# BAB II
# LANDASAN TEORI

## 2.1 Tinjauan Pustaka

Penelitian yang dilakukan Tamin, Suarjaya dan Pratama (2019) dalam penelitian yang berjudul "Rancang Bangun *Game* Platformer Daud VS Goliat Berbasis Android" konsep penelitian dari *game* ini adalah pengenalan kisah Alkitab dengan teknologi dan pembuatan game edukasi dengan mengangkat cerita yang memiliki pesan moral. Pembuatan game ini menggunakan *game engine* Unity dengan bahasa pemrograman C#. Tujuan *game* ini adalah *player* dapat memahami dan mempelajari cerita-cerita Alkitab dengan metode yang ringan dan menyenangkan. *Game* ini menggunakan tampilan visual 2D, dan terdapat tiga *scene* yaitu *scene* menu utama, *scene* dialog pemain, *scene* permainan.

Penelitian yang dilakukan Sriyadi (2018) dalam penelitian yang berjudul "Membuat *Game* Petualangan Sejarah Pangeran Jawa Berbasis Unity 3D" *game* ini memiliki cerita tentang perjuangan Pangeran Diponegoro yang melawan para penjajah, *game* ini di tujukan bagi remaja agar dapat mempelajari dan memahami kisah perjuangan para pahlawan Indonesia. *Game* ini menggunakan *game engine* Unity 3D, dan Adobe *After Effect* CS6 untuk *software* pembuat animasi.

Penelitian yang dilakukan Kusumo dan Nita (2019) dalam penelitian yang berjudul "Perancangan *Game* Android *Adventure* Gajah Mada dengan Metode *Agile Development*" *game* ini menceritakan kisah perjalanan Gajah Mada pada zaman kerajaan Majapahit. *Game* ini di bangun menggunakan Unity 3D, Blender, Adobe Fuse dan memiliki tema tampilan alam bebas yang berisikan hutan, gunung dan berbatuan. Konsep dasar dari game ini adalah menggunakan labirin untuk menemukan keris yang dulu digunakan Gajah Mada untuk berperang.

## 2.2 Pengertian Rancang Bangun

Menurut Kamus Besar Bahasa Indonesia ( KBBI) Rancang merupakan kata dasar dari merancang yang mempunyai arti mengatur segala sesuatu sebelum bertindak, mengerjakan atau melakukan sesuatu.

Menurut Maulani dkk dalam Jurnal ICIT Vol. 4 No. 2 (2018:157), Rancang bangun adalah menciptakan dan membuat suatu aplikasi ataupun sistem yang belum ada pada suatu instansi atau objek tersebut.

## 2.3 Pengertian *Game* 3D

*Game* adalah media untuk melakukan aktifitas bermain. Aktifitas bermain merupakan suatu aktifitas yang meliputi pemecahan masalah yang menjadi tantangan dari *game* tersebut, dengan mengikuti suatu aturan tertentu (Tulung dkk, 2017).

3D berarti tiga dimensi, sesuatu yang memiliki lebar, tinggi dan kedalaman (panjang). Lingkungan yang disekeliling adalah tiga dimensi dan melakukan aktivitas seperti berjalan kita melihat sekeliling dengan 3D setiap hari. Manusia dapat merasakan hubungan spasial antara objek hanya dengan melihat mereka karena kita memiliki persepsi 3D, juga dikenal sebagai kedalama presepsi. Ketika kita melihat sekeliling, retina di setiap mata membentuk gambar dua dimensi dari lingkungan kita dan otak kita memproses dua gambar ini menjadi pengalaman visual 3D ( Maulana, 2019).

## 2.4 Game Adventure

*Game Adventure* merupakan jenis *game* dimana pemain diasumsikan sebagai tokoh utama dalam cerita interaktif yang didukung oleh penjelajahan dan teka-teki (Usma, 2020).

*Game Adventure*, *game* dengan genre ini adalah sebuah permainan yang menarik pecinta *game* untuk memainkannya. Bagaimana tidak kita akan disungguhkan dengan konsep penasaran, seperti kita akan mendapatkan rintangan tertentu dalam bermain, kemudian akan mendapatkan hadiah ketika berhasil melewati rintangan tersebut. *Adventure* juga menjadi game yang paling digemari diseluruh dunia. Umumnya *game* ini lebih kepada pemecahan sebuah misteri seperti film *game* jumanji (Aula, 2020).

*Game Adventure* adalah *game* petualangan. pemain berjalan ke suatu tempat di sepanjang jalan pemain akan menemukan banyak barang dan perlengkapan yang akan pemain simpan. Kami menggunakan alat-alat ini selama perjalanan, baik untuk membantu kami maupun memandu kami. Jenis permainan ini tidak fokus

pada pertarungan, terkadang ada tapi sedikit. Umumnya *game* ini lebih menekankan pada pemecahan misteri daripada bertempur sampai mati (Kevin, 2017).

Dari ketiga pendapat diatas dapat di simpulkan bahwa definisi *game adventure* adalah *game* yang bertemakan petualangan, tidak berfokus pada pertarungan jika ada pasti *scene* pertarungan nya hanya sedikit, jenis *game* ini lebih menekankan pada teka-teki dan misteri yang harus di pecahkan untuk mencapai misi tertentu dengan berbagai kesulitan rintangan di setiap level nya dan ketika misi itu berhasil di selesaikan maka *player* akan mendapat hadiah.

## 2.5 Bahasa Pemrograman C#

C Sharp adalah suatu bahasa pemrograman berorientasi objek yang diperkenalkan oleh Microsoft pada januari 1999. Dalam bahasa pemrogramman C Sharp menyederhanakan banyak kompleksitas dari bahasa pemrogramman C++ dan menyediakan fitur seperti nilai *null able*, delegasi, ekspresi lambda dan akses memori secara langsung. C Sharp mendukung metode generik dan tipe, yang membuat peningkatan kinerja dan iterator, yang dapat memungkinkan pelaksana koleksi kelas untuk menentukan perilaku yang mudah untuk digunakan oleh klien. Bahasa pemrogramman C Sharp mendukung konsep enkapsulasi, pewarisan dan polimorfisme. Semua variable, metode, termasuk juga metode utama dan titik masuk aplikasi, dikemas dalam satu definisi kelas (Tulung dkk, 2017).

## 2.6 Unity Game Engine

Unity 3D adalah salah satu *game engine* yang berfungsi sebagai *software* pengolah gambar, input, suara, grafik, dan lain-lain yang nantinya akan digunakan untuk membuat suatu *game*, walaupun tidak selamanya untuk membuat *game* (Nugroho dan Pramono, 2017).

### 2.6.1 Sejarah Unity *Game Engine*

Unity *Technologies* dibangun pada tahun 2004 di Copenhagen Denmark oleh David Helgason sebagai CEO, Nicholas Francis sebagai CCO dan Joachim Ante sebagai CTO. Pertama kali, Unity *Technologies* menerima pendanaan dari Sequoia Capital, WestSummit *Capital* dan iGlobe *Partners*.

Tahun 2008, Unity menjadi *game engine* pertama yang mendukung penuh *platform* iPhone. Pada tahun 2009, Unity mulai meluncurkan produk mereka secara gratis.

Tahun 2019, Unity *Technologies* melakukan pengembangan *real-time* 3D *platform* untuk kreator di industri *game* dan industri lainnya (*otomotive*, arsitektur dan lain-lain).

Tahun 2020, tepatnya bulan Maret Unity *Technologies* melakukan pengembangan pada pembuatan karya seni berbasis AI-*Assisted* untuk memecahkan masalah sulit di dalam membuat dan menskalakan konten berkualitas tinggi, memungkinkan seniman 3D lebih banyak waktu untuk fokus pada kreativitas.

### 2.6.2 Fitur Unity 3D *Game Engine*

#### 1) Rendering

Graphics *engine* yang digunakan adalah Direct3D (Windows, Xbox 360), OpenGL (Mac, Windows, Linux, PS3), OpenGL ES (Android, iOS), dan *proprietary* APIs (Wii). Ada pula kemampuan untuk *bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion* (SSAO), *dynamic shadows using shadow maps, render-to-texture and full-screen post-processing effects.*

Unity dapat mengambil format desain dari 3ds Max, Maya, Softimage, Blender, modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe *Fireworks* and *Allegorithmic Substance*. Asset tersebut dapat ditambahkan ke *game project* dan diatur melalui graphical *user interface Unity.Scripting.*

*Script game engine* dibuat dengan Mono 2.6, sebuah implementasi *open-source* dari .NET Framework. Programmer dapat menggunakan Unity *Script* (bahasa terkustomisasi yang terinspirasi dari sintax ECMAScript, dalam bentuk JavaScript), C#, atau Boo (terinspirasi dari sintax bahasa pemrograman phyton). Dimulai dengan dirilisnya versi 3.0, Unity menyertakan versi *Mono Develop* yang terkustomisasi untuk *debug script*.

**2) Asset Tracking**

Unity juga menyertakan Server Unity Asset – sebuah solusi terkontrol untuk *developer* game asset dan *script*. Server tersebut menggunakan PostgreSQL sebagai *backend*, sistem audio dibuat menggunakan FMOD *library* (dengan kemampuan untuk memutar Ogg *Vorbis compressed* audio), video *playback* menggunakan Theora codec, *engine* daratan dan vegetasi (dimana mensuport *tree billboarding*, *Occlusion Culling* dengan *Umbra*), *built-in lightmapping* dan global *illumination* dengan *Beast*, multiplayer networking menggunakan *RakNet*, dan navigasi mesh pencari jalur bawaan.

**3) Platforms**

Unity support pengembangan ke berbagai plaform. Didalam *project*, *developer* memiliki kontrol untuk mengirim keperangkat *mobile*, *web browser*, desktop, and *console*. Unity juga mengijinkan spesifikasi kompresi tekstur dan pengaturan resolusi di setiap *platform* yang didukung.

Saat ini platform yang didukung adalah BlackBerry 10, Windows 8, Windows Phone 8, Windows, Mac, Linux, Android, iOS, Unity Web Player, Adobe Flash, PlayStation 3, Xbox 360, Wii U and Wii. Meskipun tidak semua terkonfirmasi secara resmi, Unity juga mendukung PlayStation Vita yang dapat dilihat pada *game Escape Plan* dan Oddworld.

**4) Asset Store**

Diluncurkan November 2010, Unity Asset *Store* adalah sebuah *resource* yang hadir di Unity editor. Asset *store* terdiri dari asset *packages*, beserta 3D models, *textures* dan *materials*, sistem *particle*, musik dan efek suara, tutorial dan project, *scripting package*, editor *extensions* dan servis online.

**5) Physics**

Unity juga memiliki *suport built-in* untuk PhysX physics *engine* (sejak Unity 3.0) dari Nvidia (sebelumnya Ageia) dengan penambahan kemampuan untuk simulasi *real-time cloth* pada *arbitrary* dan *skinned meshes*, *thick ray cast*, dan *collision layers*.

### 2.6.3 User Interface Unity 3D Game Engine

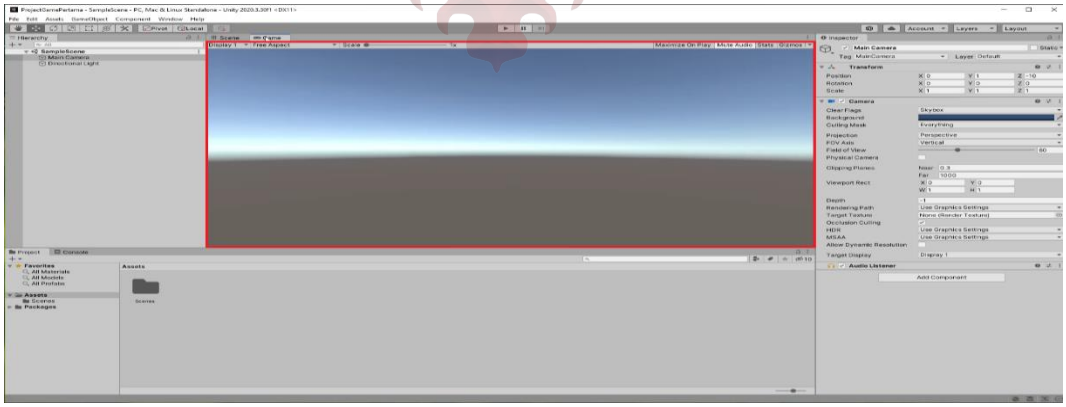
Unity selalu melakukan perbaikan pada setiap versinya. Unity 3D versi 2020.3.35f1 menyediakan berbagai macam fitur yang akan sangat membantu para *game developer*,fitur tersebut dapat di lihat pada Gambar 2.1.

Gambar 2. 1 *Project Window* Unity

Unity telah memberikan tab-tab penting yang diperlukan untuk membuat game yang simpel. Beberapa tab ini diantaranya:

**1) *Game Preview***

Pada *Game Preview* terdapat tampilan dari kamera yang ada pada *Scene*. Selain tampilan kamera juga ada tampilan-tampilan lain yang muncul hanya pada *game preview* seperti UI. Tidak seperti *scene* Editor, pada Game Preview tidak terdapat navigasi scene, akan tetapi saat *game* di play, kita bisa mencoba input atau tombol-tombol UI pada Game Preview ini, *game preview* tersebut dapat di lihat pada Gambar 2.2.



Gambar 2. 2 *Game Preview* Unity

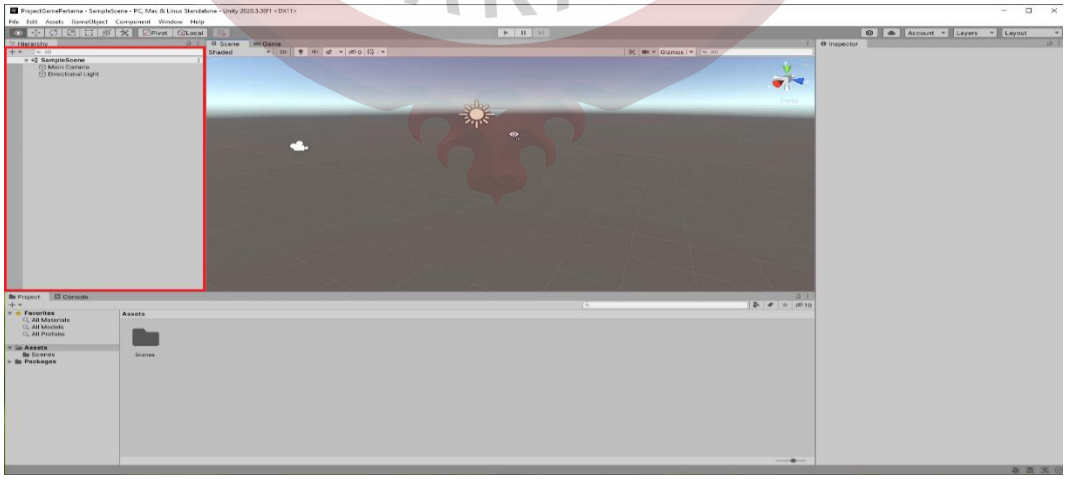
**2) *Scene* Editor**

*Scene* Editor adalah tempat utama kita untuk mengatur set dari sebuah *game* seperti pada set film. Konsepnya sama, jadi ada kamera yang akan mengarah ke sebuah set, dan kamera itulah yang nantinya akan merekam dan menampilkan tampilan *game* ke pemain., terdapat fitur navigasi dengan menahan klik kanan, lalu menggerakkannya dengan *keyword* WASD dan dengan menggerakkan mouse. *Scene* editor dapat di lihat pada Gambar 2.3.

Gambar 2. 3 *Scene Editor* Unity

**3) *Hierarchy***

*Hierarchy* adalah list *game object* yang ada pada game kita. Pada *Hierarchy* juga kita bisa melihat *game object* yang merupakan *child* atau *parent* dari *game object* lain. Tampilan *hierarchy* dapat dilihat pada Gambar 2.4.



Gambar 2. 4 *Hierarchy* Unity

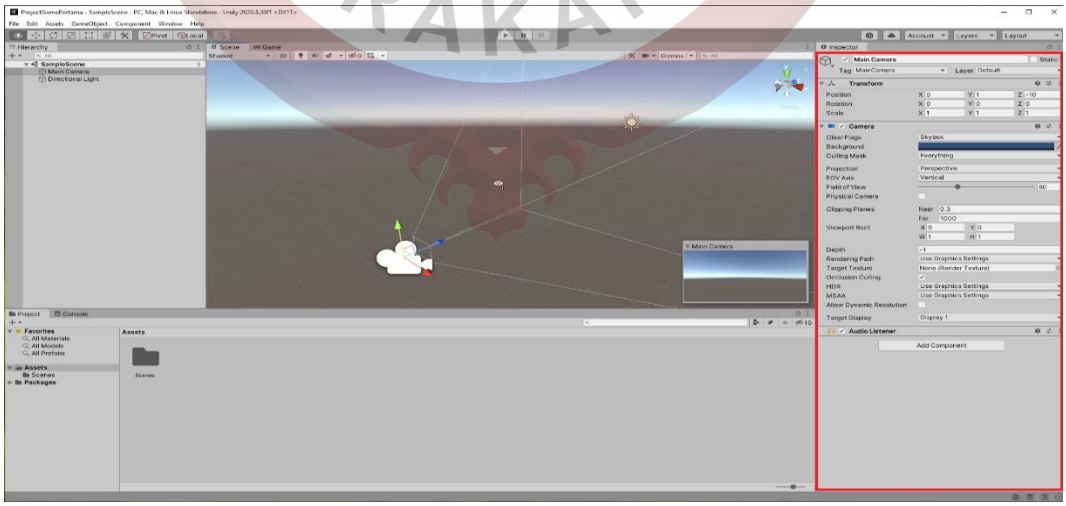
**4) *Project Explorer***

*Project Explorer*, sama seperti file *explorer* yang ada di komputer kita, berguna untuk melihat file-file yang ada pada project kita. Semua file ini ada dalam folder Assets pada folder *project* Unity kita. Kita bisa dengan mudah memasukan asset *game* seperti model karakter kita ke dalam *scene* dengan menggunakan *Project Explorer* ini. Tampilan *Project Explorer* dapat dilihat pada Gambar 2.5.

Gambar 2. 5 *Project Explorer* Unity

**5) *Inspector***

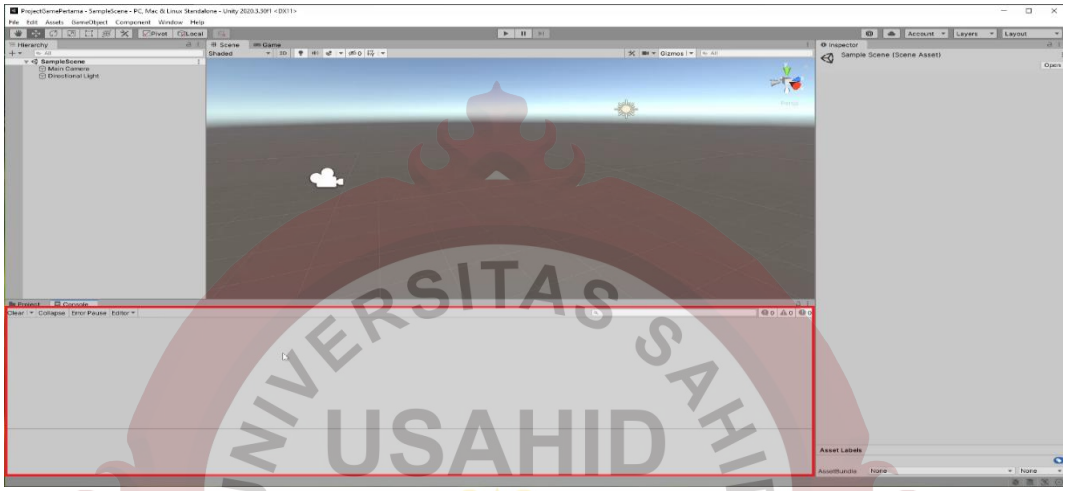
*Inspector* ini digunakan untuk melihat *game component* yang ada pada *game object* yang sedang dipilih. Kita juga bisa langsung mengganti opsi-opsi dari *game component* yang ditampilkan. Tampilan *inspector* dapat dilihat pada Gambar 2.6.



Gambar 2. 6 *Inspector* Unity

### 6) *Console Log*

*Console log* terletak bersamaan dengan *Project Explorer*. Ini digunakan untuk melihat informasi yang diberikan oleh *game object* baik itu *error*, *warning*, maupun hanya *log* untuk mengecek fungsi yang kita buat. Tampilan *Console log* dapat dilihat pada Gambar 2.7.



Gambar 2. 7 *Console Log* Unity

## 2.7 Flowchart

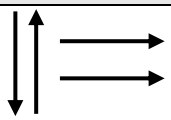
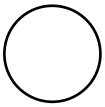
*Flowchart* atau sering disebut dengan diagram alir merupakan suatu jenis diagram yang merepresentasikan algoritma atau langkah-langkah instruksi yang berurutan dalam sistem. seorang analis sistem menggunakan *flowchart* sebagai bukti dokumentasi untuk menjelaskan gambaran logis sebuah sistem yang akan dibangun kepada *programmer*. Pada dasarnya, *flowchart* digambarkan dengan menggunakan simbol-simbol. Setiap simbol mewakili suatu proses tertentu. Sedangkan untuk menghubungkan satu proses ke proses selanjutnya digambarkan dengan menggunakan garis penghubung (Prasetyo, 2019).

Pada dasarnya, dalam merancang *flowchart* tidak ada ketentuan mutlak yang harus dipenuhi. Berikut akan dijelaskan mengenai simbol-simbol *flowchart* yang dibagi kedalam 3 kategori, diantaranya:

1) Simbol Arus (*Flow Direction Symbols*)

Simbol yang termasuk kedalam ketegori ini digunakan sebagai simbol penghubung. Simbol-simbol tersebut dapat dilihat pada tabel 2.1.

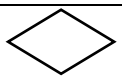
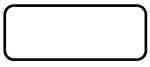
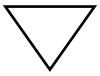
Tabel 2. 1 Simbol Arus (*Flow Direction Symbols*)

| Simbol | Nama | Fungsi |
|---|---|---|
|  | *Flow Direction Symbol/Connecting Line* | Berfungsi untuk menghubungkan simbol yang satu dengan yang lainnya, menyatakan arus suatu proses. |
|  | *Communication Link* | Berfungsi untuk transmisi data dari satu lokasi ke lokasi lain. |
|  | *Connector* | Digunakan untuk menyatakan sambungan dari proses yang satu ke proses berikutnya di halaman yang sama. |
|  | *Offline Connector* | Digunakan untuk menyatakan sambungan dari proses yang satu ke proses berikutnya di halaman yang berbeda. |

2) Simbol Proses (*Processing Symbols*)

Simbol proses digunakan untuk menyatakan simbol yang berkaitan dengan serangkaian proses yang dilakukan. Simbol tersebut dapat dilihat pada tabel 2.2.

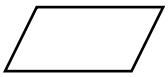
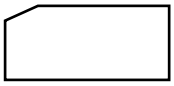
Tabel 2. 2 Simbol Proses (*Processing Symbols*)

| Simbol | Nama | Fungsi |
|---|---|---|
|  | *Processing* | Digunakan untuk menunjukan pengolahan yang akan dilakukan dalam komputer |
|  | *Manual Operation* | Digunakan untuk menunjukan pengolahan yang tidak dilakukan oleh komputer. |
|  | *Decision* | Digunakan untuk memilih proses yang kan dilakukan berdasarkan kondisi tertentu. |
|  | *Predefined Process* | Digunakan untuk mempersiapkan penyimpanan yang sedang atau akan digunakan dengan memberikan harga awal. |
|  | *Terminal* | Digunakan untuk memulai atau mengakhiri program. |
|  | *Offline Storage* | Berfungsi untuk menunjukan bahwa data akan disimpan ke media tertentu |
|  | *Manual Input Symbol* | Digunakan untuk menginput data secara manual dengan keyboard |

3) Simbol I/O (*Input-Output*)

Simbol yang termasuk kedalam bagian *input-output* berkaitan dengan masukan dan keluaran. Simbol-simbol tersebut dapat dilihat pada tabel 2.3.

Tabel 2. 3 Simbol I/O (*Input-Output)*

| Simbol | Nama | Fungsi |
|---|---|---|
|  | *Input / Output* | Digunakan untuk menyatakan input dan output tanpa melihat jenisnya. |
|  | *Punched Card* | Digunakan untuk menyatakan masukan dan keluaran yang berasal dari card. |
|  | *Disk Storage* | Digunakan untuk menyatakan masukan dan keluaran yang berasal dari disk. |
|  | *Magnetic Tape* | Digunakan untuk menyatakan masukaan dan keluaran yang berasal dari pita magnetis |
|  | *Document* | Digunakan untuk menyatakan masukan dan keluaran yang berasal dari dokumen. |
|  | *Display* | Berfungsi untuk menyatakan keluaran melalui laya monitor |

## 2.8 Unified Modelling Language (UML)

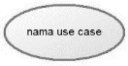
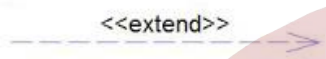
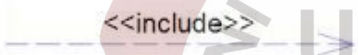
UML merupakan salah satu standart bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan mendokumentasikan sistem perangkat lunak (Hasanah, 2020). Jenis – jenis dari UML antara lain :

### 1) *Use Case* Diagram

*Use case* diagram adalah teknik untuk merekam persyaratan fungsional sebuah system, menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. *Use case* diagram menekankan kepada “apa” yang diperbuat oleh sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu,

misalnya login ke sistem, menambah *create* sebuah daftar belanja, dan sebagainya (Hasanah, 2020). Simbol-simbol *use case* diagram dapat dilihat pada Tabel 2.4.

Tabel 2. 4 Simbol dan Keterangan *Use Case Diagram*

| Simbol | Keterangan |
|---|---|
| Use Case   | Abstraksi dan interaksi antara sistem dan aktor. |
| Ekstend / extend   | Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri meski tanpa *use case* tambahan itu arah panah mengarah pada *use case* yang ditambahkan. |
| Include   | Relasi use case tambahan ke sebuah *use case* dimana *use case* yang ditambahkan membutuhkan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini arah panah *include* mengarah pada *use case* yang dipakai (dibutuhkan) atau mengarah pada *use case* tambahan. |
| Asosiasi / association  | Penghubung antara aktor dengan *Use Case* |
| Actor  | Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan *Use case.* |

**2) 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 (Hasanah, 2020). Tampilan *activity* diagram diagram dapat dilihat pada Tabel 2.5.

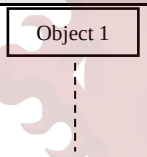
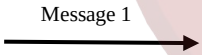
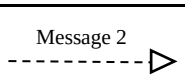
Tabel 2. 5 Simbol dan Keterangan *Activity Diagram*

| Simbol | Keterangan |
|---|---|
| Status Awal  | Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal. |
| Aktivitas  | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja. |
| Percabangan/decision  | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu. |
| Status Akhir  | Status akhir yang dilakukan system, sebuah diagram aktivitas memiliki sebuah status akhir. |

**3) *Sequence* Diagram**

Diagram Sequence adalah diagram yang dibuat untuk mengetahui alur dari interaksi antar objek (Sulianta, 2017). Tampilan *sequence* diagram dapat dilihat pada Tabel 2.6.

Tabel 2. 6 Simbol dan Keterangan *Sequence Diagram*

| Simbol | Nama | Keterangan |
|---|---|---|
|   | *Actor / Object* | Sebuah objek yang berasal dari kelas atau dapat dinamai dengan kelasnya saja. Aktor termasuk objek, Garis putus-putus menunjukkan garis hidup suatu objek. |
|  | *lifeline* | Menyatakan kehidupan suatu objek. |
|   | Pesan | Interaksi antara satu objek dengan objek lainnya. Objek dapat mengirimkan pesan ke objek lain. Interaksi antar objek ditunjukkan pada bagian operasi pada diagram kelas. |
|   | Return | Pesan kembalian dari komunikasi antar objek |
|  | Aktivasi | Menunjukkan masa hidup dari objek. |

## 2.9 Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasi lainnya dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, *SourceSafe* (Ruli, 2017).

## 2.10 Perumpamaan Domba dan Gembala

Alkitab mengajarkan kita tentang kasih Allah sang pencipta, Allah sang juru selamat dan Allah sang roh kudus. Perumpamaan kasih Allah tersebut dapat di gambarkan dari berbagai perumpamaan seperti perumpamaan Domba dan Gembala. Alkitab menggambarkan Allah yang maha pengasih sebagai Gembala yang baik, Gembala merupakan suatu pekerjaan yang menjaga dan merawat hewan ternak seperti domba. Seperti hal nya Allah yang menjaga dan merawat jemaat-Nya, tidak hanya memberi makan atau rejeki Allah juga membuktikan kasih-Nya kepada manusia melalui pengorbanan diri sebagai Yesus untuk menebus dosa manusia. Perumpamaan tersebut tertuang dalam Alkitab perjanjian baru di dalam injil Yohanes .

*Game* Gembala dan Domba Tersesat terbagi menjadi empat level dimana level pertama dan kedua berisikan penerapan Hukum Taurat yang pertama dimana Allah berperan sebagai Allah Sang Pencipta yang tertulis dalam kitab Keluaran 20:2-17 yang berisikan empat hukum yang mengatur hubungan manusia kepada Tuhan Allah dan level ketiga dan keempat berisikan penerapan Hukum Taurat yang kedua dimana Tuhan Allah berperan sebagai Allah Sang Pemelihara dan Juruselamat yang tertulis dalam kitab Ulangan 5:6-21 yang berisiskan enam hukum yang mengatur hubungan antar sesama manusia.

## 2.11 Pengujian Sistem

### a) *Black Box Testing*

*Black Box Testing* atau yang sering dikenal dengan sebutan pengujian spesifikasi fungsional merupakan metode pengujian untuk mengetahui apakah fungsi-fungsi masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Dalam pengujian ini, tester menyadari apa yang harus dilakukan oleh program tetapi tidak memiliki pengetahuan tentang bagaimana melakukannya (Hasanah, 2020).

### b) Beta

Pengujian Beta dilakukan setelah pengujian awal selesai, Pengujian Beta meliputi pengujian aplikasi ke pengguna secara langsung melalui beberapa responden ( Saputra dkk, 2022 ).

Rumus pengujian Beta :

$$Y = \frac{P}{Q} \times 100\%$$

Keterangan :

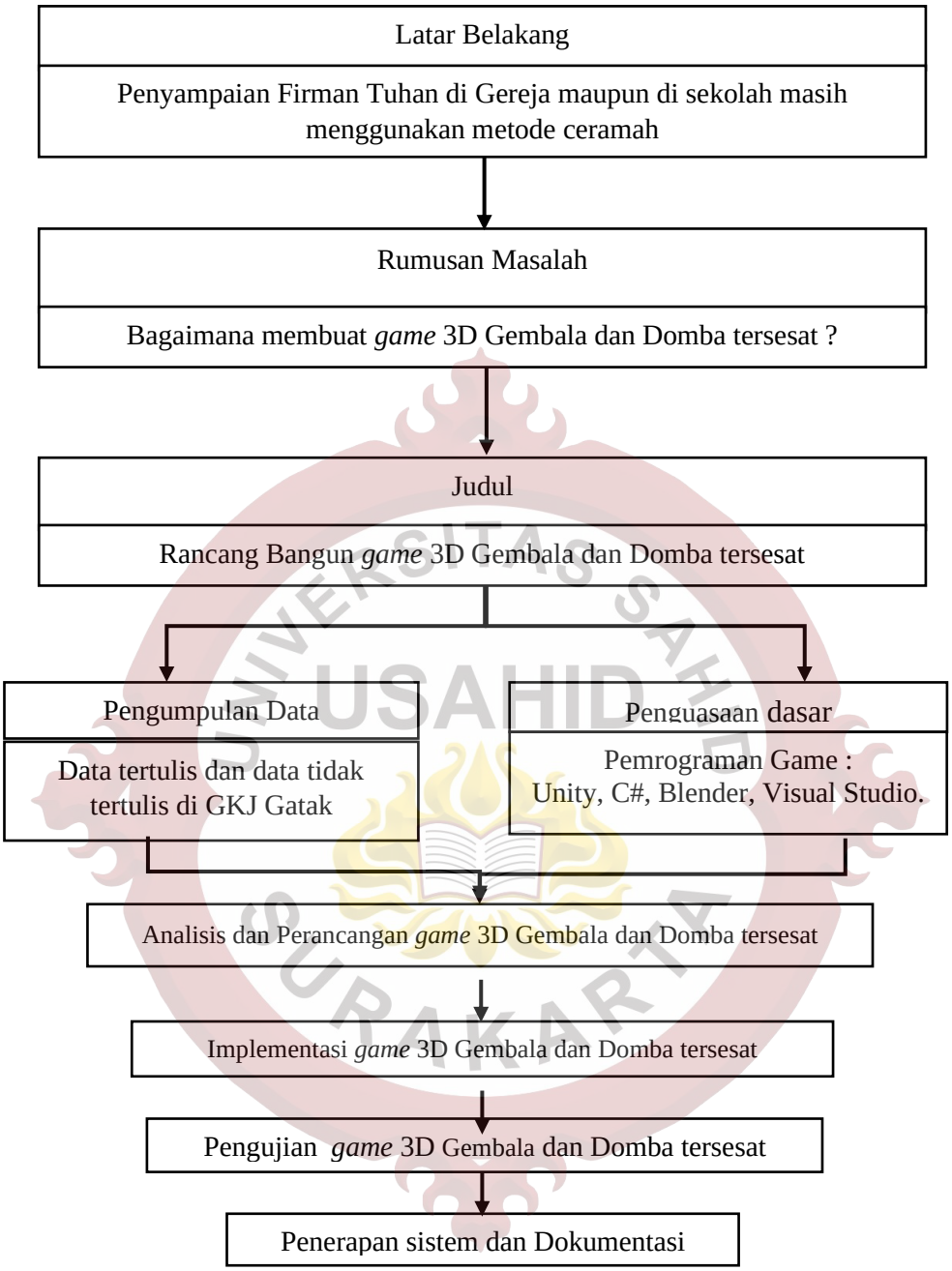
P : Banyaknya jawaban responden tiap soal

Q : Jumlah responden

Y : Nilai persentase

## 2.12 Kerangka Pemikiran

Berikut ini adalah tahapan kerangka pemikiran dalam melakukan analisis dan perancangan *game* 3D Gembala dan Domba tersesat. Kerangka pemikiran ini dapat dilihat pada Gambar 2.8.




Gambar 2. 8 Kerangka Pemikiran

Penjelasan dari kerangka pemikiran tersebut adalah :

1) Latar Belakang Masalah.

Penyampaian firman Tuhan di Gereja maupun di sekolah masih menggunakan metode ceramah.

2) Rumusan Masalah.

Rumusan masalah dari penelitian ini yaitu bagaimana membuat *game* 3D Gembala dan Domba tersesat ?

3) Judul

Judul yang di ambil dari latar belakang masalah dan rumusan masalah tersebut adalah rancang bangun *game* 3D Gembala dan Domba tersesat.

4) Pengumpulan data tertulis dan tidak tertulis

Pada penelitian dilakukan pengumpulan data secara tertulis dan tidak tertulis pada Gereja Kristen Jawa Gatak (GKJ Gatak). Pengumpulan data ini menggunakan metode observasi, wawancara dan studi pustaka.

5) Penguasaan dasar

Penguasaan dasar dalam pembuatan *game* 3D Gembala dan Domba tersesat akan membantu dalam melakukan pembuatan *game* 3D Gembala dan Domba tersesat meliputi : *game engine* Unity, bahasa pemrograman C#, Blender, Visual Studio.

6) Analisis dan Perancangan *game* 3D Gembala dan Domba tersesat

Ini merupakan hal yang paling utama dalam penelitian ini, menganalisis sesuai data yang di peroleh. Perancangan sistem menggunakan metode berorientasi objek (UML).

7) Implementasi *Game* 3D Gembala dan Domba tersesat.

Proses penerapan analisis dan desain sistem baik itu logika dan tampilan dengan menggunakan *game engine* Unity, bahasa pemrograman C#, *software* desain Blender dan kode editor Visual Studio yang akan menghasilkan *game* 3D Gembala dan Domba tersesat.

8) Pengujian *Game* 3D Gembala dan Domba tersesat.

Proses pengujian sistem secara internal dapat dilakukan baik itu secara verifikasi ataupun validasi data. Metode pengujian yang diambil adalah metode pengujian *BlackBox* dan pengujian beta. Pengujian *BlackBox* digunakan untuk menguji fungsi-fungsi khusus dari perangkat lunak yang dirancang. Pengujian beta digunakan untuk tanggapan pengguna terhadap aplikasi game, dengan melakukan kuesioner.

9) Penerapan sistem dan Dokumentasi

Sistem yang sudah diimplementasikan dan diuji coba kemudian di terapkan pada GKJ Gatak dan setelah itu di buatnya dokumentasi dari keseluruhan kegiatan penyusunan Tugas akhir.