BAB II

TINJAUAN PUSTAKA

II. 1 Penelitian Terdahulu

Untuk mendukung keberhasilan penelitian, maka dilakukan pendekatan teoritis melalui beberapa literatur yang berhubungan dengan penelitian yang dilakukan oleh penulis. Adapun tinjauan pustaka yang diperoleh dalam penelitian ini adalah sebagai berikut:

1. Berdasarkan penelitian yang ditulis oleh Dwi Septihadi (2021) dengan judul "Pengembangan Game Multiplayer Ular Tangga Bergenre Strategi Menggunakan Unity". dari penelitian ini peneliti mengembangkan suatu *game 3D* yang di buat menggunakan *Unity* sebagai *Game Engine* dan *Blender* sebagai program untuk pembuatan model *3D,* dengan hasilnya yaitu sebuah *game* ular tangga strategi yang bisa dimainkan secara *multiplayer*.
2. Berdasarkan penelitian yang dilakukan oleh Janne Soikkeli (2021) dengan judul "Creating a Rhythm Audio Game'. peneliti mengembangkan suatu *game* musik dengan hanya menggunakan implementasi *Audio* sebagai media yang di gunakan untuk bermain *game* ini, dengan kata lain *game* ini bisa dimainkan tanpa media *visual.*

3. Berdasarkan penelitian yang Ditulis oleh Ruben Rodrigues Rebelo (2016) dengan judul "Building a Music Rhythm Video Game". peneliti mengembangkan *prototype* game musik yang terinspirasi dari karateristik *Rhythm Game* lain dan dikembangkan menggunakan *Android SDK* sebagai *Game Engine* untuk mendukung berbagai fungsi dan mekanik yang terdapat dalam game tersebut.
4. Berdasarkan penelitian yang Ditulis oleh Dzulfickar Rhezaa Algadhari (2020) dengan judul "Pengembangan Rhythm Game Berbasis Android dengan Unity 2D Sebagai Media Pembelajaran Seni Musik Gamelan". dari penelitian ini peneliti memanfaatkan *Unity* sebagai *Game Engine* untuk membuat *rhythm game* yang ditujukan sebagai media audiovisual pembelajaran seni musik Gamelan.
5. Berdasarkan penelitian yang Ditulis oleh Alphianno Steady Kambodji (2016) dengan judul "Rancang Bangun Rhythm Game Sebagai Sarana Belajar Bermain Gitar". peneliti mengimplementasikan sebuah *rhythm game* yang menggunakan deteksi frekuensi nada gitar sebagai input dengan *DPE Framework* sebagai media perancangan *game* dan *Unity* sebagai media pengimplementasian fungsi *game.*

II. 2 Perancangan

Menurut Jogiyanto (2005 : 129), mengatakan analisis sistem sebagai penguraian dari suatu sistem informasi yang utuh ke dalam

bagian-bagian komponennya dengan maksud untuk mengidentifikasikan dan mengevaluasi permasalahan dan kesempatan.

II. 3 *Game*

Menurut Zamroni, Suryawan, dan Jalaluddin (2013: 489), permainan sebuah sistem dimana pemain terlibat dalam konflik buatan. Pemain berinteraksi dengan sistem dan konflik dalam permainan. Dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan.

II. 4 *Rhythm game*

Rhythm game adalah salah satu *genre game* dimana pemain harus menekan suatu tombol atau not sesuai dengan irama musik yang di mainkan untuk mendapatkan skor tertinggi atau menyelesaikan suatu lagu tanpa ada not yang tertinggal, lebih di kenal dengan sebutan "*Full Combo"* atau di singkat *"FC"*.

Terdapat beberapa contoh *rhythm game* yang terkenal dan sukses di pasaran, diantaranya adalah *Dance Dance Revolution, Guitar Hero, osu!, Etterna, Rock Band, Friday Night Funkin',* dan lain-lain.

II. 5 Pembuatan *game*

Game pada umumnya di buat menggunakan Komputer yang dibekali dengan *Game Engine* yang merupakan perangkat lunak yang di rancang untuk mempermudah pembuatan dan pengembangan *game.* Beberapa contoh *Game Engine* yang banyak digunakan seperti *Unity* dengan fleksibilitas tinggi untuk media pengembangan dan perilisan *game*,

Unreal Engine yang berfokus di pengembangan *game* berskala besar dengan kualitas grafik tinggi, dan *RPG Maker* yang memudahkan pembuatan *game* berjenis *RPG (Role-Playing Game).*

III. 6 *Unity*

Unity adalah salah satu *game engine* terkenal yang banyak dipakai dari kalangan pengembang *game indie* sampai perusahaan *game* besar. Berawal dari *game engine* yang dirancang khusus untuk Mac OS X, saat ini sudah mendukung banyak *platform* dari *desktop* (*Windows, macOS, Linux*), *web* (*Unity WebPlayer*), *mobile* (*Android, iOS*), konsol *game* (*PlayStation 4, PlayStation 5, Xbox One, Xbox Series X/S, Nintendo Switch, Google Stadia*) , dan *virtual reality* (*Oculus Rift, Oculus Quest, Valve Index, Microsoft HoloLens, PlayStation VR*).

Gambar II.1. Tampilan UI dari *Unity*

Unity menyediakan beberapa fitur unggulan yang memudahkan pembuatan *game* baik itu *game 2D* maupun *game 3D* seperti sistem *Scene* yang memudahkan pembagian menu dari *game* yang dibuat, *Asset Store*

sebagai tempat men*download* berbagai *asset* yang disediakan untuk di gunakan ke dalam game yang sedang dikembangkan, proses impor *sprite* untuk *game 2D*, dan dukungan pengaturan grafis seperti kompresi tekstur, pengaturan resolusi per *platform*, dan penambahan efek pasca-proses untuk *game 3D*.

III. 7 *Unified Modeling Language* (UML)

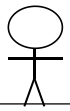
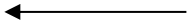
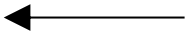
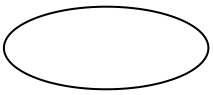
UML yaitu satu kumpulan konvensi permodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem perangkat lunak yang terkait dengan objek. UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan system yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan. (Andikos, 2019 : 39).

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Use Case* Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case diagram* dapat digambarkan dengan sumber-sumber pada Tabel II.1.

Tabel II.1. Simbol *Use Case*

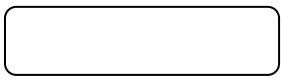
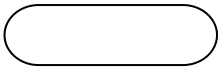
Gambar	Nama	Keterangan
	*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
	Depedency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (independent) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
	*Generalization*	Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor).
	Include	Menspesifikasikan bahwa use case sumber secara eksplisit.
	*Extend*	Menspesifikasikan bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan.
	Association	Apa yang mnghubungkan antara objek satu dengan objek lainnya.
	*System*	Menspesifikan paket yang menampilkan sistem secara terbatas.
	*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor.
	*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
	*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

(Sumber : Andikos, 2019 : 39)

2. Diagram Aktivitas (*Activity Diagram*)

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* dapat digambarkan dengan simbol-simbol seperti pada tabel II.2.

Tabel II.2. Simbol *Activity Diagram*

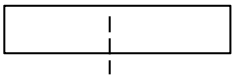
Gambar	Nama	Keterangan
	*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
	*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
	*Initial Node*	Bagaimana objek dibentuk atau diawali.
	*Activity Final*	Bagaimana objek dibentuk dan dihancurkan
	*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

(Sumber : Andikos, 2019 : 39)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. *Sequence Diagram* dapat digambarkan dengan simbol-simbol seperti pada Tabel II.3.

Tabel II.3. Simbol *Sequence Diagram*

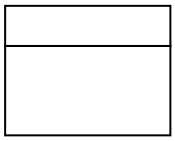
Gambar	Nama	Keterangan
	*Lifeline*	Objek entity, antarmuka yang saling berinteraksi.
	*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.
	*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.

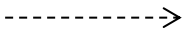
(Sumber : Andikos, 2019 : 39)

4. *Class Diagram* (Diagram Kelas)

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain. *Class diagram* dapat digambarkan dengan simbol-simbol seperti pada Tabel II.4.

Tabel II.4. *Class Diagram*

Gambar	Nama	Keterangan
	Generalization	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
	*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
	*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.

	*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
	Realization	Operasi yang benar-benar dilakukan oleh suatu objek.
	*Depedency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
	Assocation	Apa yang menghubungkan antara objek satu dengan objek lainnya

(Sumber : Andikos, 2019 : 39)