project sangatlah efektif. Kami menjadi mengerti bagaimana mengimplementasikan teori yang sudah kami pelajari ke dalam sebuah project.
* Pentingnya Github (atau layanan sejenis) dalam mengerjakan sebuah project yang dikerjakan bersama-sama. Kami mengerjakan project ini tanpa menggunakan layanan Github atau sejenisnya, dan hasilnya kami harus bekerja dua kali, kami membuang banyak waktu serta tenaga hanya untuk menggabungkan pekerjaan kami.

# DAFTAR PUSTAKA

Unity. “Manual”. <https://docs.unity3d.com/Manual/index.html>. 17 April 2017

xOctoManx. "Unity 5 Tutorial: Particle Systems #01 C# - Cartoon Explosion." YouTube. YouTube, 27 July 2015. Web. 23 Mei 2017. <<https://youtu.be/yYctRzw_SU0>>.

xOctoManx. "Unity 5 Tutorial: Particle Systems #03 C# - Ground Slam Effect." YouTube. YouTube, 30 August 2015. Web. 25 Mei 2017. <<https://youtu.be/WFfS7HcG8qE>>.

Alexander Zotov. "Creating a blood splash using particle system in Unity 2D game." YouTube. YouTube, 19 January 2017. Web. 3 June 2017. <<https://youtu.be/ge9XMrKx8m8>>.