Game Arcade Adaptive Menggunakan Unreal Engine 4

Mukhamad Subkhan
Prodi Teknik Informatika, Fakultas Teknik, Universitas Muhammadiyah Cirebon

Abstraksi

Game merupakan sebuah sistem yang menyediakan beberapa aturan atau rule yang disediakan untuk pemain, pemain harus mengikuti aturan game untuk mendapatkan beberapa kemungkinan antara menang atau kalah. Game berfungsi untuk melatih kecerdasan motorik dan reflek pemain, selain itu game menjadi sarana yang cukup efektif untuk menghilangkan stress.

Pemanfaatan teknologi saat ini sangat minim, maka dari itu penulis ingin membuat game sebagai salah satu cara untuk memanfaatkan teknologi secara positif. Sebagian besar game yang beredar di pasaran mengandung hal-hal negatif berupa kekerasan, konten dewasa dan lain-lain, maka dari itu penulis ingin membuat game yang aman untuk dimainkan semua umur.

Game yang dibuat adalah game arcade adaptive menggunakan Unreal Engine 4, bahasa pemrograman yang dipakai adalah Kismet atau bahasa pemrograman yang disediakan oleh Unreal Engine. Teknik pengumpulan data yang dilakukan dengan cara studi literatur, sedangkan metode yang digunakan adalah prototyping.

Game arcade adaptive ini dapat beradaptasi terhadap waktu yang dihabiskan player di dalam game, sistem akan menambah rintangan, kecepatan berlari dan damage seiring berjalannya waktu dan keadaan player.

Kata Kunci : Game, Arcade, Adaptive, Unreal Engine 4, Game Arcade Adaptive, Game Arcade Adaptive Menggunakan Unreal Engine 4.

PENDAHULUAN

A. Latar Belakang

Seiring berkembangnya zaman, teknologi di Dunia ini semakin berkembang pesat, salah satunya dalam industri *game*, setiap bulan, minggu bahkan hari *game* selalu muncul dan diperbarui. Berbagai tipe dan genre *game* yang dihadirkan selain itu harga dan spesifikasi *game* tersebut banyak ditawarkan oleh masing-masing *developer game* tersebut. Banyak sekali masyarakat awam yang hanya menganggap *game* sebagai sarana yang menyesatkan anak kedalam dunia anarkis dan unsur-unsur negatif, tapi developer *game* sudah mematokkan genre dan syarat berupa batas usia yang membolehkan orang untuk memainkan *game* tersebut. Jika kita teliti lebih dalam lagi *game* mempunyai hal-hal yang positif, di antaranya melatih kesabaran, melatih kecerdasan motorik, mengatur siasat dan lain-lain. Mendengarkan pernyataan di atas, maka penulis ingin membuat penililtian mengenai *Game Arcade Adaptive* yang bisa melatih kecerdasan motorik seseorang.

B. Identifikasi Masalah

Dari uraian Latar Belakang Masalah di atas, penulis dapat mengidentifikasikan masalah sebagai berikut:

1. Kurangnya pemanfaatan teknologi yang sangat pesat secara positif.
2. Diperlukannya *game engine* yang dapat mempermudah proses pembuatan *game*.
3. Diperlukan *game* yang dapat melatih kecerdasan motorik serta mampu beradaptasi dengan kecerdasan *player*.

C. Rumusan Masalah

Dari identifikasi masalah yang ada, maka penulis merumuskan masalah, sebagai berikut :

1. Bagaimana cara yang tepat untuk memanfaatkan teknologi yang sangat pesat dengan cara yang positif?
2. Bagaimana *Unreal engine* dapat mempermudah pembuatan *game*?
3. Bagaimana *game* dapat melatih kecerdasan motorik, dan mampu beradaptasi dengan kecerdasan *player*?

D. Batasan Masalah

Supaya pembahasan masalah yang dilakukan dapat terarah dengan baik dan tidak menyimpang dari pokok permasalahan, maka penulis membatasi permasalahan yang akan dibahas, yakni:

1. *Game* yang dibuat tidak mempunyai *story line*.
2. Tidak dibuatkan *save state*, atau melanjutkan *game* yang belum selesai dimainkan.
3. Hanya dapat dimainkan oleh 1 *Player*.
4. Dibutuhkan spesifikasi komputer yang cukup tinggi untuk memainkan *game* tersebut.
5. Tidak dapat menyimpan skor tertinggi.

E. Maksud dan Tujuan Penelitian

1. Maksud Penelitian

Berdasarkan permasalahan yang diteltiti, maka maksud dari penulisan adalah.

a. Mempelajari cara membuat *game*.
b. Mempelajari unsur yang ada dalam *game*.
c. Mempelajari *software* yang akan digunakan dalam pembuatan *game*.
d. Membuat *game arcade adaptive* yang dapat melatih kecerdasan motorik.

2. Tujuan Penelitian

Tujuan yang hendak dicapai dalam perancangan sistem ini adalah sebagai berikut:

a. Mampu membuat *game* sederhana.
b. Menguasai unsur di dalam *game*.
c. Menguasai *software* pembuatan *game* dan mempermudah dalam penggunaannya.
d. Menghasilkan *game* yang sesuai dengan yang direncanakan.

F. Manfaat Penelitian

1. Bagi Penulis

Memberikan pengalaman bagi penulis dalam merancang, membuat *game arcade* menggunakan *unreal engine* serta dapat mengaplikasikan ilmu yang telah ditempuh selama perkuliahan sebagai syarat memperoleh gelar sarjana.

2. Bagi Perindustrian Game dan Animasi Indonesia

Mengharumkan Negara Indonseia dengan prestasi dari genrasi muda yang kreatif, dan membuktikan bahwa Indonesia bisa membuat *game* yang tidak kalah bagusnya dengan *game* buatan Negara lain.

3. Bagi Biro Administrasi Umum

Sebagai dokumen dan referensi Universitas Muhammadiyah Cirebon guna menunjang proses perkuliahan nantinya dan juga menumbuh kembangkan minat mahasiswa Universitas Muhammadiyah Cirebon, serta dengan menciptakan *game arcade adaptive* menggunakan *unreal engine* ini, saya dapat membantu mengharumkan citra Universitas Muhammadiyah Cirebon supaya terlihat lebih baik di masyarakat. Serta membuktikan bahwa lulusan Universitas Muhammadiyah Cirebon dapat bersaing dengan mahasiswa dari Universitas lainnya yang ada di Indonesia pada umumnya, dan Cirebon khususnya.

G. Metode Penelitian

1. Metode Pengumpulan Data

Metode pengumpulan data yang dilakukan dalam penelitian ini dilakukan dengan cara, Pengumpulan data dengan menggunakan atau mengumpulkan sumber-sumber tertulis, dengan cara membaca, mempelajari dan mencatat hal-hal penting yang berhubungan dengan masalah yang sedang dibahas guna memperoleh gambaran secara teoritis.

2. Metode Pengembangan Perangkat Lunak

Metode yang digunakan untuk membangun sistem ini adalah *Prototyping*. *Prototyping* merupakan salah satu metode pengembangan perangat lunak yang banyak digunakan. Dengan metode *prototyping* ini pengembang dan pelanggan dapat saling berinteraksi selama proses pembuatan sistem., dengan beberapa tahapan, yaitu:

a. Pengumpulan kebutuhan

Pelanggan dan pengembang bersama-sama mendefinisikan format seluruh perangkat lunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.

b. Membangun *prototyping*

Membangun *prototyping* dengan membuat perancangan sementara yang berfokus pada penyajian kepada pelanggan (misalnya dengan membuat *input* dan *format output*)

c. Evaluasi *protoptyping*

Evaluasi ini dilakukan oleh pelanggan apakah *prototyping* yang sudah dibangun sudah sesuai dengan keinginann pelanggan. Jika sudah

sesuai maka langkah 4 akan diambil. Jika tidak *prototyping* direvisi dengan mengulangu langkah 1, 2 , dan 3.

d. Mengkodekan system

Dalam tahap ini *prototyping* yang sudah disepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai

e. Menguji system

Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus diuji terlebih dahulu sebelum digunakan. Pengujian ini dilakukan dengan *White Box*, *Black Box*, Basis *Path*, pengujian arsitektur dan lain-lain

f. Evaluasi Sistem

Pelanggan mengevaluasi apakah sistem yang sudah jadi sudah sesuai dengan yang diharapkan, jika ya, langkah 7 dilakukan; jika tidak, ulangi langkah 4 dan 5.

g. Menggunakan system

Perangkat lunak yang telah diuji dan diterima pelanggan siap untuk digunakan.

H. Jadwal Penelitian

Penelitian dilakukan pada tanggal 01 Oktober sampai dengan 29 Februari 2016.

Tabel 1.1 Jadwal Penelitian

No.	Jenis Kegiatan	Bulan Ke -			
		1	2	3	4
1	*Pengumpulan Kebutuhan*				
2	*Membangun Prototyping*				
3	*Evaluasi Prototyping*				
4	*Mengkodekan Sistem*				
5	*Menguji Sistem*				
6	*Evaluasi Sistem*				
7	*Menggunakan Sistem*				

ANALISIS DAN PERANCANGAN SISTEM

A. Analisis Sistem

Game yang akan dibuat bergenre *Arcade* yang bertujuan untuk mendapatkan skor sebanyak-banyaknya, skor tersebut didapatkan dari jumlah koin yang berhasil terkumpul, koin tersebut tersebar secara acak di arena *game* dan *player* bertugas untuk melalui koin tersebut untuk diambil.

Selain itu, *game* ini mampu beradaptasi dengan kecerdasan *player*, jika *player* dapat melewati rintangan 1 maka sistem akan menambahkan tingkat kesulitan, diantaranya lebih sering dimunculkannya rintangan, penambahan *damage*, dan juga menambahkan *movement speed* setiap detiknya.

Game Engine yang digunakan untuk pembuatan *game* menggunakan *Unreal Engine 4*, yang berfungsi untuk pembuatan arena *game*, animasi, dan pembuatan perintah-perintah komputer berupa *blueprInt* yang di dalamnya terdapat bahasa komputer *kismet* (mesin *scripting visual* pada UE3) untuk membuat *game*.

Output dari sistem ini adalah berupa *game* 3D dengan *gameplay arcade* yaitu *player* bertugas melalui rintangan dan mengambil sejumlah koin yang terdapat pada arena *game* yang kemudian akan dikalkulasikan menjadi skor akhir.

B. Deskripsi Sistem

Game yang akan dibuat merupakan *game* bergenre *Arcade*, yang artinya *player* harus melewati berbagai rintangan yang terdapat pada arena *game*, selain tu *player* bertugas untuk mengumpulkan koin sebanyak-banyaknya untuk dikumpulkan dan dijadikan skor akhir, *player* diberikan *HP* (*Health point*) untuk indikator jumlah energi *player*, *HP* dapat berkurang ataupun bertambah sesuai situasi yang terjadi saat berada di arena *game*, *HP* dapat berkurang jika *player* menabrak sebuah batu, atau sistem secara otomatis mengurangi beberapa persen *HP* setiap detiknya, dan *HP* dapat bertambah dengan m*end*apatkan koin besar di arena *game*.

Rintangan yang terdapat dalam *game Arcade* yang akan dibuat yaitu berupa batu, tembok, jembatan, dan jurang, jika *player* menabrak batu maka beberapa persen *HP* dikurangi, dan jika *player* terjatuh kedalam jurang maka *player* akan mati, dan memuat ulang atau (*restart level*), selain itu jika *player* menabrak tembok atau gagal berbelok maka *player* akan mati dan sistem akan melakukan *restart level*..

Karena *game* ini tidak mempunyai *story line* maka alur *game* yang dihadirkan akan terus berulang secara acak, maka tujuan utama dari *game* ini yaitu mengumpulkan koin sebanyak-banyaknya untuk m*end*apatkan skor.

C. Perancangan Sistem

1. *Use Case* Diagram

a. Definisi Aktor

Definisi Aktor berfungsi untuk menjelaskan peranan aktor yang merupakan unsur penting yang terdapat dalam *game*. Definisi tersebut dapat dilihat pada tabel 4.1 berikut :

Tabel 4.1 Definisi Aktor

No.	Aktor	Deskripsi
1.	*Player*	Merupakan aktor yang berperan dalam mengontrol karakter di antaranya berbelok kekanan, berbelok kekiri, dan melompat

b. Definisi *Use Case*

Definisi *Use Case* berfungsi sebagai penjelasan mengenai proses yang terdapat pada setiap *Use Case*. Dapat dilihat definisi tersebut pada tabel 4.2 berikut :

Tabel 4.2 Definisi *Use Case*

No.	Nama *Use Case*	Deskripsi
1.	*Start Game*	*Player* memulai untuk memainkan *game*.
2.	*Running*	Sistem membuat karakter terus berlari.
3.	Belok	*Player* dapat membelokkan karakter untuk menghindari rintangan atau mengumpulkan koin
4.	*Turn corner*	*Player* dapat membelokkan karakter 90 derajat untuk beralih ke arena sebelah kanan tau kiri untuk menghindari tabrakan.
5.	Lompat	*Player* dapat melompat dari satu arena kearena yang berada di seberang.
6.	M*end*apatkan Koin	*Player* mengumpulkan koin untuk dijadikan skor.
7.	M*end*apatkan koin besar	*Player* m*end*apatkan ektra koin dan tambahan *HP*.
8.	*Spawn Floor*	Beralih kelantai selanjutnya setelah melewati salah satu lantai.
9.	*Spawn Coin*	Sistem memanggil koin secara acak dengan menggunakan *coin* area *(box collision)*.
10.	*Spawn* Blocker	Sistem memanggil batu pada salah satu dari 3 *arrow* yang telah ditempatkan pada *FloorTile*
11.	*Spawn Bridge*	Sistem memanggil sebuah jembatan *(shape_cube)* pada salah satu dari *arrow* yang ditempatkan pada *FloorBridge*
12.	Skor	Sistem menampilkan skor pada layar
13.	Menambahkan *Damage*	Sistem menambahkan *damage* awal ditambah *damage* tambahan setiap musuh menabrak blocker
14.	*Timer*	Sistem memberikan waktu bermain untuk *player* dengen menggunakan *HP*.
15.	Menambahkan Tingkat Kesulitan	Sistem akan sering memunculkan rintangan jika *player* dianggap pintar memainkan *game*.
16.	Menambah Kecepatan Berlari	Sistem menambahkan *value movement walk speed* setiap detiknya

c. Skenario Umum

1. *Player* selaku aktor utama bertugas untuk mengontrol karakter, untuk berbelok, dan melompat.
2. *Player* dapat berbelok 90 derajat ke kanan atau kekiri setelah menginjak *turn* area *(box collision)*

3. *Player* m*end*apatkan koin kecil *(static mesh)* dan akan dikalikan satu untuk dijadikan skor, dan akan ditampilkan melalui *HUD Widget*.
4. *Player* m*end*apatkan koin besar untuk *(static mesh)* m*end*apatkan tambahan 20 skor dan memulihkan 30% *HP*.
5. Saat *player* melalui *end trigger (trigger* box*)* maka Sistem memanggil lantai selanjutnya secara *random*.
6. Sistem mengurangi 1% *HP* setiap detiknya sebagai energi yang terpakai untuk berlari.
7. *HP player* akan berkurang jika menabrak sebuah batu.
8. *Player* akan mati jika menabrak sebuah dinding atau terjatuh kejurang.
9. Sistem akan memanggil 1 *static mesh* (batu atau jembatan) pada 3 *arrow* yang tempatnya berbeda.
10. Sistem akan melakukan *restart level* jika *player* mati.
11. Sistem akan memanggil static mesh (koin kecil dan koin besar) pada *coin* area *(box collision)*.
12. Sistem akan menampilkan skor di layar.
13. Sistem menambahkan *damage* setiap *player* menabrak *blocker*.
14. Sistem menggunakan *HP* untuk dijadikan *timer*, setiap detiknya *HP* akan dikurangi sebesar 0.001
15. Sistem menambahkan tingkat kesulitan saat *player* berhasil melewati rintangan 1.Sistem menambahkan *value walk speed* setiap detiknya.

d. Skenario *Use Case*

Skenario pada setiap bagian *Use Case* yang menunjukkan penjelasan setiap bagian-bagian di dalam *Use Case* tersebut.

Tabel 4.3 Skenario *Use Case* 1. *Start Game*

Identifikasi	
Nama Use Case	*Start Game*
Aktor	*Player*
Tujuan	Memulai *Game*
Keadaan Awal	*Player* menekan tombol *play*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* menekan tombol *play* dan otomatis *game* mulai diputar	
	2. Sistem menampilkan arena *game (Floor)* secara acak dengan *function AddFloorTile.*
3. Selesai	
Kondisi akhir	*Player* berada di arena *game*

Tabel 4.4 Skenario *Use Case* 2. *Running*

Identifikasi	
Nama Use Case	*Running*
Aktor	*Player*
Tujuan	Berlari di arena *game*
Keadaan Awal	*Player* telah berada di arena *game*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* menginjak lantai	
	2. Sistem mengeksekusi perintah *event thick* dan dihubungkan pada *function add movement input* yang artinya *player* akan terus bergerak kedepan jika tidak terkena hambatan.
3. Selesai.	
Kondisi akhir	4. *Player* terus bergerak kedepan.
Skenario Alternatif – *Input* Gagal	
Aksi Aktor	**Reaksi Sistem**
1. *Player* meabrak *EndwallTrigger,*	

atau *GroundTrigger*.	
	2. Sistem mengeksekusi perintah *custom event Death*
	3. Sistem melakukan *Restart Level* pada *function Execute Console Command*

Tabel 4.5 Skenario *Use Case* 3. Belok

Identifikasi	
Nama Use Case	Belok
Aktor	*Player*
Tujuan	Mengarahkan karakter kekanan atau kekiri
Keadaan Awal	*Player Running*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player Running*	
2. *Player* menekan tombol A atau D pada *event InputAxis MoveRight*	
	3. Sistem mengeksekusi perintah untuk mengubah arah gerakan *player* pada *function get right vector* yang dihubungkan pada *function movement input.*
4. Selesai	
Kondisi akhir	5. Karakter bergerak kekanan atau kekiri

Tabel 4.6 Skenario *Use Case* 4. *Turn Corner*

Identifikasi	
Nama Use Case	*Turn corner*
Aktor	*Player*
Tujuan	Berbelok 90 derajat kearah kanan atau kiri
Keadaan Awal	*Player* berada pada *TurnZone (Collision Box)*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* masuk kedalam *TurnZone (Collision Box)*	
2. *Player* Menekan tombol A atau D (memberikan perintah *InputAxis MoveRight)*	
	3. Sistem mengeksekusi perintah dengan menggunakan *function Combine Rotator* yang telah di*input*kan 90 pada Z dan -90 pada Z, yang artinya karakter akan berbelok 90 derajat kearah kanan atau kiri.
4. Selesai	
Kondisi akhir	5. *Player* Berbelok 90 derajat kearah kanan atau kiri sesuai *input*.

Tabel 4.7 Skenario *Use Case* 5. Lompat

Identifikasi	
Nama Use Case	Lompat
Aktor	*Player*
Tujuan	Melompat
Keadaan Awal	*Player Running*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**

1. *Player Running*.	
2. *Player* menekan tomol *Space* atau memberikan perintah *Input Action Jump*.	
	3. Sistem mengeksekusi perintah melompat dengan menggunkan *function Jump*.
4. Selesai.	
Kondisi akhir	*Player* melompat.

Tabel 4.8 Skenario *Use Case* 6. Mendapat Koin

Identifikasi	
Nama Use Case	M*end*apatkan Koin
Aktor	*Player*
Tujuan	M*end*apatkan Skor
Keadaan Awal	*Player* menabrak *Coin (Static Mesh)*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* menabrak *Coin (Static Mesh)*	
	2. Sistem menghitung Skor dengn menggunakan *Function Add Coin* dan disimpan kedalam *variable total coin*
	3. Sistem menghilangkan *Coin (Static Mesh)* melalui *function DestroyActor*
	4. Sistem menampilkan skor pada *RunHUD* yang telah terhubung dengan *Variable Total Coin*
5. *Player* dapat melihat skor pada *HUD*	
6. Selesai	
Kondisi akhir	7. *Player* m*end*apatkan skor sementara.

Tabel 4.9 Skenario *Use Case* 7. Mendapatkan Koin Besar

Identifikasi	
Nama Use Case	M*end*apatkan Koin Besar
Aktor	*Player*
Tujuan	M*end*apatkan ekstra skor dan *HP regen*
Keadaan Awal	*Player* menabrak *BigCoin (Static Mesh)*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* menabrak *BigCoin (Static Mesh)*	
	2. Sistem menghitung Skor dengn menggunakan *Function Add BigCoin* dan disimpan kedalam *variable total coin*
	3. Sistem menghilangkan *BigCoin (Static Mesh)* melalui *function DestroyActor*
	4. Sistem menampilkan skor pada *RunHUD* yang telah terhubung

	dengan *Variable Total Coin*
5. *Player* dapat melihat skor pada *HUD*	
6. Selesai	
Kondisi akhir	7. *Player* m*end*apatkan ekstra skor dan 0.3 *HP Regen*

Tabel 4.10 Skenario *Use Case* 8. *Spawn Floor*

Identifikasi	
Nama Use Case	*Spawn Floor*
Aktor	*Player*
Tujuan	Memanggil lantai selanjutnya
Keadaan Awal	*Player* menabrak *EndTrigger (Collision Box)*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* berjalan melalui *EndTrigger (Collision Box)*	
	2. Sistem mengeksekusi *CustomEvent AddFloorTile* yang berfungsi untuk memanggil lantai selanjutnya pada *Floor Tiles* secara acak.
3. *Player* melewati *End Trigger (Collision Box)* 7 kali	
	4. Sistem mengeksekusi *function Add FloorCurve* yang berfungsi untuk memanggil lantai dari *Floor Curves secara acak.*
5. Selesai.	
Kondisi akhir	6. Sistem memanggil *Floor* yang lain.

Tabel 4.11 Skenario *Use Case* 9. *Spawn Coin*

Identifikasi	
Nama Use Case	*Spawn Coin*
Aktor	*Player*
Tujuan	Memanggil Koin
Keadaan Awal	*Player* menabrak *EndTrigger (Collision Box)*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* menabrak *EndTrigger (Collision Box)*	
	2. Sistem merlakukan *Spawn Floor Tile*, yang berfungsi untuk memnggil *floor tile*.
	3. Sistem memanggil 10 koin yang ditempatkan di atas *CoinArea (Collion* Box*)* secara *random* melalui *Function Spawn Coin*
4. Selesai.	
Kondisi akhir	5. Koin ada di atas *coin area*

Tabel 4.12 Skenario *Use Case* 10. *Spawn Blocker*

Identifikasi	
Nama Use Case	*Spawn Blocker*
Aktor	*Player*
Tujuan	Memanggil *Blocker (Static Mesh)*
Keadaan Awal	*Player* menabrak *EndTrigger (Collision Box)*

Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* menabrak *EndTrigger (Collision Box)*	
	2. Sistem merlakukan *Spawn Floor Tile*, yang berfungsi untuk memnggil *floor tile*.
	3. Sistem memanggil 1 Blocker yang ditempatkan di antara 3 *SpawnPoInt(Arrow)* yang berbeda secara acak melalui *function SetSpawnPoInt*dan *Function Random Integer In Range.*
4. Selesai.	
Kondisi akhir	5. Blocker Terpanggil

Tabel 4.13 Skenario *Use Case* 11. *Spawn Bridge*

Identifikasi	
Nama Use Case	*Spawn Bridge*
Aktor	*Player*
Tujuan	Memanggil *Bridge (Static Mesh)*
Keadaan Awal	*Player* menabrak *EndTrigger (Collision Box)*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* menabrak *EndTrigger (Collision Box)*	
	2. Sistem merlakukan *Spawn Floor Curve*, yang berfungsi untuk memnggil *FloorCurve*.
	3. Sistem memanggil 1 *Bridge (StaticMesh)* yang ditempatkan di antara 3 *SpawnPoInt(Arrow)* yang berbeda secara acak melalui *function SetSpawnPoInt*dan *Function Random Integer In Range.*
4. Selesai.	
Kondisi akhir	5. *Bridge* Terpanggil

Tabel 4.14 Skenario *Use Case* 12. Skor

Identifikasi	
Nama Use Case	Skor
Aktor	*Player*
Tujuan	Menghitung, menampilkan Skor di layar
Keadaan Awal	*Player* m*end*apatkan *Coin* dan *BigCoin*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* menabrak *Coin* dan *BigCoin*	
	2. Sistem mengkalkulasikan koin tersebut kedalam bentuk *integer+integer*, dengan *coin* + 1, dan *bigcoin* + 30.
	3. Sistem menampilkan skor pada *HUD* yang terhubung melalu *variable integer* dari masing-masing jenis koin
	4. Sistem mereset *level* dan melakukan eksekusi dari *function Execute Console*

	Command "ResetLevel"
5. Selesai.	
Kondisi akhir	6. Didapatkan hasil skor

Tabel 4.15 Skenario *Use Case* 13. Menambahkan *Damage*

Identifikasi	
Nama Use Case	Menambahkan *Damage*
Aktor	*Player*
Tujuan	Menambahkan *value damage*
Keadaan Awal	*Player* melewati *Blocker*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* melewati *Blocker*	
	2. Sistem mengkalkulasikan *value damage* awal dengan ditambahkan *value damage* baru.
	3. Sistem menyimpan *value damage* baru.
	4. Sistem mengurangi *value Current Health* sesuai dengan *damage* baru.
5. Selesai.	
Kondisi akhir	6. *Value Damage* ditambah.

Tabel 4.16 Skenario *Use Case* 14. *Timer*

Identifikasi	
Nama Use Case	*Timer*
Aktor	*Player*
Tujuan	Mengurangi *HP player* per detik
Keadaan Awal	*Player* memulai *game*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* memulai *game*	
	2. Sistem menghitung *value Current* helath – *drain health* setap detiknya, dengan menggunakan *function get delta seccond*
3. Selesai.	
Kondisi akhir	4. *Value Current Health* berkurang.

Tabel 4.17 Skenario *Use Case* 15. Meningkatkan Tingkat Kesulitan

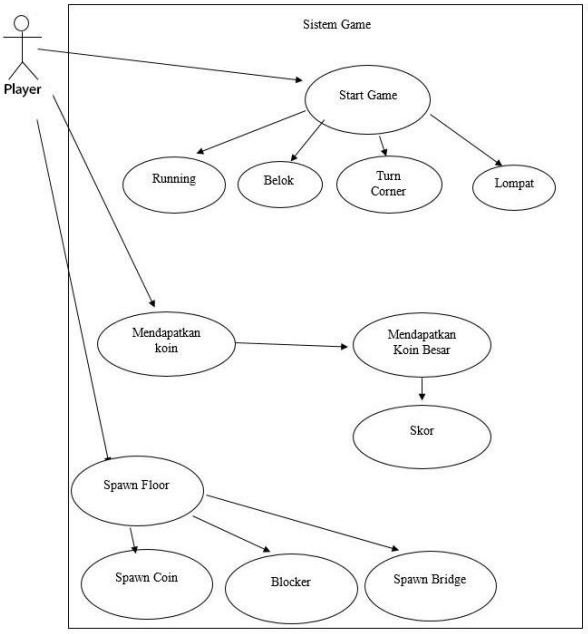
Identifikasi	
Nama Use Case	Meningkatkan Tingkat Kesulitan
Aktor	*Player*
Tujuan	Mengurangi jumlah *Current Straight*
Keadaan Awal	*Player* melewati *Floor Curve*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* melewati *Floor Curve*	
	2. Sistem mengurangi *value Current Straight* dengan *value* awal 20 dikurangi 1 *value*
	3. Sistem membatasi minimal *Current Straight* dengan menggunakan *Variable Min Straight PoInt*sebesar 3 *value*.
4. Selesai	

Kondisi akhir	5. *Value Current Straight* berkurang.

Tabel 4.18 Skenario *Use Case* 16. Menambah Kecepatan Berlari

Identifikasi	
Nama Use Case	Menambah Kecepatan Berlari
Aktor	*Player*
Tujuan	Menambah *Value Walk Speed*
Keadaan Awal	*Player* memulai *game*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. *Player* memulai *game*	
	2. Sistem menghitung *value Max Walk Speed* + 1 *value* setap detiknya, dengan menggunakan *function get delta seccond*
3. Selesai	
Kondisi akhir	4. *Value Max Walk Speed* bertambah 1 setiap detiknya.

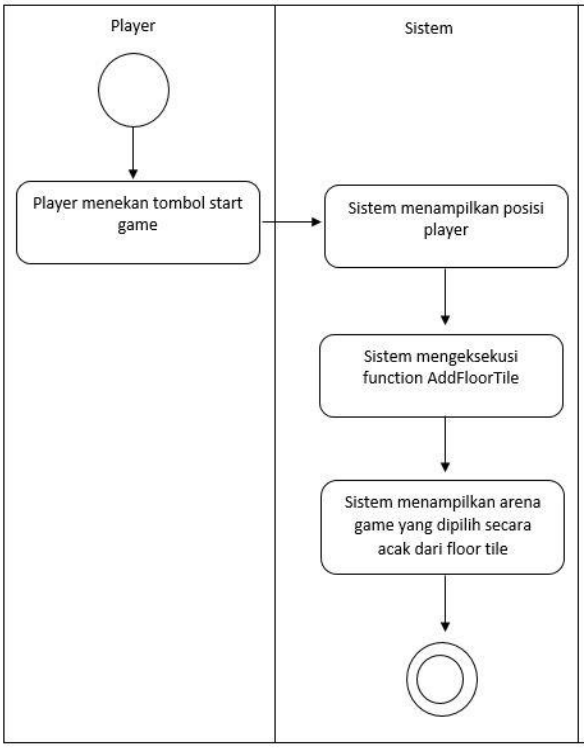
e. *Use Case* Diagram



Gambar 4.1 *Use Case* Diagram

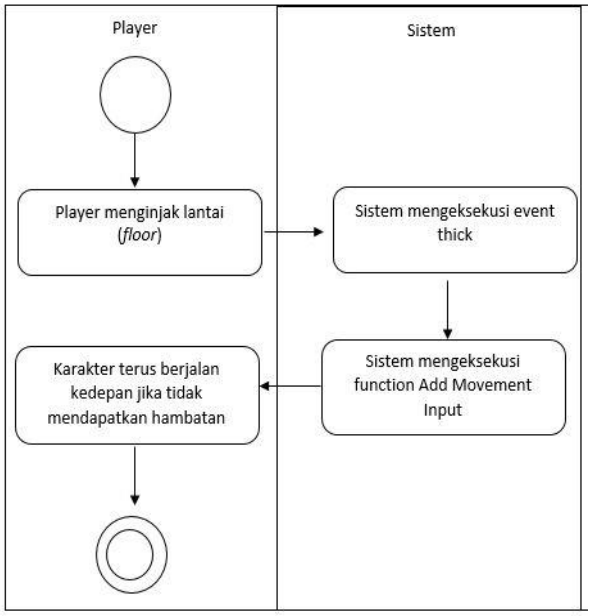
2. *Activity* Diagram

a. *Acticity* Diagram *Start Game*



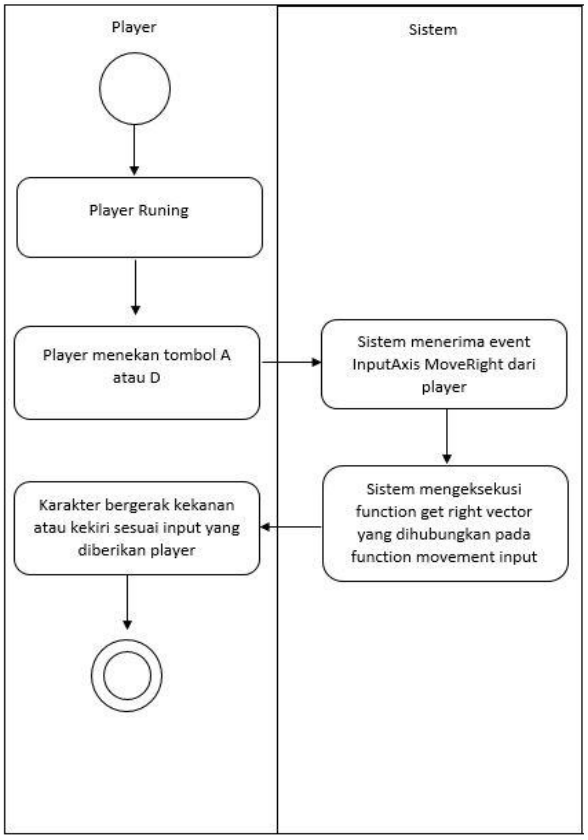
Gambar 4.2 *Activity Diagram Start Game*

b. *Acticity* Diagram *Running*



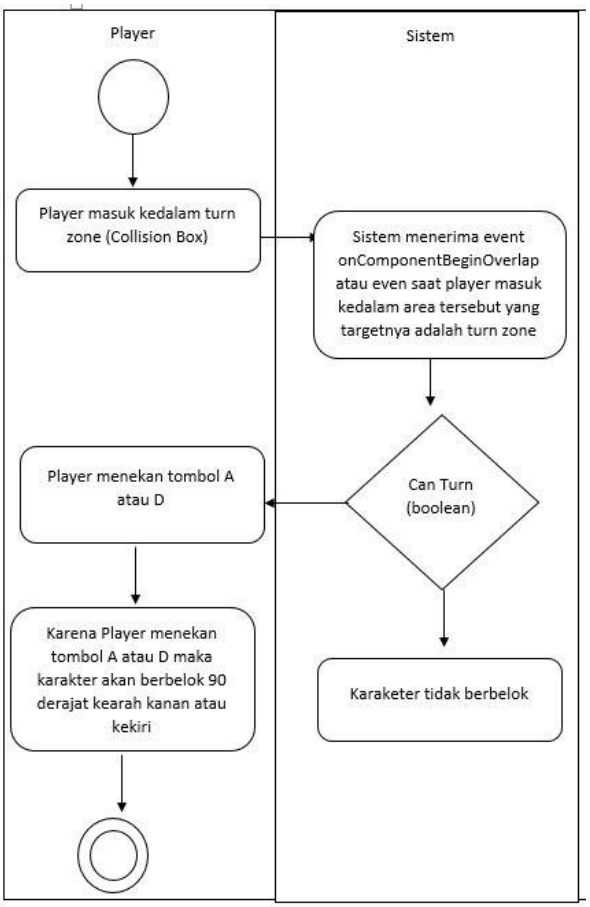
Gambar 4.3 *Activity Diagram Running*

c. *Acticity* Diagram Belok



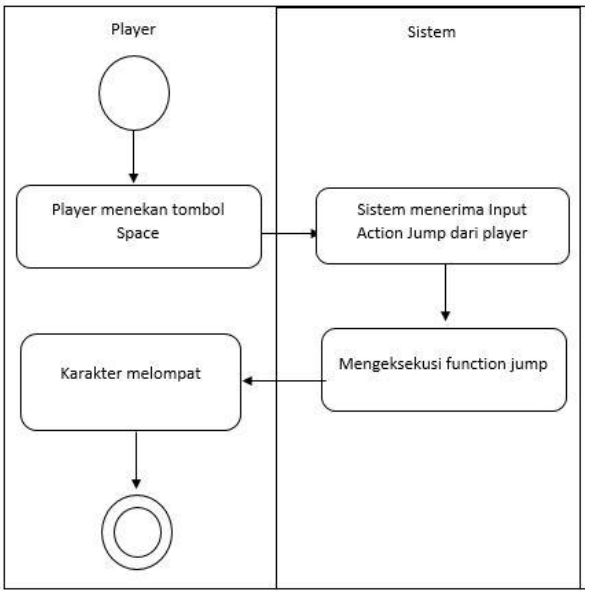
Gambar 4.4 *Activity Diagram* Belok

d. *Acticity* Diagram *Turn Corner*



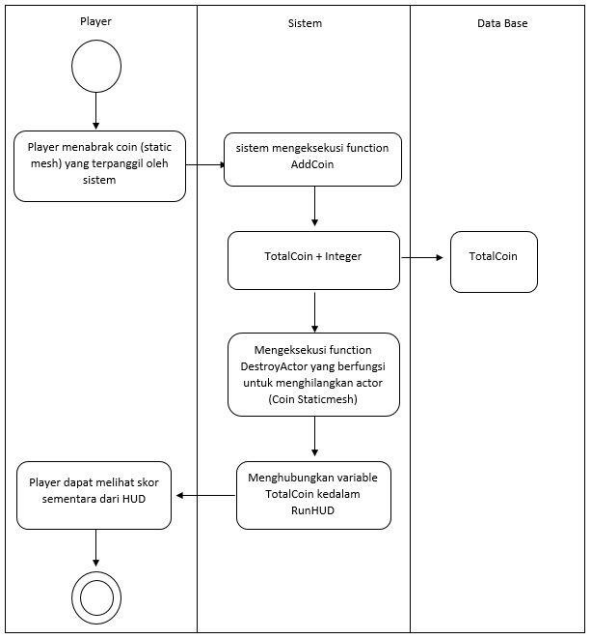
Gambar 4.5 *Activity Diagram Turn corner*

e. *Acticity* Diagram Lompat



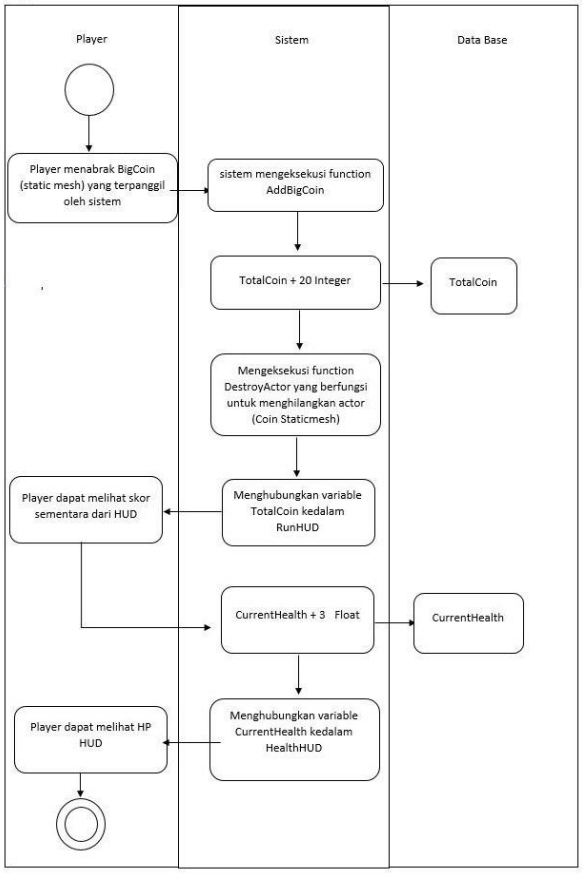
Gambar 4.6 *Activity Diagram* Lompat

f. *Acticity* Diagram Mendapatkan Koin



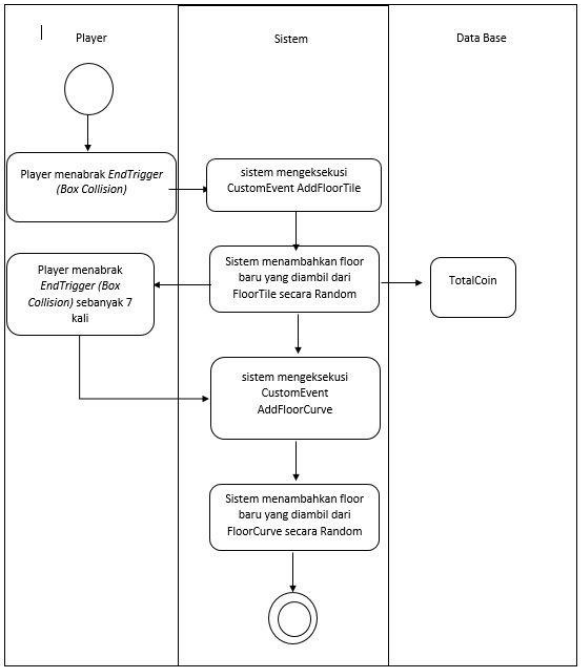
Gambar 4.7 *Activity Diagram* Mend*a*patkan Koin

g. *Acticity* Diagram Mendapatkan Koin Besar



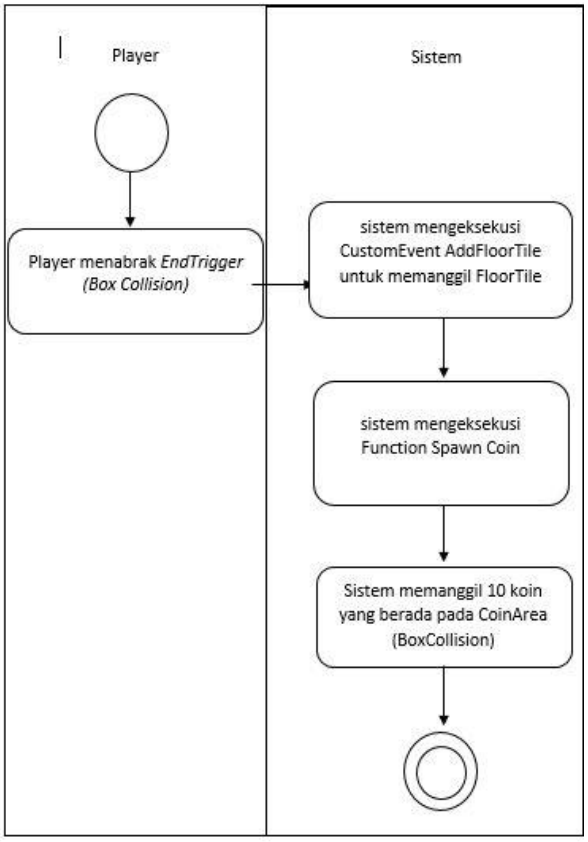
Gambar 4.8 *Activity Diagram* Mend*a*patkan Koin Besar

h. Activity* Diagram *Spawn Floor



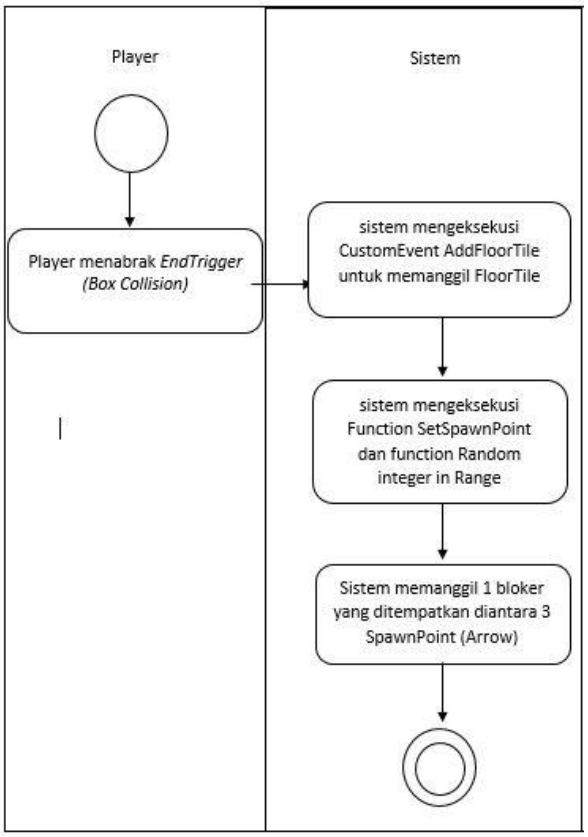
Gambar 4.9 *Activity Diagram Spawn Floor*

i. Activity* Diagram *Spawn Coin



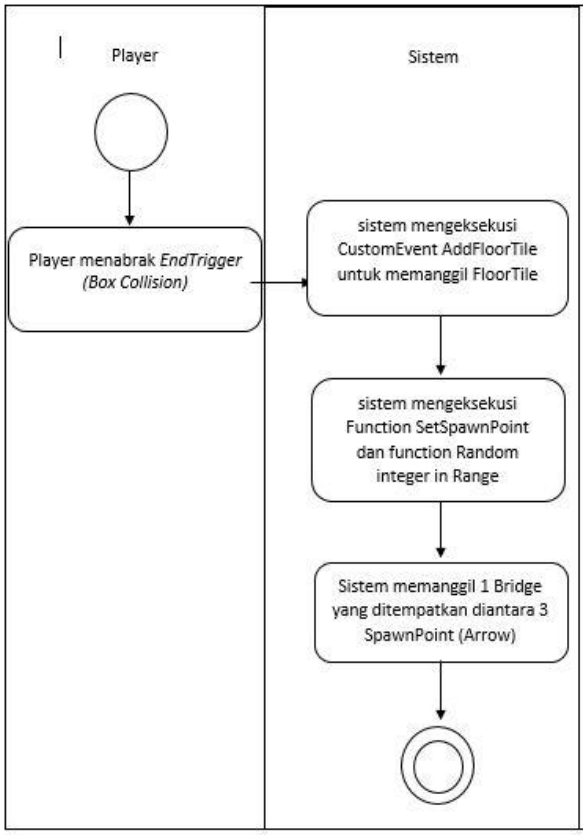
Gambar 4.10 *Activity Diagram Spawn Coin*

j. Activity* Diagram *Spawn Blocker



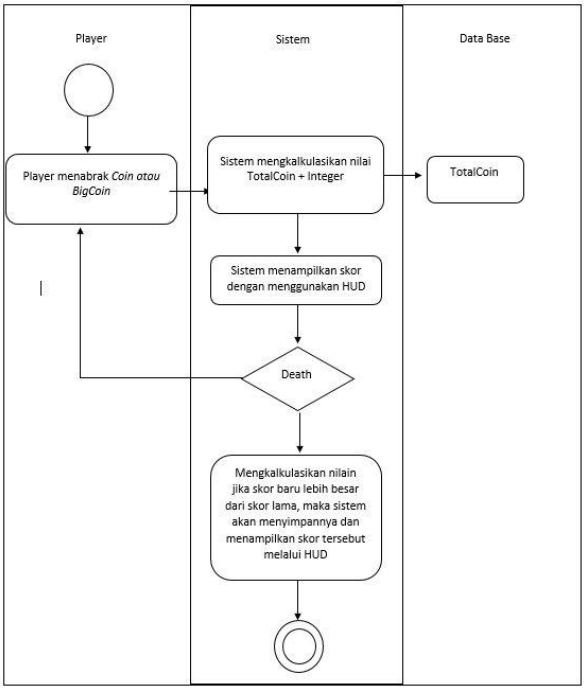
Gambar 4.11 *Activity Diagram Spawn Blocker*

k. Activity* Diagram *Spawn Bridge



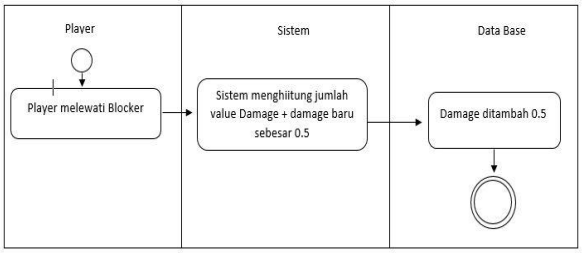
Gambar 4.12 *Activity Diagram Spawn Bridge*

***l. Activity* Diagram Skor**



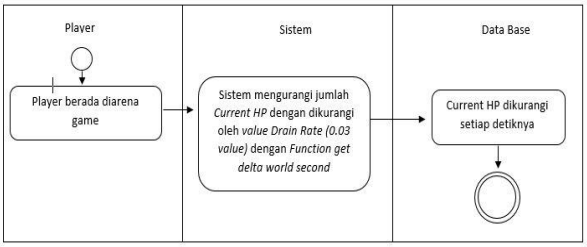
Gambar 4.13 *Activity Diagram* Skor

m. Activity* Diagram Menambahkan *Damage



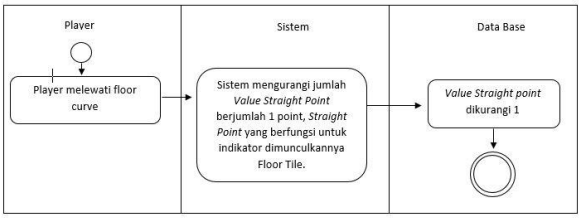
Gambar 4.14 *Activity Diagram* Menambahkan *Damage*

n. Activity* Diagram *Timer



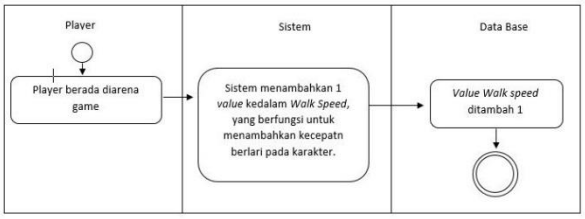
Gambar 4.15 *Activity Diagram Timer*

***o. Activity* Diagram Menambahkan Tingkat Kesulitan**



Gambar 4.16 *Activity Diagram* Menambahkan Tingkat Kesulitan

***p. Activity* Diagram Menambahkan Tingkat Kesulitan**



Gambar 4.17 *Activity Diagram* Menambahkan Kecepatan Berlari

3. Struktur Tabel

a. Struktur Tabel *RunCharacter*

Nama Tabel : *RunCharacter*
Kunci *Field* : *RunCharacter*

No.	**Nama *Field***	***Type***	***Size***	**Keterangan**
1	*RunCharacter*	*Actor*	-	*Primary key*
2	*MinHealth*	Int	0	Minimal *HP*
3	*MaxHealth*	Int	10	Maksimal *HP*
4	*CurrentHealth*	Int	8	*HP* saat *game* baru dimulai
5	*DrainRate*	Int	0.1	Jumlah *HP* yang dikurangi setiap detiknya
6	*BaseTurnRate*	Int	7	Banyaknya red orbs
7	*DesireRotation*	Rotator	-	Arah rotasi antara Roll,Pitch, dan Yaw
8	*CanTurn*	Boolean	-	Kondisi saat akan berbelok
9	Total *Coin*	*Integer*	0	Jumlah koin
10	*GameHUD*	*RunHUD*	-	*SecondaryKey*
11	*WalkSpeedRate*	*Float*	1	Tambahan *value* untuk *walk speed*
12	*Damage*	*Float*	0.5	*Damage* saat menabrak *Blocker*

b. Struktur Tabel *BP_FloorTile*

Nama Tabel : *BP_FloorTile*
Kunci *Field* : *BP_FloorTile*

No.	**Nama *Field***	***Type***	***Size***	**Keterangan**
1	*BP_FloorTile*	*Actor*	0	*Primary key*
2	*SpawnPoint*	*Array Transform*	0	*Spawn Floor*
3	*SpawnPoint2*	*Array Transform*	0	*Spawn Floor*

c. Struktur Tabel *RunGameMode*

Nama Tabel : *RunGameMode*
Kunci *Field* : *RunGameMode*

No.	**Nama *Field***	***Type***	***Size***	**Keterangan**

1	*RunGame Mode*	*Game* Mode	0	*Primary key*
2	*NextPawn Point*	*Transform*	1	*Floor* yang akan dipanggil setelah *player* menabrak *EndTrigger*
3	*FloorTile*s	*Array Actor*	2	Jenis *floor* di dalam *FloorTiles*
4	*Floor*Curves	*Array Actor*	5	Jenis *Floor* di dalam *FloorCurves*
5	*CurrentStraight*s	*Int*	0	Jumlah tabrakan aktor pada *EndZone (*CollisionBox*)*
6	*Straight*Point	*Int*	20	
7	*MinStraightPoint*	*Int*	3	Minimal untuk *Straight*Point

d. Struktur Tabel *HealthHUD*

Nama Tabel : *HealthHUD*
Kunci *Field* : *Healthpercentage*

No.	**Nama *Field***	***Type***	***Size***	**Keterangan**
1	*Healthpercentage*	*Float*	-	*Primary key*
2	*HealthText*	*Text*	-	Jumlah *HP* dalam bentuk angka

e. Struktur Tabel *RunHUD*

Nama Tabel : *RunHUD*
Kunci *Field* : *RunHUD*

No.	**Nama *Field***	***Type***	***Size***	**Keterangan**
1	*RunHUD*	UserWidget	-	*Primary key*
2	*RunCharacter*	*RunCharacter*	-	*SecondaryKey*
4	Total*Coin*	*Integer*	-	*SecondaryKey*

D. Analisis Kebutuhan Sistem

Analisis kebutuhan sistem menentukan seluruh kebutuhan yang ada pada sistem secara lengkap. Analisis kebutuhan sistem dibagi menjadi dua yaitu analisis kebutuhan fungsional dan analisis kebutuhan non fungsional.

1. Analisis Kebutuhan Fungsional

a. Sistem harus dapat merespon perintah *player* dengan cepat.
   1. *Player* dapat mengontrol karakter untuk berbelok kekanan, kiri, dan melompat.
   2. *Player* dapat melakukan regenerasi *HP* yang disediakan sistem.
   3. *Player* dapat melihat skor.

b. Sistem dapat mengerjakan beberapa fungsi di antaraya :
   1. Menghitung jumlah koin yang didapat.
   2. Menempatkan arena *game* secara acak.
   3. Menempatkan rintangan dan koin secara acak.
   4. Sistem dapat mengkalkulasikan *damage*, waktu dan menambah tingkat kesulitan.

c. Sistem harus dapat membuat laporan berupa menampilkan skor yang berhasil didapatkan *player*.

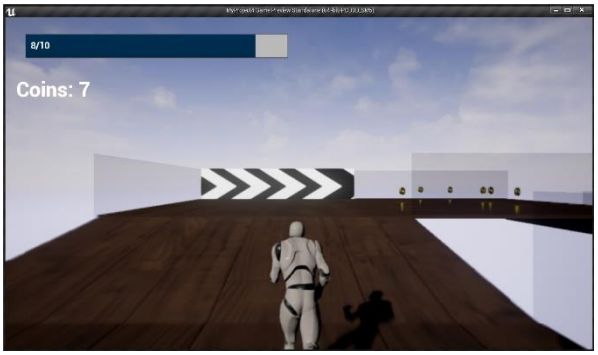
HASIL DAN PEMBAHASAN SISTEM

A. Penggunaan Aplikasi (*Game*)

Game yang dibuat merupakan *game* bergenre *arcade adaptive*, yang bertujuan untuk melatih kecerdasan motorik, sistem akan menambahkan tingkat kesulitan sesuai dengan keadaan *player* saat bermain. Selain itu *game* ini diharapkan dapat membuat *player* atau orang yang memainkannya merasa senang dan bisa menghilangkan rasa jenuh.

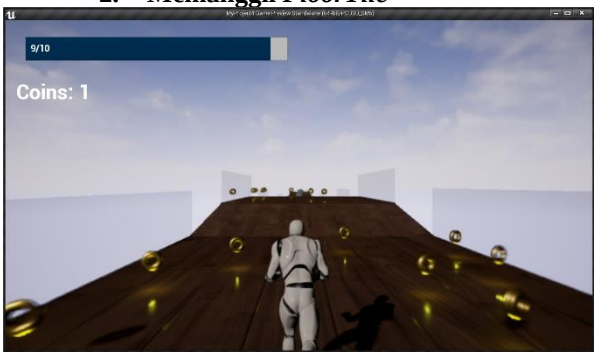
Adapun *rule game* yang diterapkan yaitu *player* ditugaskan untuk megontrol karakter untuk m*end*apatkan koin sebanyak-banyaknya dan menghadapi rintangan yang dimunculkan di dalam arena *game* secara acak.

1. Memunculkan Karakter Dan Mengontrolnya



Pada saat *game* mulai dimainkan, maka karakter utama akan dimunculkan, karakter tersebut akan terus berjalan kedepan, dan *player* bertugas untuk mengontrol karakter yaitu belok, belok 90 derajat, dan melompat

2. Memanggil *FloorTile*

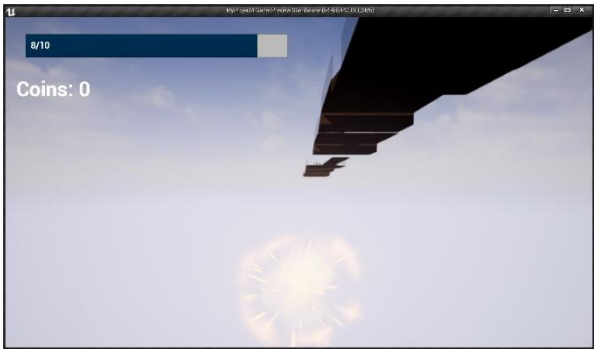


FloorTile merupakan arena *game* utama yang akan selalu dipanggil, di arena inilah *Coin*, dan *Blocker* akan dipanggil sesuai tempat yang telah diprogram, selain itu ada 2 *floor* berbeda yang merupakan bagian dari *FloorTile* yaitu *BP_FloorTile_RampUp* dan *BP_FloorTile_RampDown*.

3. Memanggil *FloorCurve*

Selain *FloorTile*, di dalam *game* ini terdapat lantai yang lain yang jumlah dan waktu pemanggilannya tidak sebanyak *FloorTile*, yaitu *FloorCurve*, terdapat 5 jenis lantai di dalam *FloorCurve* yaitu *BP_Floor_Bridge*, *BP_Floor_Jump*, *BP_FloorEnhergyUP*, *BP_FloorTile_LeftCorner*, dan *BP_FloorTile_RightCorner*.

4. Update Health Point



HP atau *health point* merupakan indikator *Player* dapat bertahan di dalam arena *game* ataupun tidak, *HP* bisa bertambah ataupun berkurang sesuai *rule* yang ada pada *game*.

5. *Update* Skor

Skor, merupakan Text yang dihubungkan dengan *Variable* Total*Coin*, Skor akan bertambah 1 *value* jika *player* menabrak *coin*, dan akan bertambah sebanyak 20 *value* jika menabrak *Big Coin*.

B. Rencana Pengujian

Pengujian adalah proses penelusuran program untuk dicari ada atau tidaknya kesalahan atau fungsi yang tidak sesuai dari tujuan pengembangan program yang dibuat, agar dapat dilakukannya perbaikan jika terdapat kesalahan dalam *game* yang dibuat. Pengujian akan dilakukan dengan menggunakan metode *black box*.

Item Uji	**Butir Uji**	**Metode**
Karakter	*Running*	*Black Box*
	Belok	*Black Box*
	Turn Corner	*Black Box*
	Melompat	*Black Box*
Update Skor	Mendapatkan Koin	*Black Box*
	Mendapatkan Koin Besar	*Black Box*

Update HP	Menambah *Value* 0.2 *HP* dari *BigCoin*	*Black Box*
	Mengurangi 0.03 *HP* dari waktu yang dipakai	*Black Box*
	Mengurangi *Value* 0.3 *HP* dari *Blocker*	*Black Box*
	Me*reset* Level saat terjatuh	*Black Box*
	Me*reset* Level saat gagal berbelok di *Turnzone*	*Black Box*
FloorTile	Memanggil *BP_FloorTile*	*Black Box*
	Memanggil *BP_FloorTile_RampDown*	*Black Box*
	Memanggil *BP_FloorTile_RampUp*	*Black Box*
FloorCurves	Memanggil *BP_Floor*Bridge	*Black Box*
	Memanggil *BP_Floor_Jump*	*Black Box*
	Memanggil *BP_FloorTile_RightCorner*	*Black Box*
	Memanggil *BP_FloorTile_LeftCorner*	*Black Box*
	Memanggil *BP_Floor*_EnergyUP	*Black Box*
Up Level	*Timer*	*Black Box*
	UpWalkSpeed	*Black Box*
	UpDamage	*Black Box*
	Update Straight Point	*Black Box*

KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan uraian pada bab-bab sebelumnya, maka kesimpulan yang dapat penulis ambil, yaitu:

1. Dengan membuat *game* secara tidak langsung kita bisa memanfaatkan teknologi secara positif.
2. *Unreal engine* menjadi *game engine* yang cukup membantu dalam proses pembuatan *game*, karena menggunakan bahasa pemrograman yang cukup mudah untuk dimengerti dan sangat fleksible.
3. *Game* yang dibuat dibuat mampu beradaptasi dengan kecerdasan *player*, dengan cara menambahkan kecepatan berlari, penambahan damage, dan sering dimunculkannya rintangan jika *player* tersebut dinyatakan pro.

B. Saran

Saran yang dapat penulis sampaikan dalam melaksanakan pembuatan *game arcade adaptive* adalah:

1. Peningkatan spesifikasi *Hardware PC* untuk meningkatkan kenyamanan dalam memainkan *game* ini.
2. Memainkan *game* arcade menjadi sarana yang tepat untuk meningkatkan kecerdasan motorik dan efektif menghilangkan jenuh.
3. Ukuran *Unreal engine 4* yang cukup besar membuat performa komputer menjadi berat, sebaiknya pihak *Unreal engine* lebih mempertimbangkan lagi tentang ukuran *softwar*enya.

DAFTAR PUSTAKA

[1] Asori, Lutfi. 2011 *Pembuatan Game Petualangan Kapitan Pattimura Merebut Benteng Zeelandia dengan Software Rpg Maker Xp*
Sumber: Jurnal Dan Penelitian Sekolah Tinggi Manajemen Informatika dan Computer Amikom Yogyakarta

[2] Christiano, Anthonius Teddy. 2014. *Pembuatan Game Tactical RPG Legen of Fantasia*
Sumber: Jurnal Ilmiah Mahasiswa Universitas Surabaya

[3] Fauzy, Muhammad. 2013. *Pembuatan Game Eduksi Pengenalan Karier Untuk Anak Usia 6-8 Tahun*
Sumber: Jurnal Algoritma Sekolah Tinggi Teknologi Garut

[4] Nilwan, Agustinus. 2012. Pemrograman Animasi dan Game Profesional
Sumber: Buku Pemrograman Animasi dan Game Profesional

[5] Pradana, Harly Yoga. 2012. *Game Pembelajaran Musikal Untuk Anak-anak*
Sumber: jurnal dan penelitiann Universitas Pembangunan Nasional Veteran Yogyakarta

[6] Setiawan , Rico Firstano Finnanta. 2012. *Pembuatan Game Evaluasi Operasi Matematika Dasar Untuk Siswa Sekolah Dasar Kelas Tiga*
Sumber: jurnal ilmiah mahasiswa universitas Surabaya