KALBIScientia
Jurnal Sains dan Teknologi

ISSN 2356 - 4393

Pengembangan Aplikasi Permainan Lari dengan Menggunakan Sensor Gerak Microsoft Kinect V2

Muhammad Hanim Mustafa Siregar[1)], Tedi Lesmana Marselino[2)]

Teknik Informatika, Institut Teknologi dan Bisnis Kalbis
Jalan Pulomas Selatan kav.22, Jakarta 13210
[1)] *Email: hanim342@gmail.com*
[2)] *Email: tedi.lesmana@kalbis.ac.id*

***Abstract**: This research is aimed to develop a game application that implements gesture recognition technology as the method of interaction with the game application. Gesture recognition technology is a technology that can recognize a person's gestures and translate those gesture into instructions that can be understood by a computer. By implementing gesture detection technology, the user must physically move to control the game application. Development of the game application is done using Interactive Multimedia System of Design and Development (IMSDD) methodology developed by Dastbaz. The software that is used is the game development engine called Unity3D paired with Microsoft Visual Studio Community 2015 IDE software and using the C# programming language. This research produces a game application for Windows operation system and uses the Microsoft Kinect motion sensor version 2 to interpret gestures into computer instructions.*

***Keywords**: gesture recognition, IMSDD, Microsoft Kinect v2, Unity3D, video game*

***Abstrak:** Penelitian yang dilakukan bertujuan untuk mengembangkan sebuah aplikasi permainan lari yang menerapkan teknologi pengenalan gerak-isyarat sebagai metode interaksi dengan aplikasi permainan. Teknologi pengenalan gerak-isyarat merupakan sebuah teknologi yang dapat mengenali gerak-isyarat seorang manusia dan menerjemahkan gerakan tersebut menjadi instruksi yang dapat dipahami oleh komputer. Dengan menerapkan teknologi gerak-isyarat, pengguna harus melakukan gerakan fisik untuk mengendalikan aplikasi permainan. Pengembangan aplikasi permainan dilakukan dengan menggunakan metode Interactive Multimedia System of Design and Development (IMSDD) yang dikembangkan oleh Dastbaz. Perangkat lunak yang digunakan adalah aplikasi pengembangan game Unity3D berpasangan dengan IDE Microsoft Visual Studio Community 2015 dan menggunakan bahasa pemograman C#. Penelitian menghasilkan sebuah aplikasi permainan untuk sistem operasi Windows dan menggunakan sensor gerak Microsoft Kinect versi 2 untuk menerjemahkan gerak-isyarat menjadi instruksi pada komputer.*

***Kata Kunci**: pendeteksian isyarat, IMSDD, Microsoft Kinect v2, Unity3D, video game*

I. PENDAHULUAN

Riset telah membuktikan bahwa anak-anak yang menggunakan media elektronik secara berlebihan lebih berkemungkinan untuk menderita obesitas, kekurangan aktivitas fisik, dan kesehatan yang lemah [1]. Salah satu contoh media elektronik yang populer adalah *video game. Video game* merupakan suatu media elektronik yang berhasil membawa inovasi interaktivitas ke dalam dunia hiburan yang sebelumnya bersifat pasif. Namun dengan kelebihannya tersebut, bermain *video game* membuat pemain jarang bergerak sehingga kekurangan olahraga.

Karena alasan tersebut, dalam penelitian ini peneliti membangun aplikasi permainan lari dengan *genre Endless Runner,* yaitu *game* dimana karakter pemain maju secara terus menerus di dalam dunia *game* dengan tujuan mencapai skor tertinggi sebelum karakter pemain mati, menggunakan sensor gerak *Microsoft Kinect v2.* Sebelumnya sudah ada aplikasi permainan olahraga yang menerapkan sensor gerak *Microsoft Kinect v2* sebagai masukan, yaitu *Kinect Sports*, namun aplikasi tersebut hanya memanfaatkan pergerakan horizontal pemain untuk menggeser karakter ke kiri atau ke kanan dan memajukan karakter secara otomatis. Aplikasi yang dikembangkan oleh peneliti memanfaatkan fitur *Gesture Recognition* sehingga selain bergeser ke kiri atau ke kanan, pemain didorong untuk berolahraga dengan berlari di tempat untuk menggerakkan karakter maju.

Tujuan dari penelitian ini adalah untuk membuat aplikasi permainan interaktif yang dapat mendorong

olahraga sambil bermain dengan menggunakan masukan gestur-isyarat.

II. METODE PENELITIAN

A. Interactive Multimedia System Design dan Development Cycle Dastbaz

Metode pengembangan perangkat lunak yang digunakan pada penelitian ini adalah *Interactive Multimedia System Design & Development (IMSDD) Cycle Dastbaz* [2]. Dastbaz menjelaskan tahapan-tahapan pengembangan dan perancangan sistem multimedia interaktif, yaitu [2]:

1. Analisis kebutuhan sistem

Tahapan ini menganalisis aspek-aspek yang dibutuhkan oleh sistem yang akan dibangun. Tahapan ini terdiri dari tiga tahap, yaitu: (a) Definisi sistem, mendefinisikan sistem untuk menentukan tujuan pembuatan sistem; dan (b) Kebutuhan *hardware* dan *software,* mengevaluasi perangkat keras dan perangkat lunak yang dibutuhkan oleh sistem.

2. Perancangan aplikasi

Tahapan ini mendeskripsikan secara lengkap rancangan sistem yang akan dibangun. Tahapan ini terdiri dari empat tahap, yaitu: (a) Metafora desain, menentukan model pada dunia nyata sebagai kunci perancangan antar muka pada sistem; (b) Jenis informasi dan format, menentukan jenis-jenis informasi yang dibutuhkan oleh sistem; (3) *Storyboard,* menggambarkan tampilan aplikasi pada setiap tahapan pemakaian; dan (4) Struktur navigasi, menjelaskan cara kerja navigasi pada aplikasi.

3. Implementasi

Setelah rancangan selesai ditentukan, maka tahap implementasi sistem dimulai dengan pengembangan aplikasi. Tahapan ini terdiri dari dua tahap, yaitu: (a) Pembuatan purwarupa dan sistem; dan (b) Melakukan *testing* pada purwarupa untuk mengidentifikasi masalah pada sistem dan rancangannya.

4. Evaluasi

Pada tahap ini sistem diuji untuk menentukan apakah sistem yang dibuat telah sesuai dengan tujuan awal pembuatan sistem.

B. Kerangka Berpikir

Pada penelitian ini, peneliti mengembangkan sebuah aplikasi komputer bernama "KinectRunner" yang menggunakan teknologi *gesture recognition.* Untuk pengembangan aplikasi dalam penelitian ini, konsep aplikasi diambil dari *video game* yang ber-*genre Endless Runner* seperti Subway Surfers.

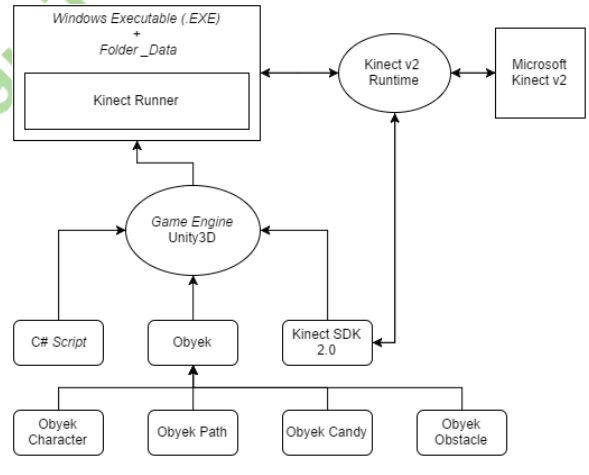
Pada Subway Surfers, pemain mengendalikan karakter yang berlari secara terus-menerus dan mengumpulkan koin emas untuk meningkatkan skor serta menghindari benturan dengan berbagai obyek seperti kereta atau pembatas jalan. Seluruh masukan pada Subway Surfers dilakukan melalui layar sentuh dengan menggunakan jari untuk menggeser karakter ke kiri dan ke kanan.

Aplikasi ini menggabungkan metode permainan Subway Surfers tersebut dengan teknologi *gesture recognition* pada perangkat *Kinect* yang dapat menangkap gerakan fisik pada tubuh manusia. Sehingga untuk pengoperasian aplikasi ini perlu berlari di tempat untuk menggerakkan karakter pemain maju dan untuk menggeser karakter pemain ke kiri atau ke kanan.

Tabel 1 Perbandingan subway surfers dengan kinectrunner

Subway Surfers	KinectRunner
Tidak menggunakan Kinect	Menggunakan Kinect
Masukan menggunakan layar sentuh	Masukan menggunakan gerak-isyarat
Karakter pemain berlari maju secara otomatis	Karakter pemain berlari maju jika pengguna berlari di tempat
Bekerja pada sistem operasi Android dan iOS	Hanya bekerja pada sistem operasi Windows 8 ke atas

Aplikasi ini dibuat menggunakan aplikasi pengembangan *game (Game Engine)* Unity3D dan ditujukan untuk sistem operasi *Windows 8* atau versi lebih tinggi.



Gambar 1 Diagram kerangka berpikir

Seperti yang terlihat pada Gambar 1, pengembangan aplikasi dilakukan dengan menggabungkan berbagai obyek dengan C# script untuk memberikan obyek-obyek tersebut fungsionalitas di dalam lingkungan pengembangan Unity3D. Untuk memberikan akses sensor Microsoft Kinect v2 kepada Unity3D, menggunakan plugin Kinect SDK yang menghubungkan Unity3D dengan

Kinect v2 Runtime yang memanajemen hubungan laptop dengan sensor Microsoft Kinect v2. Setelah pengembangan aplikasi selesai, hasil pengembangan disimpan sebagai folder dengan isi file Windows Executable (.EXE) dan folder berisi data aplikasi.

B. Multimedia

Multimedia adalah penggunaan komputer untuk menggabungkan teks, suara, gambar, animasi dan video dengan alat bantu *(tool)* dan koneksi *(link)* sehingga pengguna dapat bernavigasi, berinteraksi, berkarya dan berkomunikasi (Hofstetter 2001). Multimedia sering digunakan dalam dunia hiburan dan dunia *game* [3].

C. Natural User Interface

NUI *(Natural User Interface)* adalah istilah umum untuk beberapa teknologi seperti *speech recognition*, *multitouch* dan *kinetic interface* seperti *kinect*. Teknologi ini lebih unggul dari *graphical user interface* seperti interaksi menggunakan *keyboard* dan *mouse* yang umum digunakan di beberapa sistem operasi seperti Windows, Mac, Linux dan lain-lain. Teknologi ini memunculkan ciri lain dari NUI seperti interaksi antar user dan komputer akan terjadi tanpa perantara (media interakasi tidak akan terlihat) (Webb and Ashley, 2012) [4].

D. Gesture

Gesture adalah sikap atau pola suatu gerakan yang terlihat memperlihatkan suatu pesan. *Gesture* sendiri termasuk gerakan tangan, wajah atau bagian tubuh lainnya. *Gesture* memungkinkan individu untuk berkomunikasi berbagai perasaan dan pikiran, dari penjelasan, persetujuan dan kasih sayang dengan bahasa tubuh ketika berbicara [5].

E. Gesture Recognition

Gesture Recognition adalah antarmuka yang dapat mengenali gerak-isyarat seorang manusia dan mentranslasikan gerakan tersebut sebagai instruksi yang dapat dipahami oleh komputer. Sistem permainan Nintendo Wii dan Playstation Move memangaatkan joystick berbasis *accelerometer* dan *gyroscope* untuk mengenali kemiringan, perputaran, dan akselerasi. Sebuah jenis NUI yang lebih intuitif dilengkapi dengan kamera dan aplikasi dalam perangkat kerasnya. Sebagai contoh, Kinect dari Microsoft adalah sebuah sensor untuk Xbox 360 yang mengizinkan pemain berinteraksi menggunakan gerakan badan dan perintah ucapan. Kinect mengenali badan dan suara dari tiap pemain. *Gesture recognition* juga dapat digunakan untuk berinteraksi dengan komputer [6].

F. Microsoft Kinect V2

Microsoft Kinect v2 adalah perangkat input untuk mendeteksi gerakan yang diproduksi oleh *Microsoft* untuk *game console Xbox One,* serta untuk komputer berbasis *Windows*. Dengan menggunakan kamera yang mirip dengan *webcam, Kinect* memungkinkan pengguna untuk mengendalikan dan berinteraksi dengan game *console* atau komputer tanpa menggunakan perangkat input seperti *game controller, keyboard,* atau *mouse* melainkan menggunakan *natural user interface* yang memanfaatkan pergerakan gestur dan perintah suara [7]. Kinect memiliki fitur-fitur yang meliputi kamera RGB, *depth sensor* atau sensor kedalaman, dan *multi-array michrophone*. Dari fitur-fitur tersebut, *depth sensor* merupakan fitur yang berperan dalam membedakan *Kinect* dengan kamera. *Depth sensor* digunakan untuk mendapatkan data sebuah area dalam bentuk 3D tanpa memperdulikan kondisi cahaya pada area tersebut.

Sensor *Kinect* berdasarkan pada *optical lenses* dan juga sensor *Kinect* memiliki beberapa kekurangan, sensor *Kinect* dapat bekerja dengan baik pada kondisi sebagai berikut [7]: *Horizontal viewing angle*: 70º; *Vertical viewing angle*: 60º; Jarak terbaik untuk user menggunakan *Kinect v2* adalah 50 cm sampai 4.5 meter; *Depth range*: 50cm sampai 4.5 meter; dan Suhu 5º sampai 35º Celcius (41º sampai 95º Fahrenheit)

1. *RGB Camera*

Kamera RGB berfungsi untuk pengenalan wajah dan fitur deteksi lainnya dengan mendeteksi tiga komponen warna yaitu *Red*, *Green*, dan *Blue*. Microsoft menamakannya "*RGB Camera*" dengan karena ditujukan pada tiga komponen warna tersebut. Kamera RGB ini mirip dengan webcam, yang dapat menangkap video pada resolusi 640 x 480 dengan warna 32bit pada 30 *frames* per detik. Beberapa resolusi dan *frame* standard yang dapat digunakan *Kinect* [7]: (a) Resolusi 1920 x 1080 Fps30 dengan format RGB; (b) Resolusi 1920 x 1080 Fps12 dengan format RAW YUV; dan (c) Resolusi 1920 x 1080 Fps15 dengan format YUV.

2. *Depth Sensor*

Depth sensor terdiri atas kombinasi dari *infrared laser projector* dan *monochrome CMOS sensor* yang mengambil data video dalam 3D

tanpa memperdulikan kondisi cahaya. *Infrared* pada *Kinect* tidak dapat dilihat secara kasat mata serta tidak berbahaya bagi tubuh manusia. *Infrared* mengirimkan ribuan sinar yang memantul untuk mengetahui objek yang ada didepannya. Sinar yang dihasilkan oleh *infrared* inilah yang ditangkap oleh kamera *monochrome CMOS sensor* dan mengukur waktu sinar yang ada setelah terpantul oleh objek yang ada di depannya. Hal ini memungkinkan *Kinect* untuk memetakan gambar 3D yang ada didepannya sampai kedalaman satu centimeter dan tiga milimeter untuk lebar dan tinggi [7].

3. *Skeletal Tracking*

Skeletal tracking adalah fitur yang disediakan oleh *Kinect SDK*, fitur ini dapat melacak titik sendi utama tubuh manusia. *Skeletal tracking* ini tidak lepas dari penggunaan *depth sensor*. *Depth sensor* yang memetakan objek-objek berdasarkan jarak yang akan dibandingkan dengan data hasil *training* sebelumnya. Data *training* tersebut dibuat menggunakan 100.000 *frame* gambar manusia yang diambil dari posisi yang berbeda-beda pada saat berdiri oleh karena itu ketika pada saat duduk *Kinect* tidak dapat mendeteksi kaki manusia tersebut [7].

G. Game Engine

Game engine adalah sebuah sistem perangkat lunak *(software)* yang dirancang untuk pembuatan dan pengembangan suatu *video game. Game engine* memberikan kemudahan dalam menciptakan konsep sebuah *game* yang akan di buat. Mulai dari *scripting,* A.I., dan bahkan sistem *networking. Game engine* dapat dikatakan sebagai jiwa dari seluruh aspek sebuah *game.* Tujuan digunakannya *game engine* adalah untuk mempermudah pembuatan bagian-bagian tertentu dalam *game,* membagi-bagi pengembangan *game* menjadi modul-modul tertentu, dan memudahkan kolaborasi antar pihak [8].

H. Unity3D

Unity adalah sebuah *game engine* yang digunakan untuk membangun *game* yang dapat dijalankan di berbagai platform seperti pada situs web*, Windows, MacOS, Linux, iOS,* dan *Android.*

Unity didesain untuk mudah digunakan dan penuh perpaduan dengan aplikasi yang professional. Grafis pada *Unity* dibuat menggunakan API OpenGL dan DirectX sehingga dapat menghasilkan grafis tingkat tinggi untuk membuat *video game* 3D, animasi *real-time* 3D, visualisasi arsitektur dan aplikasi interaktif lainnya [9].

Lisensi untuk pemakaian *Unity* ada dua bentuk, *Unity Personal* dan *Unity Pro.* Versi *Personal* tersedia secara gratis sedangkan versi *Pro* harus dibeli. *Unity Pro* memiliki beberapa keunggulan dibandingkan dengan *Unity Personal,* diantaranya kemampuan untuk merubah layar *splash,* merubah tampilan warna pada *editor,* dan bantuan teknisi *Unity* secara langsung [10].

K. Endless Runner

Endless Runner adalah sejenis *video game* dimana karakter pemain maju secara terus menerus di dalam dunia *game* yang tidak berakhir. Kendali permainan biasanya dibatasi untuk membuat karakter melompat, menyerang, atau melakukan gerakan khusus. Tujuan dari jenis *game* ini adalah untuk membuat karakter maju sejauh mungkin sebelum karakter pemain mati [11].

III. HASIL DAN PEMBAHASAN

A. Analisis Kebutuhan Sistem

Sistem yang dikembangkan bertujuan untuk menambahkan elemen olahraga fisik ke dalam video game ber-genre Endless Runner. Sistem ini akan menggunakan gerakan gesture-isyarat sebagai masukan dalam permainan, sehingga pengguna harus menggerakkan tubuhnya untuk bermain.

Pada penelitian ini, pengembangan aplikasi membutuhkan beberapa perangkat keras dan perangkat lunak. Perangkat tersebut tentunya berbeda antara pengembang dan target pengguna. Daftar perangkat keras dan lunak yang dibutuhkan serta digunakan oleh peneliti dalam proses pengembangan aplikasi ditunjukkan pada Tabel 2.

Tabel 2 Kebutuhan perangkat pengembangan

PERANGKAT KERAS	PERANGKAT LUNAK
Komputer atau Laptop	Sistem Operasi *Windows* 10 Pro 64-bit
Microsoft Kinect V2	*Unity3D v5.x.x*
Microsoft Kinect Adapter for Windows	*Microsoft Visual Studio 2015 Community Edition*
	Kinect SDK v2

Laptop yang digunakan pada penelitian ini adalah Microsoft Surface Pro 3 dengan spesifikasi sebagai berikut: (1) Prosesor Intel Core i5-4300U; (2) Grafik Intel HD Graphics 4400; (3)Sistem operasi Windows 10 Pro 64-bit; (4) Ram DDR3 4GB

Untuk perangkat keras dan lunak yang dibutuhkan oleh pengguna dalam mengoperasikan aplikasi dari hasil penelitian ini ditunjukkan pada Tabel 3.

Tabel 3 Kebutuhan perangkat pengguna

PERANGKAT KERAS	PERANGKAT LUNAK
Komputer atau Laptop	Sistem Operasi *Windows 8, Windows 8.1,* atau *Windows 10* Pro 64-bit
Microsoft Kinect V2	*Kinect Runtime v2*
Microsoft Kinect Adapter for Windows	

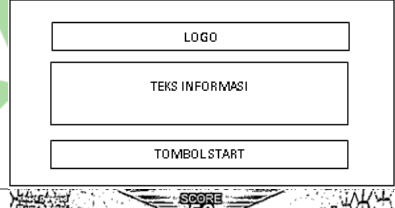
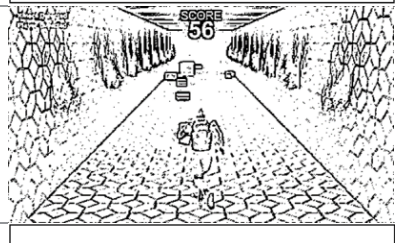
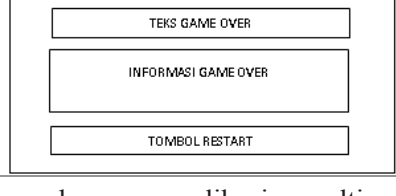
B. Perancangan Aplikasi

Metafora desain antar muka aplikasi akan meniru dari berbagai permainan komputer yang menggunakan perangkat tambahan *Microsoft Kinect,* sedangkan perancangan alur permainan akan meniru dari berbagai permainan dengan *genre Endless Runner.* Aplikasi yang dikembangkan akan menggunakan teks, grafik, suara, dan animasi. Teks dan grafik diatur untuk menyampaikan informasi kepada pemain tentang status permainan dan elemen navigasi. Suara dan animasi akan berguna untuk menampilkan *feedback* dari elemen-elemen interaktif kepada pemain.

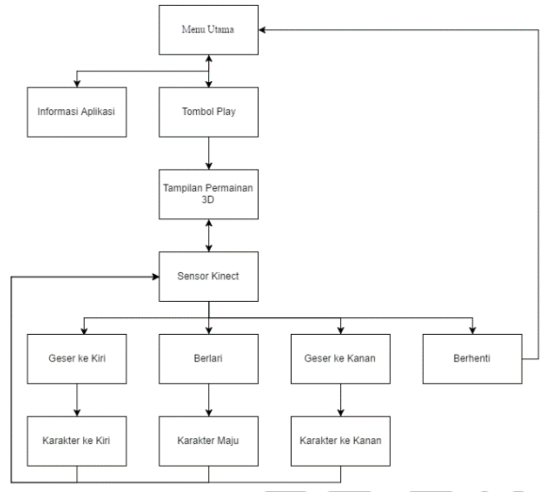
Format yang digunakan untuk grafik adalah *file* dengan format PNG yang berkualitas tinggi dan dapat menyimpan data *transparency.* Dari segi suara digunakan file berformat WAV, OGG, dan MP3 yang dapat memberikan kualitas suara yang cukup tinggi dengan ukuran file yang cukup kecil.

Pada penelitian ini perancangan storyboard diperlukan, storyboard yang akan digunakan ditunjukkan pada Tabel 4.

Tabel 4 Storyboard aplikasi

Scene 1	
Scene 2	
Scene 3	

Pada pengembangan aplikasi multimedia interaktif, dibutuhkan struktur navigasi yang jelas. Dengan demikian struktur navigasi pada aplikasi yang dikembangkan adalah seperti pada Gambar 2.



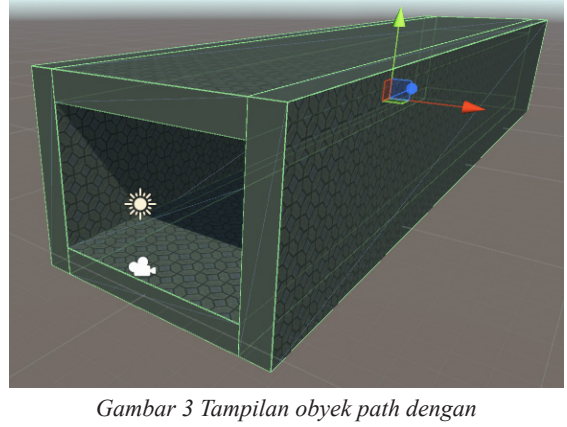
Gambar 2 Struktur navigasi

C. Implementasi

1. Pembuatan Model 3D Jalan

Dibuat model 3D yang akan berperan sebagai jalan menggunakan kubus dalam Unity. Langkah awal yaitu membuat *GameObject* kosong yang akan menampung obyek lantai, dinding, dan lantai. Langkah berikutnya yaitu membuat lantai bernama Floor di dalam obyek Path yang telah dibuat menggunakan *GameObject Cube.* Kemudian dengan cara yang serupa, dibuat dua kubus dalam obyek *Path* bernama WallLeft dan WallRight yang berfungsi sebagai dinding. Langkah terakhir pembuatan model 3D jalan adalah menambahkan obyek atap dengan nama *Ceiling* dalam obyek *Path*.

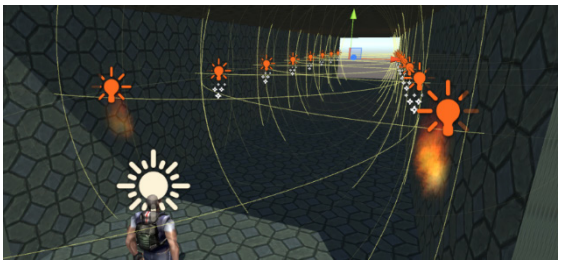
Kemudian untuk mempercantik tampilan, obyek Path akan ditambahkan tekstur Stone Floor Texture Tile yang telah diunduh dari Unity Asset Store. Untuk permukaan dinding dan lantai dibuat dua obyek tekstur yaitu StoneWall dan StoneFloor. Tampilan obyek Path setelah ditambahkan tekstur terlihat pada Gambar 3.



Gambar 3 Tampilan obyek path dengan tekstur

Berikutnya hapus obyek Directonal Light pada *Hierarchy,* dan kemudian di tambahkan cahaya

pada dinding untuk menerangi jalan di depan *Character* menggunakan obyek Torch. Obyek Torch mengandung obyek *Point Light* yang memancarkan cahaya dan sudah tersedia di dalam aset Campfire Pack yang telah diunduh dari Unity Asset Store. Tampilan Path yang selesai terlihat pada Gambar 4.

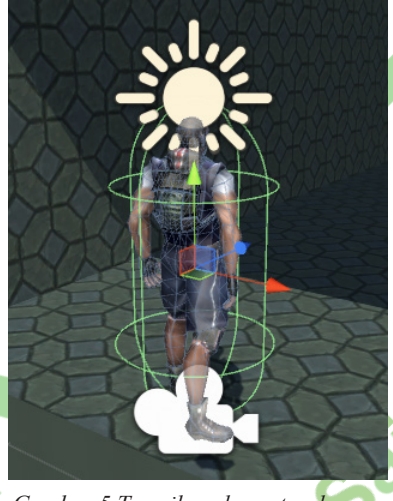


Gambar 4 Tampilan obyek path yang selesai

2. Pengaturan Karakter

Langkah awal penambahan karakter adalah dengan memasukkan aset *Max Adventure Model* yang telah diunduh dari Unity Asset Store sebagai obyek *Character* dan memberikannya tag Player.

Langkah berikutnya adalah pemasangan komponen *Character Controller* pada obyek *Character* sehingga terlihat seperti pada Gambar 5.



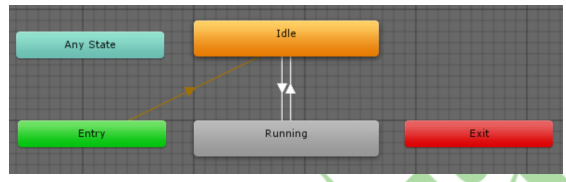
Gambar 5 Tampilan character dengan charactercontroller

Aset *Max Adventure Model* sudah dilengkapi dengan beberapa animasi. Langkah terakhir adalah mengatur animasi pada *Character* dengan membuat dan mengatur obyek *CharacterAnimator* yang menggunakan panel *Animator.*

Sebelum mengatur animasi pada *CharacterAnimator*, pertama perlu ditambahkan suatu variabel *boolean* bernama started. Variabel started akan bernilai *true* ketika permainan sudah dimulai dan membuat obyek *Character* mulai berlari.

Berikutnya menambahkan semua *state* animasi ke dalam *state machine* yang sudah tersedia dalam panel *Animator* dan mengatur semua transisi antara *state* tersebut. Setiap *state* adalah representasi untuk setiap animasi yang akan digunakan dan setiap transisi adalah perubahan antara *state* animasi yang dikendalikan oleh variabel *started*.

Tampilan *state machine* setelah semua pengaturan selesai terlihat seperti pada Gambar 6.



Gambar 6 Tampilan state machine animasi

3. Pembuatan Script GameManager dan UIManager

Dua *script* dibutuhkan untuk membantu memanajemen keadaan game, yaitu *script* GameManager dan *script* UIManager. GameManager memanajemen segala keadaan permainan dan UIManager memanajemen semua obyek antarmuka serta skor.

Sebelum membuat *script* tersebut, beberapa *script* pendukung lainnya perlu dibuat, yaitu *script Constants* dan GameState. Constants menampung variabel yang akan digunakan berkali-kali dalam aplikasi dan GameState berisi *enum* yang menampung semua *state* aplikasi (Start, Playing, Dead). Kedua *script* tersebut disimpan di dalam folder Scripts pada Assets dan kemudian akan digunakan oleh *script* lain yang membutuhkan.

Setelah membuat *script* Constants dan *script* GameState, berikutnya dibuat *script* GameManager dan UIManager yang disimpan di tempat yang sama dengan *script* sebelumnya. Agar *script* GameManager dan *script* UIManager dapat diakses oleh *script* lain, maka diperlukan obyek kosong untuk menampungnya. Buat obyek kosong pada *Hierarchy* dengan nama _GameManager kemudian tambahkan *script* GameManager dan *script* UIManager pada _ GameManager sebagai komponen.

4. Pembuatan Obyek Obstacle dan Candy

Kemudian dibuat dan diatur obyek Obstacle yang berfungsi sebagai rintangan bagi pemain, dan obyek Candy yang dikumpulkan pemain untuk meningkatkan skor. Semua obyek yang berfungsi sebagai Obstacle dan Candy masing-masing ditempatkan ke dalam folder Candy dan folder Obstacles pada folder Prefab

Berikutnya ditambahkan *script* Obstacle pada semua obyek dalam folder Obstacle sebagai komponen dan ditambahkan *script* Candy pada semua obyek dalam folder Candy sebagai komponen.

5. Pemanggilan Obyek Obstacle dan Obyek Candy

Sebelum dapat memanggil obyek Obstacle dan Candy, dibuat beberapa obyek yang berfungsi sebagai titik koordinat dimana obyek tersebut diletakkan secara acak. Langkah pertama adalah dengan membuat obyek kosong di dalam obyek Path yang bernama ObjSpawnPoints, dan mengatur posisi serta rotasinya. Langkah berikutnya adalah membuat obyek kosong sebagai anak dari ObjSpawnPoints yang berfungsi sebagai setiap koordinat pemanggilan obyek Obstacle dan Candy.

Berikutnya pada obyek ObjSpawnPoints ditambahkan *script* ObjSpawner sebagai komponen. Kemudian pada setiap properti pada komponen ObjSpawner, dimasukkan obyek-obyek yang akan dipanggil melalui *script.*

6. Penerapan Perulangan Obyek Path

Obyek Path diberikan *script* sehingga akan terpanggil berulang-ulang dan memberi ilusi bahwa jalan yang dilalui obyek *Character* adalah tidak terbatas. Langkah pertama adalah dengan membuat obyek *Collider* yang akan memicu pembuatan instansi Path baru. Dibuat obyek kosong di dalam obyek Path yang bernama PathSpawner.

Kemudian ditambahkan komponen *Box Collider* pada obyek tersebut dan centang *checkbox* pada properti *Is Trigger*. Langkah berikutnya adalah dengan membuat obyek kosong yang berfungsi sebagai koordinat dimana obyek Path berikutnya akan dipanggil. Buat obyek kosong di dalam obyek Path yang bernama PathSpawnPoint, dan atur posisinya.

Berikutnya ditambahkan *script* PathSpawner pada obyek PathSpawner sebagai komponen. Kemudian pada setiap properti pada komponen PathSpawner, dimasukkan obyek-obyek yang akan dipanggil.

7. Pemasangan dan Pengaturan Kamera pada obyek Character

Di dalam obyek *Main Camera* ditambahkan suatu komponen yaitu *script* CameraFollowZ. *Script* tersebut dapat membuat obyek *Main Camera* mengikuti obyek *Character* pada posisi Z. Kemudian pada properti *Object To Follow* dimasukkan obyek *Character* dari *Hierarchy.* Akhirnya, pada komponen *Camera* atur *Field of View* Menjadi 30.

8. Pembuatan Obyek Fog

Fog adalah obyek dengan jenis *Particle System* yang berfungsi untuk menghalangi jalan di depan pemain sehingga tidak terlihat ketika obyek Obstacle dan Candy dipanggil ke dalam *scene.*

Pembuatannya adalah dengan klik-kanan pada obyek Main Camera → klik *Particle System,* dan kemudian diatur posisinya.

9. Pembuatan Obyek ObjDestroyer

Kemudian dibuat obyek ObjDestroyer yang berfungsi untuk menghancurkan obyek Obstacle dan Candy yang sudah keluar dari pandangan pemain untuk membebaskan dan menghemat memori yang digunakan. Buat obyek kosong dengan nama ObjDestroyer, kemudian jadikan obyek tersebut sebagai anak dari obyek *Main Camera*.

Berikutnya, tambahkan komponen *Box Collider* dan Kemudian pada *Inspector*, klik *dropdown* berlabel *Tag* → klik *Add Tag* → klik pada ikon tambah dan beri nama tag baru tersebut Destroyer. Atur *Tag* ObjDestroyer menjadi Destroyer pada *Inspector.*

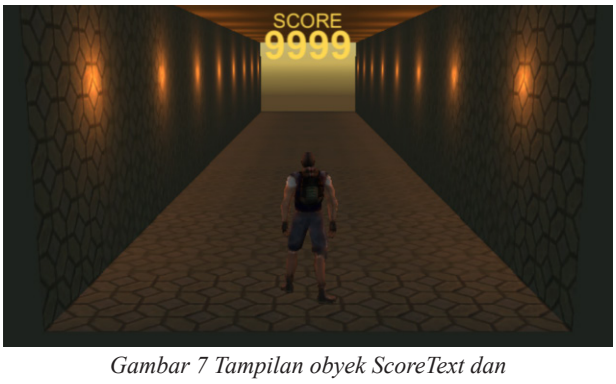
Akhirnya, tambahkan komponen *Rigidbody* pada ObjDestroyer dengan cara klik *Add Component* → klik *Physics* → klik *Rigidbody,* kemudian centang *checkbox* pada properti *Is Kinematic*

10. Pembuatan Teks Skor

Berikutnya dibuat dua obyek berjenis *Text* masing-masing bernama ScoreText dan Label_Score. Obyek ScoreText berfungsi untuk menampilkan skor dan obyek Label_Score berfungsi sebagai label untuk obyek ScoreText. Skor akan meningkat ketika obyek *Character* berjalan maju atau ketika obyek *Character* mengumpulkan obyek Candy.

Kemudian dibuat obyek *Text* kedua yang diberikan nama Label_Score dan di atur sama seperti obyek ScoreText.

Tampilan teks Skor setelah pengaturan selesai terlihat pada Gambar 7.



Gambar 7 Tampilan obyek ScoreText dan Label_Score dalam aplikasi

Akhirnya pada obyek _GameManager, masukkan obyek ScoreText ke dalam properti *Score Text* pada komponen *UI Manager*

11. Pembuatan Script Pergerakan Karakter dan Pendeteksian Gestur

Pada obyek *Character* yang sebelumnya telah dibuat, ditambahkan suatu komponen yaitu *script* *Character*KinectMovement. *Script* tersebut berfungsi untuk menggerakan obyek *Character* berdasarkan input dari sensor Kinect. *Script* tersebut dibuat dan disimpan di dalam folder Scripts pada *Assets*

Setelah membuat *script Character* Kinect Movement, berikutnya dibuat *script* GesturePartResult, Run Gesture Segments, dan RunGesture yang disimpan di tempat yang sama dengan *script* sebelumnya.

Gesture Part Result berisi *enum* yang menampung semua *state* gesture (Succeeded, Failed, Undetermined), Run Gesture Segments berfungsi untuk mendeteksi suatu bagian dari gestur (kaki kanan terangkat, kaki kiri terangkat), dan Run Gesture berfungsi untuk mendeteksi gestur lari secara keseluruhan.

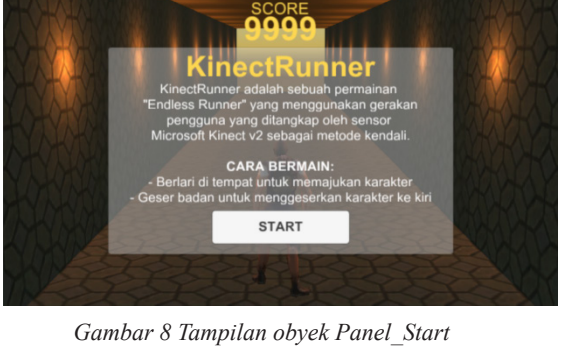
12. Pembuatan Menu Utama dan Game Over

Ketika aplikasi pertama dijalankan, sebuah menu utama yang berisi informasi cara bermain akan muncul. Begitu juga ketika permainan berakhir, sebuah menu untuk mengulangi permainan akan muncul. Untuk membuat kedua menu tersebut, ditambahkan obyek *Panel* yang kemudian ditambahkan obyek *Text*.

Kemudian untuk mengisi teks pada obyek Panel_Start, beberapa obyek *Text* dibuat yang berisi judul aplikasi dan informasi tentang aplikasi.

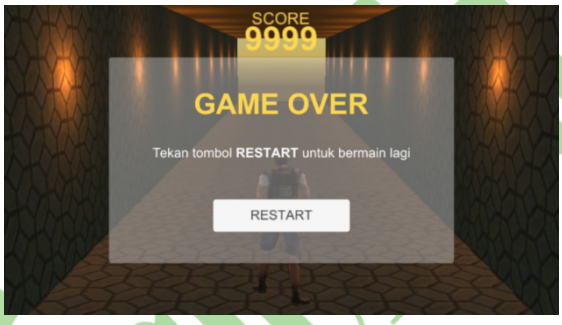
Berikutnya ditambahkan teks informasi pada obyek Panel_Start. Cara menambahkan teks informasi tersebut sama seperti penambahan obyek Label_ Title sebelumnya. Hanya saja dengan nama obyek yang berbeda, yaitu Label_Instructions. Kemudian dibutuhkan tombol yang akan menyembunyikan menu utama dan memulai permainan. Tombol tersebut dibuat menggunakan obyek *Button*.

Untuk menambahkan tombol ke dalam obyek Panel_Start, klik-kanan pada obyek Panel_Start → klik *UI* → klik *Button,* kemudian ubah nama obyek *Button* tersebut menjadi StartButton. Tampilan menu utama yang sudah selesai terlihat pada Gambar 8.



Gambar 8 Tampilan obyek Panel_Start dalam aplikasi

Akhirnya untuk membuat menu *Game Over,* caranya sama seperti pembuatan obyek Panel_Start dan pembuatan obyek *Text* sebelumnya, hanya saja dengan nama obyek yang berbeda yaitu Panel_ GameOver. Tampilan menu *Game Over* yang sudah selesai terlihat pada Gambar 9.



Gambar 9 Tampilan obyek Panel_ GameOver dalam aplikasi

13. Impor Kinect Plugin ke dalam Proyek Unity

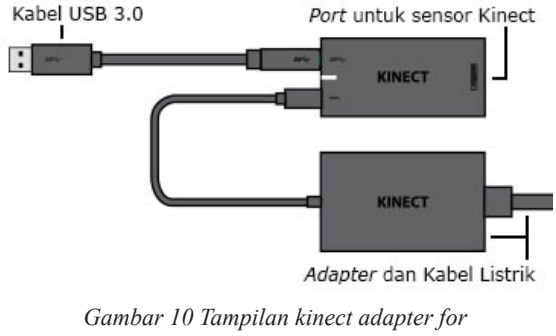
Selanjutnya dilakukan import plugin Kinect ke dalam Unity. Plugin Kinect dibutuhkan sebagai alat interaksi antara lingkungan pengembangan Unity dengan sensor Kinect sehingga Unity dapat memanfaatkan data yang masuk ke dalam sensor Kinect.

14. *Pembuatan Windows Executable*

Pembuatan *Windows Executable (EXE)* dilakukan setelah *scene* sudah berjalan dengan baik. Untuk menjalankan aplikasi pada komputer dengan sistem operasi Windows, maka dilakukanlah penyimpanan aplikasi kedalam format EXE.

15. Menghubungkan Sensor Kinect

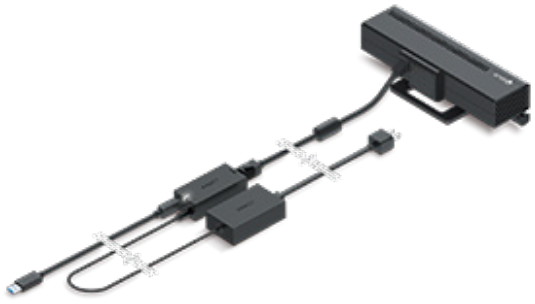
Untuk menghubungkan sensor Kinect dengan perangkat pengembangan laptop atau komputer, pertama harus mengunduh dan memasang perangkat lunak Kinect 2.0 SDK yang bersisi *driver* dan API untuk menggunakan sensor Kinect. File instalasi



Gambar 10 Tampilan kinect adapter for Windows

Kinect 2.0 SDK dapat diunduh dari situs resmi *Microsoft* pada: https://www.microsoft.com/en-us/download/details.aspx?id=44561

Setelah instalasi Kinect 2.0 SDK, sensor Kinect dihubungkan ke perangkat keras *Kinect Adapter for Windows* yang terlihat pada Gambar 10. Cara menghubungkan sensor Kinect dengan *Kinect Adapter for Windows* terlihat pada Gambar 11.



Gambar 11 Tampilan hubungan sensor kinect dengan kinect adapter for Windows

Akhirnya, sambungkan kabel USB 3.0 dari *Kinect Adapter for Windows* ke *port* USB 3.0 pada perangkat pengembangan laptop atau komputer.

Jika penghubungan berhasil, lampu indikator pada perangkat keras Kinect Adapter for Windows akan menyala putih seperti pada Gambar 12.

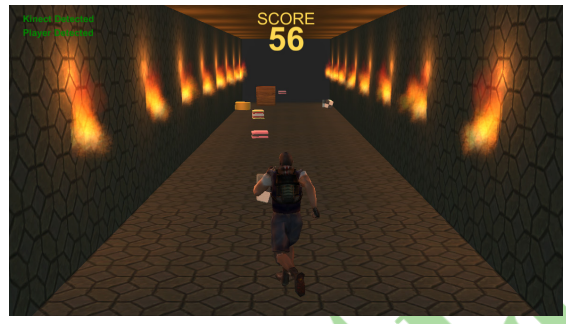
Gambar 12 Tampilan lampu indicator pada Kinect Adapter for Windows

D. Hasil Pembuatan Aplikasi

1. Tampilan Antar Muka Aplikasi

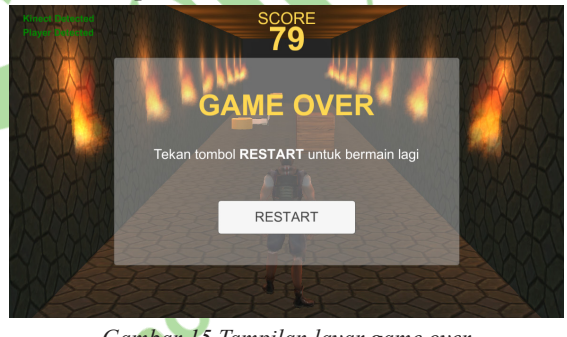
Gambar 13 Tampilan menu utama

Gambar 13 menunjukkan tampilan dari menu utama aplikasi yang pertama kali muncul ketika aplikasi dijalankan. Pada menu utama terdapat beberapa obyek yaitu nama aplikasi, informasi aplikasi, dan tombol untuk memulai permainan.



Gambar 14 Tampilan permainan

Gambar 14 menunjukkan tampilan dari aplikasi ketika permainan sedang berjalan. Pada antar muka permainan terdapat beberapa obyek yaitu indikator teks yang menunjukkan status koneksi sensor Kinect dan indikator teks yang menunjukkan status pendeteksian pemain



Gambar 15 Tampilan layar game over

Gambar 15 menunjukkan tampilan dari menu *game over* aplikasi yang muncul ketika permainan berakhir. Pada menu *game over* terdapat beberapa obyek yaitu teks *game over*, informasi *game over,* dan tombol untuk mengulang permainan.

2. Script Program Pengambilan Data dari Sensor Kinect

Selama aplikasi berjalan, pada setiap *frame* aplikasi mengambil dan menyimpan data dari sensor Kinect. Data tersebut berupa posisi dan orientasi setiap orang yang terdeteksi oleh sensor Kinect beserta posisi dan orientasi semua persendian setiap orang tersebut.

Untuk mengambil data dari sensor Kinect cukup dengan memanggil *method* GetData() dari obyek BodySourceManager. BodySourceManager sendiri adalah *class* khusus yang sudah tersedia dari *Kinect SDK 2.0* dan berisi *method* untuk mengakses fungsi sensor Kinect.

Seperti yang terlihat pada Gambar 16, *method* Kinect_Init() mengembalikan sebuah *array* obyek yang bernama data dengan jenis *Body[]*. *Body[]* adalah sebuah *array* yang berisi obyek *Body,* dimana

setiap obyek *Body* sendiri berisi data mengenai setiap tubuh manusia yang terdeteksi oleh sensor Kinect seperti posisi dan orientasi. Obyek data yang dikembalikan kemudian digunakan oleh *method* lainnya untuk menggerakan karakter.

Variabel trackedIndex adalah variabel dengan jenis *ulong* dan menampung nomor indeks obyek *Body* yang sedang diikuti pada *array* data. Penyimpanan nomor indeks ini dilakukan untuk memastikan hanya satu tubuh saja yang digunakan oleh aplikasi, yakni tubuh yang diwakili oleh obyek *Body* pada *array* data dengan nomor indeks trackedIndex. Ketika tidak ada tubuh yang diikuti oleh sensor Kinect, trackedIndex bernilai -1.

```
private Body[] kinect_Init() {
if (BodySourceManager == null) {
   return null;
}
_BodyManager = BodySourceManager.GetComponent<
BodySourceManager>();
if (_ BodyManager == null) {
   return null;
}
// Cet body data.
Body[] data = _ BodyManager.GetaData();
if (data == null) {
   return null;
}
// Check if trackedIndex is still active.
if (trackedIndex !=-1) {
   Body body = data[trackedIndex];
   if (!body.Istracked) {
         trackedIndex = -1;
   }
}
// Get new trackedIndex if not active.
if (trackedIndex == -1) {
   // Get first tracked body's index.
   for (int i = 0; i < data.lenght; i++) {
         if (data[i] == null) {
               continue;
         }
         if (data[i] == null) {
               trackedIndex = i;
               break;
         }
   }
}
return data;
}
```

Gambar 16 Script untuk mengambil Data dari sensor Kinect

3. *Script Program Pendeteksian Gestur*

Untuk mendeteksi gestur lari, gestur tersebut dibagi menjadi dua *script* bagian gesture untuk mendeteksi gestur ketika lutut kiri terangkat dan ketika liutu kanan terangkat.

Seperti yang terlihat pada Gambar 17, terdapat dua *class* bernama LeftKneeUp dan RightKneeUp

```
using Windows.Kinect;
public interface IGestureSegment {
   GesturePartResult Update(Body body);
}
public class RightKneeUp : IGestureSegment {
   public GesturePartResult Update(Body body) {
         if (body.Joints[JoinType.KneeRight].Position.Y >
body.Joints[JointType.KneeLeft]
         return GesturePartResult.Succeeded;
         else
         return GesturePartResult.Failed;
   }
}
public class LeftKneeUp : IGestureSegment {
   public GesturePartResult Update(Body body) {
         if (body.Joints[JoinType.KneeLeft].Position.Y >
body.Joints[JointType.KneeRight]
         return GesturePartResult.Succeeded;
         else
         return GesturePartResult.Failed;
   }
}
```

Gambar 17 Script untuk Deteksi Bagian Gestur

```
// Constructor.
Public RunGesture() {
      // Initialize gesture segments.
      RightKneeUp rightKneeUp = new rightKneeUp();
      LeftKneeUP LeftKneeUP = new LeftKneeUP[] {
      // Set sequence gesture segments.
      _segments = new IgestureSegment[] {
            rightKneeUp,
            leftKneeUp,
            rightKneeUp,
      };
}
```

Gambar 18 Konstruktor dari Script RunGesture

```
// Main gesture checker.
Public bool Update(Body body) {
      GesturePartResult result = _segments[_currentSegment].Update(body)
      if (result == GesturePartResult result.Succed) {
            if (_currentSegment + 1 < _segments.Lenght) {
                  _currentSegment++;
                  _timeCount = 0;
            }
            Else {
                  If (GestureRecognized != null) {
                  GestureRecognized(this, new
GestureRecognizedEventArgs(body.Trackingid));
                  Reset();
                  Return true;
            }
      }
}
Else if (_timeCount > WINDOW_TIME) {
      Reset();
}
Else {
      _timeCount += 1 * Time.deltaTime;
}
Return false;
}
```

Gambar 19 Script pendeteksian gestur lari

yang masing-masing melakukan pengecekan posisi lutut dimana jika pengecekan berhasil akan mengembalikan status Succeeded, dan jika tidak berhasil mengembalikan status Failed.

Pada Gambar 18, terlihat konstruktor dari *class* RunGesture yang menetapkan urutan bagian gestur yang harus berhasil untuk terdeteksinya gesture lari

pada *array* _segments.

Kemudian pada Gambar 19, adalah *method* pada *class* RunGesture yang dijalankan pada tiap frame dan smelakukan pengecekan terhadap bagian gesture yang berhasil terdeteksi.

4. Script Program untuk Menggerakan Karakter

Untuk menggeser karakter ke kiri dan kanan, pada setiap *frame* dilakukan pengecekan posisi pemain, kemudian posisi karakter dalam permainan disamakan dengan posisi pemain tersebut.

Seperti yang terlihat pada Gambar 20, posisi rusuk pemain disimpan ke dalam variabel xPos. Kemudian setelah variabel tersebut diberi batas minimum dan maksimum, posisi baru yang tersimpan dalam variabel xPos diterapkan pada posisi karakter.

```
// Set horizontal position.
Float xpos = data[trackedIndex].Joints[JointType.SpineMid].Position.X * 9;
Transform.position = new Vector3(Mathf.Clamp(xpos, -3, 3), transform.position.y,
transform.position.z);
```

Gambar 20 Script untuk menggeser karakter

Untuk mendorong karakter maju, ditentukan sebuah batas waktu pendeteksian gestur. Cara kerjanya adalah jika pemain tidak melakukan gestur sebelum batas waktu tercapai, karakter akan berhenti. Selama pemain terus berlari, karakter akan terus maju. Seperti yang terlihat pada Gambar 21, *method* Gesture_GestureDetect() mengembalikan status pendeteksian gestur lari. Jika gestur lari berhasil terdeteksi, maka *timer* batas waktu akan diulang. Jika gestur lari gagal terdeteksi, *timer* batas waktu akan berkurang tiap detik.

Selama *timer* batas waktu lebih dari nol, karakter akan maju dengan kecepatan yang telah ditentukan.

```
// Reset timer if gesture detected, decrement otherwise.
Bool? gestureDetected = Gesture_ gestureDetected(data)
If (gestureDetected == null) {
        Return;
}
Else if (gestureDetected == true) {
        gestureTimer = gestureTimeout;
}
Else {
        If (gestureTimer > -1)
                gestureTimer = 1* Time.deltaTime; Debug.Log("Timer:" +
gestureTimer);
}
// Apply forwad movement if timer > 0
If (gestureTimer > 0) {
        // Get character local forward direction.
        moveDirection = transform.forward;
        // Transform local forward into world forward.
        moveDirtection = transform.TransformDirection(moveDirection);
        //Multiply by Speed.
        // Increase score when running.
        UIManager.Instance.IncreaseScore(scorePersecond * Time.deltaTime
}
Else {
        If (moveDirection.z > 0) {
                moveDirection = Vector3.Lerp(moveDirection, new
Vector3(moveDirection.x, moveDirection.y, 0), Speed);
        }
}
```

Gambar 21 Script untuk memajukan karakter

E. Evaluasi

1. Evaluasi Pengujian Koneksi Kinect dengan Deteksi Pemain

Pada layar aplikasi terdapat dua indikator teks yang menunjukkan keadaan koneksi sensor Kinect dan keadaan deteksi pemain Kedua indikator tersebut berjalan dengan baik dan menunjukkan dengan jelas keadaan sensor Kinect dan pemain.

2. Evaluasi Pengujian Menu Utama

Pada menu utama terdapat sebuah tombol untuk memulai permainan yang hanya berfungsi jika sensor Kinect dan pemain terdeteksi. Tombol akan non-aktif dan berwarna abu-abu seperti jika sensor Kinect dan pemain tidak terdeteksi. Tombol tersebut berjalan dengan baik dan berfungsi sesuai rancangan yang sudah dibuat.

3. Evaluasi Pengujian Skeletal Tracking

Saat permainan dimulai, pemain dapat menggeser karakter ke kiri dan ke kanan menggunakan alat input sensor Kinect. Pergerakan karakter sudah sesuai dengan input yang diinginkan.

4. Evaluasi Pengujian Gesture Recognition

Saat permainan dimulai, pemain dapat berlari di tempat untuk membuat karakter maju ke depan menggunakan alat input sensor Kinect. Pergerakan karakter sudah sesuai dengan input yang diinginkan.

5. Evaluasi Pengujian Skor

Saat permainan berjalan, jika pemain berlari di tempat atau menyentuh salah satu obyek skor, tampilan skor pada layar aplikasi akan bertambah. Kondisi tersebut berjalan dengan baik dan berfungsi sesuai rancangan yang dibuat.

6. Evaluasi Pengujian Kondisi Game Over

Saat permainan berjalan, jika pemain tidak berlari di tempat untuk waktu tertentu atau menabrak salah satu obyek rintangan, permainan akan berakhir dan menu *Game Over* akan muncul. Kondisi tersebut berjalan dengan baik dan berfungsi sesuai rancangan yang dibuat.

7. Evaluasi Pengujian Menu Game Over

Pada menu *Game Over* terdapat sebuah tombol untuk mengulangi permainan. Tombol tersebut berjalan dengan baik dan berfungsi sesuai rancangan yang dibuat.

IV. SIMPULAN

Berdasarkan hasil implementasi dan pengujian pada penelitian ini, maka dapat disimpulkan sebagai berikut: (1) Proses integrasi sensor Kinect dengan aplikasi permainan cukup sulit karena sedikitnya

informasi dan panduan yang jelas tentang cara menggunakan sensor Kinect dengan *Unity3D* secara efektif; (2) Aplikasi permainan lari dapat berjalan dengan baik, pemain dapat mengumpulkan skor dan permainan berakhir jika pemain menabrak rintangan atau berhenti terlalu lama. Namun, umpan balik dari mengumpulkan skor dan menabrak rintangan hanya berupa suara dan tanpa efek visual yang jelas; (3) Metode interaksi dengan tombol pada menu menggunakan *mouse,* sedangkan berinteraksi dengan permainan menggunakan gerakan tubuh. Menggunakan dua metode interaksi tersebut secara bergantian cukup sulit; dan (4) dan tidak efisien, karena pengguna harus keluar jarak pandang sensor Kinect untuk menggunakan *mouse* demi memulai permainan dan kembali masuk lagi kedalam jarak pandang sensor.

V. DAFTAR RUJUKAN

[1] L. L. D. Rosen, et al. "Media and technology use predicts ill-being among children, preteens and teenagers independent of the negative health impacts of exercise and eating habits," National Center for Biotechnology Information, U.S. National Library of Medicine, 23 Februari 2015. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4338000/. [Accessed 18 Desember 2015].

[2] D. M. Dastbaz, Designing Interactive Multimedia Systems, Singapore: McGraw-Hill Education, 2003.

[3] "Apa itu Multimedia," Satria Multimedia, [Online]. Available: http://www.satriamultimedia.com/artikel_apa_itu_multimedia.html. [Accessed 20 April 2016].

[4] A. T. Wibowo, "Teknologi Natural User Interface Menggunakan Kinect Sebagai Pemicu Kerja Perangkat Keras Berbasis Web," [Online]. Available: http://sir.stikom.edu/516/1/2012-ICCS-19.pdf. [Accessed 18 Desember 2015].

[5] Hartono, et al. "Pendeteksian Gerak Menggunakan Sensor Kinect for Windows," 2015. [Online]. Available: http://studentjournal.petra.ac.id/index.php/teknik-informatika/article/viewFile/3703/3371. [Accessed 18 Desember 2015].

[6] D. Nawanda, "Apa itu Natural User Interface?," Unite UX, 21 July 2014. [Online]. Available: http://uniteux.com/apa-itu-natural-user-interface/. [Accessed 15 January 2015].

[7] E. Sanders, et al. "GAME 3D NO WAY OUT DENGAN FITUR VIRTUAL REALITY," Seminar Nasional Teknologi Informasi 2015, 2015.

[8] M. W. Grivin, "GAME ENGINE," [Online]. Available: http://www.academia.edu/6220575/GAME_ENGINE. [Accessed 12 March 2016].

[9] R. M. Y. &. Aristiawan, "Unity 3D - Game Engine," HermanClass, 6 October 2013. [Online]. Available: http://www.hermantolle.com/class/docs/unity-3d-game-engine/. [Accessed 21 March 2016].

[10] "Unity-Store," Unity Technologies, [Online]. Available: https://store.unity.com/. [Accessed 21 March 2016].

[11] "Platform Game," Wikimedia Foundation, [Online]. Available: https://en.wikipedia.org/wiki/Platform_game#Endless_running_game. [Accessed 16 July 2016].