BAB II

TINJAUAN PUSTAKA

2.1 Teori Dasar

Teori dasar adalah sekumpulan prinsip-prinsip atau konsep-konsep yang mendasari suatu ilmu atau disiplin ilmu tertentu. Teori dasar biasanya merupakan dasar dari suatu pemahaman yang lebih luas tentang suatu topik dan dapat digunakan sebagai dasar untuk mengembangkan teori atau hipotesis baru. Teori dasar juga sering digunakan sebagai dasar untuk menguji dan mengembangkan ide-ide baru atau untuk memprediksi hasil dari suatu eksperimen atau observasi.

2.1.1 Pengertian *Game*

Game adalah permainan media elektronik, hiburan multimedia, dan dibuat semenarik mungkin agar pemain bisa mendapatkan sesuatu yang disenangi. Bermain *game* adalah salah satu sarana pembelajaran. (Barros, Marisa, & Wijaya, 2018)

Tujuan dari pembuatan game yakni:

1. Sebagai sarana *Entertainment*
2. Sarana Mendidik
3. Penyampaian pesan
4. Melatih *Skill*

Dalam sebuah *game* kita harus menentukan sebuah tujuan dengan *goals/objective, goal* ini harus disampaikan dengan jelas dengan memberikan sebuah objektif log atau petunjuk agar pemain bisa menyelesaikan *objective* permainan dengan aman.(Hicks et al., 2016) *Level design* pada *game* juga sangat penting agar pemain akan tertarik dan tetap memainkan *game* tersebut.(Khalifa. 2016)

Dengan adanya *game* pendidikan yang diciptakan semenarik mungkin, anak tidak akan menyadari jika yang sedang dikerjakan itu ialah belajar dan anak akan menikmati proses belajar.

2.1.2 Game Puzzle

Gambar 2.1 *Minesweeper*
Sumber: https://en.wikipedia.org/wiki/Minesweeper_(video_game)

Setiap *genre* memiliki tantangannya tersendiri, dan beberapa pendekatan bekerja lebih baik daripada yang lain dalam pengaturan yang berbeda. Oleh karena itu relevan untuk mempertimbangkan pendekatan mana yang telah digunakan untuk pengujian strategi bermain.(Kristensen & Burelli, 2020)

Puzzle yang berarti teka-teki adalah genre permainan yang dimainkan dengan menyelasaikan sebuah rintangan berupa logika dan fisik untuk mencapai *goal*/*objektif* pada permainan tersebut.

Subgenre *puzzle game* adalah sebagai berikut :

1. *Physics* : Subgenre yang sangat popular berfokus pada *puzzle* menggunakan sebuah objek berupa fisika dan *gravity*.
2. *Coding* : Subgenre yang dimainkan seperti saat anda memprogramming sebuah elemen-elemen pada game tersebut.
3. *Exploration* : Biasanya digunakan pada game berupa detektif *point-and-click*. Terkadang *subgenre* ini menggunakan genre yang lain juga seperti *Maze* dan *Adventure*.
4. *Sokoban* : Subgenre ini menggunakan sebuah dimainkan dengan cara mendorong (atau dibeberapa permainan bisa ditarik) kotak (atau objek lain).
5. *Hidden Object* : Mencari sebuah objek-objek yang tersembunyi atau seperti *subgenre Exploration* dimana anda akan bermain seperti detektif point-and-click.
6. *Tile-Matching* : Subgenre yang sangat umum digunakan, *subgenre* ini dimainkan dengan cara menyatukan bentuk berderet (ada yang berwarna) atau warna yang sama seperti *Tetris* dan *Zuma*.

Subgenre yang kita gunakan tidak ada diatas melainkan subgenre permainan *puzzle* di nyata yaitu *Word Search*. *Word Search* merupakan game pencari kata

dimana kita menghubungkan sebuah kata, subgenre ini merupakan gabungan dari *Crossword* dan *Arrowword*.

2.1.3 *Android*

Gambar 2.2 *Android*
Sumber: https://en.wikipedia.org/wiki/Android_(operating_system)

Android adalah sistem operasi berbasis *Linux* yang dikembangkan oleh *Open Handset Alliance*, sebuah konsorsium industri yang didirikan oleh *Google*. *Android* dirancang untuk digunakan pada perangkat seluler, seperti smartphone dan tablet. Sistem operasi ini menawarkan beragam fitur, seperti aplikasi, kontak, kalender, dan lainnya, yang dapat di-*instal* atau di-*uninstall* oleh pengguna. *Android* juga memungkinkan pengguna untuk mengakses internet, mengirim pesan, dan melakukan telepon atau *video call*. Sistem operasi ini merupakan salah satu yang paling populer di dunia, dan telah terpasang pada hampir semua jenis perangkat seluler di berbagai negara.

2.2 Teori Khusus

2.2.1 *Adobe Illustrator*

Gambar 2.3 *Adobe Illustrator*
Sumber: https://blog.myskill.id/bidang-profesi/tools-adobe-illustrator

Aplikasi ini adalah sebuah software yang dikembangkan oleh sebuah perusahaan bernama *Adobe*. Aplikasi digunakan untuk membuat sebuah karya dan desain grafis digital. Karya dibuat berupa Ikon, dan ilustrasi, logo.

Berikut beberapa Fitur-fitur aplikasi ini adalah yaitu:

1. *Pen Tool* : *Tool* ini bisa menggambar semua bentuk atau freestyle dengan sebuah pen/pena.
2. *Pathfinder* : *Tool* ini digunakan untuk memodifikasi bentuk dari suatu benda gambar.
3. *Clipping Mask* : Digunakan untuk *copy* warna-warna atau *pattern* dari benda gambar.
4. *Type* : *Tool* yang digunakan untuk memberikan sebuah *Textbox* kemudian bisa diketik apa pun yang anda mau.

5. *Pattern Creator* : Fitur ini digunakan untuk membuat sebuah pola-pola yang sudah disiapkan oleh aplikasi ini.
6. *Live Trace* : Berfungsi untuk scan sebuah benda gambar kemudian gambar itu akan dipecah menjadi beberapa part.

2.2.2 *Unity*

Gambar 2.4 *Unity*
Sumber: https://en.wikipedia.org/wiki/Unity_%28game_engine%29

Software Game Engine yang dikembangkan oleh perusahaan bernama *Unity Technology*, dirilis pada tahun 2005 sebagai *Game Engine* yang dikhususkan untuk *Mac Os* di 2018, kemudian *Game Engine* dipublikasikan menjadi *Cross-Platform* setelah kemudian tahunnya.

Game Engine ini bisa digunakan pada OS *Windows*, *Linux*, dan *MacOS*. Sementara itu, *Software* yang dibuat dengan *Unity* ini juga bisa digunakan oleh berbagai *device* atau *platform* seperti PC dan Smartphone, contohnya :

1. *Smartphone* : IOS dan *Android*
2. *Deskop* : *Windows*, *Linux*, dan *Mac*
3. Webgl
4. Konsol/*Console* : *PS4*, *PS5*, *Nintendo Switch*, *Stadia*, *Xbox One* dan *Xbox Series X/S*
5. Platform VR : *Steam* VR, *ARCore Google*, *HOLOLens*, dan *Playstation VR*

Software ini biasanya digunakan untuk mengembangkan sebuah *Game*. *Game* yang dikembangkan pada umumnya adalah *Game* bergrafis 3D dengan menggunakan *Unity* 3D, tetapi juga bisa digunakan untuk membuat *game* bergrafis 2D bahkan menggabungkan kedua grafis tersebut juga bisa dikembangkan.

Bahasa pemograman yang digunakan pada umumnya *Unity* ini adalah C#. Tetapi, juga bisa menggunakan bahasa pemograman lainnya seperti C++, *Rust*, *IronPython*, *Lua*, dan bahasa pemograman ekslusif untuk *Game Engine* ini yang mirip dengan *JavaScript* yaitu *UnityScript*.

Fitur-fitur Unity adalah sebagai berikut:

1. *Rendering*

Rendering di *Unity* memiliki fungsi lain seperti untuk *Parallax Mapping*, *Bump Mapping*, dan *Reflection Mapping*. Di *Unity ShaderLab* ini digunakan sebagai bahasa pemograman untuk shader pada grafis yang digunakan. Graphics yang digunakan tergantung platform apa yang anda gunakan, contohnya *Windows* dan *MacOS* menggunakan *OpenGL*, *Android* dan IOS menggunakan *OpenGL ES*, dan *Direct3D* digunakan oleh *XBOX*.

2. *Scripting*

Script-nya dibuat menggunakan *Mono* (Implementasi *open-source* oleh *.NET Framework*). *UnityScript*(Seperti *JavaScript* dan *ECMAScript*), C# atau *Boo* (Seperti *syntax Python*) digunakan sebagai bahasa pemograman umumnya.

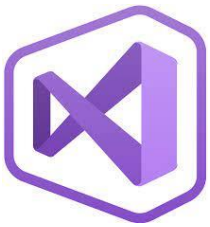
3. *Asset Tracking*

Server Unity Asset sebagai manajemen untuk *script* dan asset *game*. *PostgreSQL* digunakan oleh *server* sebagai *BackEnd*, *audio* akan dikumpulkan dan dibuat oleh *FMOD Library*, *VideoPlayback* menggunakan *Theora codec*, dan *RakNet* digunakan sebagai *MultiPlayer Networking*.

4. *Asset Store*

Dirilis pada tahun 2010, Store di gunakan untuk serana Jual-Beli *asset* yang akan digunakan di *Unity Editor*.

2.2.3 Visual *Studio*



Gambar 2.5 Visual *Studio*
Sumber: (Enterprise, 2019)

Visual *Studio* adalah sebuah *Integrated Development Environment* (IDE) yang dikembangkan oleh *Microsoft*. IDE ini digunakan oleh para pengembang perangkat lunak untuk membuat, mengedit, dan mengelola kode dalam berbagai bahasa pemrograman, seperti *C#*, *C++*, dan Visual *Basic*. Visual *Studio* menyediakan beragam fitur dan alat yang membantu dalam proses pengembangan perangkat lunak, termasuk penyorotan sintaksis, *debugging*, *refactoring*, dan pengelolaan kode sumber. (Enterprise, 2019)

Visual *Studio* juga mendukung pengembangan aplikasi untuk berbagai platform, termasuk *Windows, Android, iOS, web,* dan *cloud*. IDE ini menyediakan *template* proyek yang siap pakai untuk berbagai jenis aplikasi, seperti aplikasi desktop, aplikasi *web*, aplikasi *mobile*, permainan, dan lain-lain. Visual *Studio* juga terintegrasi dengan berbagai layanan dan platform *Microsoft*, seperti *Azure* untuk pengembangan *cloud*, *SQL Server* untuk pengelolaan basis data, dan *Team Foundation Server* untuk kolaborasi tim.

Selain fitur dan dukungan yang kuat untuk pengembangan perangkat lunak, Visual *Studio* juga memiliki antarmuka pengguna yang intuitif dan mudah digunakan. IDE ini menyediakan berbagai tata letak dan tema yang dapat disesuaikan sesuai dengan preferensi pengembang. Visual *Studio* juga memiliki kemampuan untuk mengintegrasikan dengan alat pengembangan pihak ketiga dan memperluas fungsionalitasnya melalui ekstensi.

Dalam rangka memaksimalkan penggunaan Visual *Studio*, *Microsoft* juga menyediakan dokumentasi, tutorial, dan sumber daya lainnya yang membantu pengembang mempelajari dan menguasai IDE ini. Visual *Studio* telah menjadi salah *satu* IDE yang populer dan banyak digunakan oleh pengembang perangkat lunak di seluruh dunia. Bahasa pemrograman yang gunakan pada penelitian ini adalah *C#* dan *UnityScript* dengan bantuan dari beberapa fitur dari *Unity Engine*.

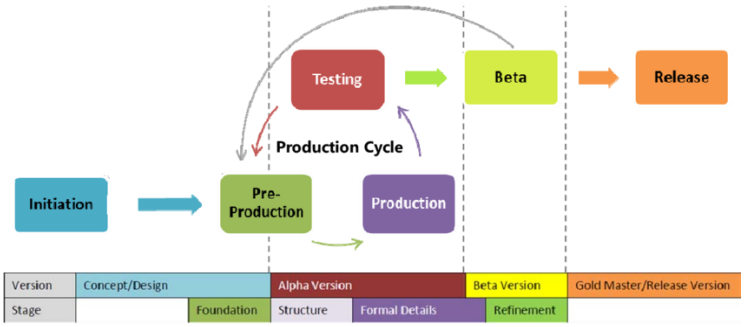
2.2.4 Visual *Paradigm*

Gambar 2.6 *Visual Paradigm*
Sumber: https://www.visual-paradigm.com/

UML merupakan sebuah aplikasi pemodelan, dimana pemodelannya digunakan untuk merancang visual diagram, dirilis pada tahun 1997 dikembangkan oleh *Object Management Group*.

UML *Tool* yang akan saya gunakan untuk menggambar perancangan diagram adalah Visual *Paradigm*. Tool ini merupakan salah satu *tool* UML yang sangat praktis dan sederhana untuk digunakan, dan juga memiliki fitur-fitur yang sangat membantu dalam pembuatan diagram.

2.2.5 GDLC (*Game Development Life Cycle*)



Gambar 2.7 *Game Development Life Cycle*
Sumber: (Krisdiawan, 2018)

Menuru (Krisdiawan, 2018) GDLC adalah suatu proses pengembangan sebuah game yang menerapkan pendekatan iteratif yang terdiri dari 6 fase pengembangan, dimulai dari fase inisialisasi/pembuatan konsep, *preproduction*, *production*, *testing*, *beta* dan *release*. GDLC merupakan sebuah metode yang menangani pengembangan game dimulai dari titik awal hingga paling akhir.

Berikut adalah penjelasan singkat mengenai fase-fase dalam GDLC:

1. *Initiation* : Pada fase ini, ide dasar atau konsep game ditentukan. Hal ini meliputi pemilihan genre, tema, target audiens, dan fitur-fitur utama yang akan ada dalam *game.*
2. *Pre-production* : Fase ini melibatkan penentuan tujuan, sasaran, dan ruang lingkup proyek *game*. Di sini, tim pengembang juga mengidentifikasi aset yang diperlukan dan pembuatan desain rinci dari aspek-aspek *game*, seperti desain *level*, desain karakter, mekanisme *gameplay*, tata letak visual, dan suara. Tujuan dari fase ini adalah untuk menciptakan panduan yang jelas bagi pengembang dalam membangun *game*.
3. *Production* : Fase produksi adalah tahap implementasi nyata dari desain yang telah dibuat. Tim pengembang bekerja untuk membuat *asset*, mengodekan logika permainan, membuat *level*, menyusun animasi, dan mengintegrasikan semua elemen *game* menjadi satu kesatuan.
4. *Testing* : Fase pengujian adalah tahap di mana *game* diuji secara menyeluruh untuk menemukan *bug*, kesalahan, atau masalah lainnya. Tes ini melibatkan permainan yang intensif, pengujian fungsional, dan pengujian kinerja untuk

memastikan *game* berjalan dengan baik dan memberikan pengalaman yang baik bagi pemain.

5. *Beta* : Fase *Beta* terjadi setelah fase pengujian internal dan sebelum peluncuran resmi *game* Fase *Beta* penting dalam GDLC karena memberikan kesempatan bagi pengembang untuk mendapatkan umpan balik langsung dari pengguna sebelum peluncuran resmi *game*. Dengan melibatkan peserta *beta,* pengembang dapat mengidentifikasi dan memperbaiki masalah yang belum terdeteksi sebelumnya, meningkatkan pengalaman pengguna, dan memastikan kualitas game yang lebih baik saat diluncurkan ke publik.
6. *Release* : Setelah semua perbaikan dan pembaruan telah selesai dan pengujian *internal* dan *beta* telah dijalankan tanpa ada masalah, *game* siap untuk peluncuran resmi. Fase *Beta* berakhir dan *game* dapat dirilis ke publik atau pasar yang dituju.

2.2.6 UML (*Unified Modeling Language*)

Unified Modelling Language merupakan alat perancangan sistem yang berorientasi pada objek. Secara filosofi kemunculan UML diilhami oleh konsep yang telah ada yaitu konsep permodelan *Object Oriented* (OO), karena konsep ini menganalogikan sistem seperti kehidupan nyata yang didominasi oleh obyek dan digambarkan atau dinotasikan dalam simbol-simbol yang cukup spesifik maka OO memiliki proses standard dan bersifat independen.

Unified Modeling Language (UML) adalah sebuah bahasa standar yang digunakan untuk mendokumentasikan, merancang, dan mengkomunikasikan desain perangkat lunak. UML menyediakan notasi grafis yang dapat digunakan untuk

menggambarkan berbagai aspek dari sistem yang akan dibangun, termasuk struktur, fungsi, interaksi antar komponen, dan proses bisnis.

1. *Use Case* Diagram

Use Case Diagram adalah salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang digunakan untuk menggambarkan interaksi antara aktor (pengguna) dan sistem yang sedang dikembangkan. *Use case* diagram membantu dalam memahami kebutuhan fungsional sistem dengan mengidentifikasi dan menggambarkan berbagai skenario penggunaan.

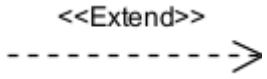
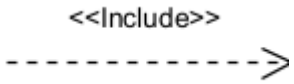
Dalam *Use Case* diagram, aktor digambarkan sebagai entitas luar yang berinteraksi dengan sistem, seperti pengguna, sistem eksternal, atau perangkat keras. Sedangkan *Use Case* (kasus penggunaan) menggambarkan aksi atau fungsi-fungsi yang dapat dilakukan oleh sistem dalam respons terhadap interaksi dengan aktor.

Beberapa elemen yang ada dalam *Use Case* diagram antara lain:

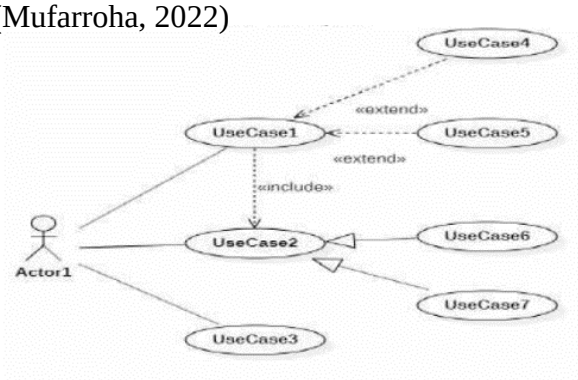
Tabel 2.1 Simbol *Use Case* Diagram

Simbol	Deskripsi
Aktor 	Merepresentasikan entitas luar yang berinteraksi dengan sistem. Aktor digambarkan sebagai ikon manusia atau blok sederhana

Use Case 	Menggambarkan fungsi atau aksi yang dapat dilakukan oleh sistem dalam respons terhadap interaksi dengan aktor.
Association 	Menggambarkan hubungan antara aktor dan *use case*. *Association* menunjukkan bahwa aktor terlibat dalam satu atau lebih use case, atau bahwa *use case* menggunakan atau berinteraksi dengan aktor tertentu.
Generalization 	Menunjukkan bahwa *actor* atau *use case* anak adalah bentuk khusus dari *actor* atau *use case* induk, mewarisi atribut dan perilaku dari induknya.
System Boundary 	Menunjukkan batasan atau ruang lingkup sistem yang sedang dianalisis

Extend 	Digambarkan sebagai panah dengan label "«*extend*»" yang menghubungkan *use case* yang melakukan ekstensi dengan *use case* yang di-*extend*. Ini menunjukkan bahwa *use case* tersebut dapat memperluas fungsionalitas *use case* lain dalam situasi tertentu
Include 	Digambarkan sebagai panah dengan label "«*include*»" yang menghubungkan *use case* induk dengan *use case* yang di-*include*. Ini menunjukkan bahwa use case yang di-*include* akan selalu terjadi sebagai bagian dari use case induk

Sumber: (Mufarroha, 2022)



Gambar 2.8 *Use Case* Diagram
Sumber: (Mufarroha, 2022)

Use Case diagram membantu dalam memahami perspektif pengguna sistem dan mengidentifikasi fungsi-fungsi yang harus ada dalam sistem. Diagram ini juga membantu dalam komunikasi antara tim pengembang perangkat lunak dan pemangku kepentingan lainnya dalam proyek pengembangan perangkat lunak.

2. *Activity* Diagram

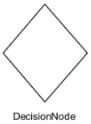
Activity diagram adalah salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang digunakan untuk menggambarkan alur kerja atau urutan aktivitas dalam suatu proses. *Diagram* ini digunakan untuk memodelkan bagaimana objek dan elemen sistem berinteraksi dalam rangka mencapai tujuan yang diinginkan.

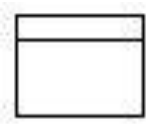
Dalam *Activity Diagram*, aktivitas direpresentasikan sebagai tindakan atau aksi yang dilakukan oleh objek dalam sistem. Alur aktivitas tersebut digambarkan sebagai urutan langkah-langkah atau tindakan-tindakan yang dihubungkan dengan panah. Aktivitas juga dapat memiliki keadaan awal (*start state*) dan keadaan akhir (*end state*) yang menunjukkan dimulainya dan berakhirnya aktivitas tersebut.

Beberapa elemen yang ada dalam *Activity Diagram* antara lain:

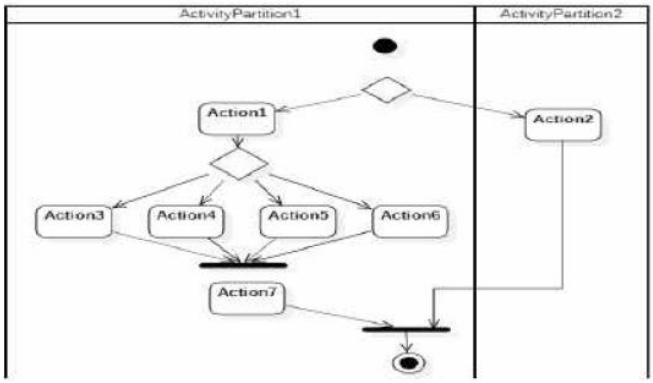
Tabel 2.2 Simbol *Activity* Diagram

Simbol	Deskripsi
Activity 	Merupakan tindakan atau aksi yang dilakukan oleh objek dalam sistem.

	Aktivitas direpresentasikan sebagai persegi panjang.
Initial Node 	Menunjukkan awal dari alur aktivitas. Keadaan awal direpresentasikan sebagai lingkaran dengan panah masuk.
Final Node 	Menunjukkan akhir dari alur aktivitas. Keadaan akhir direpresentasikan sebagai lingkaran dengan panah keluar
Decision 	Mewakili titik dalam alur kerja di mana keputusan diambil berdasarkan kondisi tertentu. Biasanya digambarkan sebagai berlian dengan panah masuk dan keluar
Join	Mewakili penggabungan jalur alur kerja yang terpisah menjadi satu jalur. Biasanya digambarkan sebagai garpu atau tanda panah yang menyatukan jalur alur kerja menjadi satu
Swimlane	Konsep yang digunakan untuk membagi dan mengelompokkan

	aktivitas-aktivitas dalam diagram berdasarkan aktor atau entitas yang terlibat. *Swimlane* secara visual membagi diagram menjadi beberapa kolom yang mewakili setiap *swimlane*.

Sumber: (Mufarroha, 2022)



Gambar 2.9 *Activity* Diagram
Sumber: (Mufarroha, 2022)

Activity Diagram membantu dalam memvisualisasikan dan memahami alur kerja atau proses yang terjadi dalam sistem. Diagram ini dapat digunakan untuk menganalisis dan merancang proses pengembangan, mengidentifikasi langkah-langkah yang diperlukan dalam suatu tugas, dan memodelkan alur kontrol dalam sistem. *Activity Diagram* juga dapat membantu dalam komunikasi antara pengembang dan pemangku kepentingan dalam proyek pengembangan perangkat lunak.

3. *Sequence* Diagram

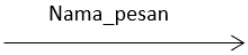
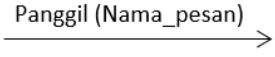
Sequence diagram adalah salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang digunakan untuk menggambarkan interaksi antara objek-objek dalam suatu sistem. Diagram ini menunjukkan urutan pesan atau pemanggilan metode antara objek-objek tersebut, serta urutan eksekusi dari tindakan-tindakan yang dilakukan oleh objek dalam sistem.

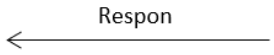
Dalam *sequence* diagram, objek-objek direpresentasikan sebagai kotak persegi panjang dan pesan-pesan antara objek-objek direpresentasikan sebagai panah yang mengarah dari objek pengirim ke objek penerima. Urutan pemanggilan metode ditunjukkan dengan urutan panah-panah yang menghubungkan objek-objek. Diagram ini juga dapat menunjukkan kembalinya nilai dari metode yang dipanggil, serta pemanggilan metode yang bersamaan atau secara paralel.

Beberapa elemen yang ada dalam *Sequence* Diagram antara lain:

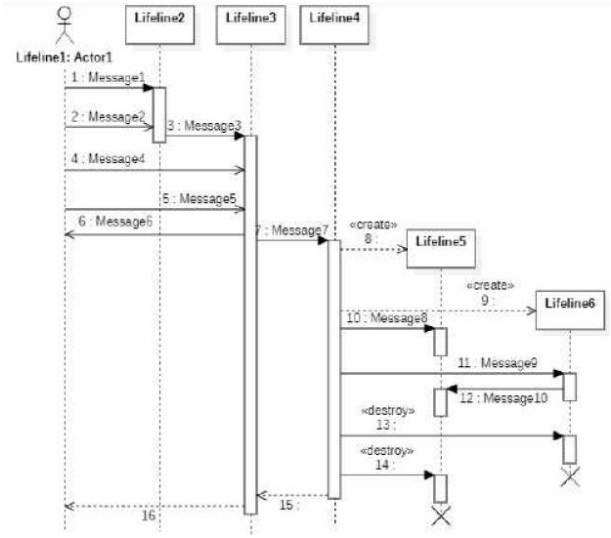
Tabel 2.3 Simbol *Sequence* Diagram

Simbol	Deskripsi
Aktor 	Simbol aktor digunakan untuk mewakili entitas eksternal yang berinteraksi dengan sistem. Aktor dapat berupa pengguna, perangkat, sistem lain, atau entitas lain yang

	terlibat dalam interaksi dengan sistem yang sedang dianalisis.
Objek 	Simbol objek digunakan untuk mewakili *instance* kelas atau objek dalam sistem.
Lifeline	Hidup (*lifeline*) menggambarkan keberadaan objek selama interaksi dalam *Sequence* Diagram.
Pesan 	Pesan digunakan untuk menggambarkan komunikasi atau pemanggilan metode antara objek-objek dalam *Sequence* Diagram. Pesan dapat berupa pesan yang dikirim dari satu objek ke objek lain, pemanggilan metode, atau tanggapan terhadap pesan sebelumnya.
Method Call 	Simbol pemanggilan metode digunakan untuk menunjukkan pemanggilan metode dari satu objek ke objek lain.

Response  	Tanggapan digunakan untuk menunjukkan respons atau balasan dari sebuah pesan sebelumnya.

Sumber: (Mufarroha, 2022)



Gambar 2.10 *Sequence* Diagram
Sumber: (Mufarroha, 2022)

Sequence diagram membantu dalam memodelkan dan memahami interaksi antara objek-objek dalam sistem secara kronologis. Diagram ini berguna dalam menganalisis dan merancang alur eksekusi dari tindakan-tindakan yang dilakukan oleh objek, serta menggambarkan hubungan dan ketergantungan antara objek-objek tersebut. Sequence diagram juga dapat digunakan untuk mendokumentasikan desain perangkat lunak dan berkomunikasi antara tim pengembang dalam proyek pengembangan perangkat lunak.

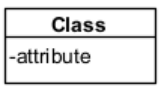
4. *Class* Diagram

Class diagram adalah salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang digunakan untuk menggambarkan struktur statis dari suatu sistem perangkat lunak. Diagram ini menunjukkan kelas-kelas yang ada dalam sistem, atribut-atribut yang dimiliki oleh setiap kelas, serta hubungan dan asosiasi antara kelas-kelas tersebut.

Selain itu, *Class* Diagram juga menunjukkan hubungan antara kelas-kelas dalam sistem, seperti hubungan asosiasi, agregasi, komposisi, generalisasi, dan ketergantungan. Hubungan asosiasi digambarkan dengan garis lurus antara kelas-kelas yang terhubung, sementara hubungan agregasi dan komposisi ditunjukkan dengan panah dan berbagai simbol untuk menunjukkan hubungan "memiliki" antara kelas-kelas. Hubungan generalisasi digambarkan dengan panah yang mengarah dari kelas yang lebih spesifik ke kelas yang lebih umum, menunjukkan pewarisan atau hubungan "adalah bagian dari". Ketergantungan antara kelas-kelas ditunjukkan dengan garis putus-putus atau panah putus-putus.

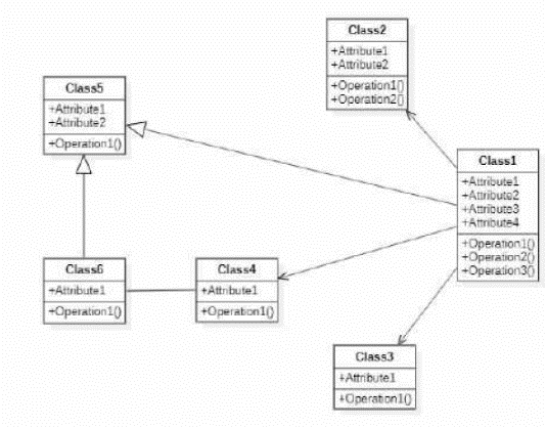
Beberapa elemen yang ada dalam *Class* diagram antara lain:

Tabel 2.4 Simbol *Class* Diagram

Simbol	Deskripsi
Class 	Simbol kelas digunakan untuk menggambarkan kelas dalam sistem.

	Kelas menggambarkan struktur dan perilaku objek dalam sistem.
Generalization	Generalisasi dalam *Class* Diagram menggambarkan hubungan "is-a" atau pewarisan antara kelas-kelas. Generalisasi menunjukkan bahwa suatu kelas adalah turunan atau merupakan spesialisasi dari kelas lain yang lebih umum.
Association	Hubungan asosiasi digunakan untuk menggambarkan hubungan antara dua kelas. Hubungan ini menunjukkan bahwa suatu kelas memiliki ketergantungan terhadap kelas lain.
Dependency 	Hubungan ketergantungan digunakan untuk menggambarkan ketergantungan antara kelas-kelas. Hubungan ini menunjukkan bahwa suatu kelas membutuhkan kelas lain dalam konteks tertentu.

Sumber: (Mufarroha, 2022)



Gambar 2.11 *Class* Diagram
Sumber: (Mufarroha, 2022)

Class Diagram membantu dalam pemodelan struktur perangkat lunak dengan jelas dan memahami hubungan antara kelas-kelas dalam sistem. Diagram ini berguna dalam analisis, perancangan, dan dokumentasi perangkat lunak, serta membantu komunikasi antara tim pengembang dalam pengembangan sistem. *Class* diagram juga digunakan sebagai dasar untuk menghasilkan kode program yang terstruktur dan memastikan konsistensi antara desain dan implementasi perangkat lunak.

2.3 Penelitian Terdahulu

1. Penelitian **Dewa Ayu Indah Cahya Dewi dan Santi Ika Murpratiwi** dengan judul ***Game Development of "Kwace Adat Bali" for The Socialization of Balinese Traditional Dress-Up Ethics*.** Hasil dari penelitian ini adalah *game* edukasi yang diharapkan sebagai sarana sosialisasi dalam etika berpakaian tradisional Bali yang sesuai bagi generasi muda. Dalam permainan ini, gaya berpakaian tradisional Bali

diklasifikasikan menjadi tiga jenis, yaitu pakaian tradisional ringan (payas alit), pakaian tradisional menengah (payas madya), dan pakaian tradisional besar (payas agung) yang dikombinasikan dengan algoritma acak '*shuffle'* untuk mengacak *item* permainan. (Dewi & Murpratiwi, 2020)

2. Penelitian **Rizki Alief Wicaksana dan Hotma Pangaribuan** dengan judul Rancang Bangun Aplikasi ***Game*** **Edukasi Pengenalan Huruf Alfabet dengan Teknologi *Augmented Reality* Berbasis *Android***. Hasil Penelitian yang yang didapatkan adalah dalam TK Hang Nadim *Malay School*, terdapat penggunaan media pembelajaran berupa *game* edukasi pengenalan alfabet yang bertujuan untuk meningkatkan minat belajar anak-anak. *Game* edukasi tersebut dirancang secara interaktif dan mudah dipahami oleh para murid, sehingga dapat menjadi media pembelajaran yang efektif dan menarik. (Wicaksana & Pangaribuan, 2020)
3. Penelitian **Willyanto Diharjo** dengan judul **Game Edukasi Bahasa Indonesia menggunakan Metode *Fisher Yates Shuffle* Pada Genre *Puzzle*.** Hasil dari penelitian ini adalah *game puzzle* yang menggunakan metode *Yates Shuffle* berguna untuk dimainkan kalangan anak-anak untuk memahami sinonim, antonim, akronim dan homonim. (Diharjo, Sani, & Arif, 2020)
4. Penelitian **Emanuel Pratalaharja dan Bayu Prakoso Dirgantoro** dengan judul ***Reintroducing Indonesian Traditional Games through an Interactive Multiplayer Table Game*** **Gobak Sodor.** Hasil dari penelitian ini adalah *game multiplayer* interaktif yang bertujuan untuk mengenalkan

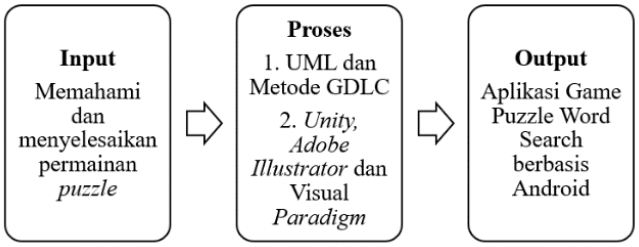
kembali tentang permainan tradisional Gobak Sodor.(Pratalaharja & Dirgantoro, 2021)

5. Penelitian **Markani Kania** dengan judul **Aplikasi *Game* Edukasi *Puzzle* dengan Kecerdasan Buatan berbasis *Android*.** Hasil dari penelitian ini adalah *game puzzle* berbasis android yang diciptakan dengan menggunakan *Unity* 3D dengan control *Finite State Machine*. (Kania, 2021)
6. Penelitian **Bayu Ryanto, Nining Rahaningsih dan Ade Irma Purnamasari** dengan judul **Game *Puzzle* Hewan uutuk Peningkatan Kemampuan Bahasa Inggris Siswa *PAUD* Menggunakan Metode *Addie***. Hasil dari penelitian ini adalah terciptanya *game puzzle* hewan untuk meningkatkan kemampuan dalam berbahasa inggris menggunakan metode *ADDIE*. (Ryanto, Rahaningsih, & Purnamasari, 2022)
7. Penelitian **Ardiawan Bagus Harisa, Zahrotul Umami dan Filmada Ocky Saputra** dengan judul ***Board Game Design for Conservation Organization*.** Hasil dari penelitian ini adalah game edukasi *Game Board* yang digunakan sebagai sarana pembelajaran untuk belajar dan memahami tentang Elang Jawa. (Harisa, Umami, & Saputra, 2022)
8. Penelitian **Ahmad Ade Nugraha dan Alvanov Zpalanzani Mansoor** dengan judul ***Designing Educational Games as Learning Media for* Lontara/Bugis *Script for Children 7-8 Years old*.** Hasil dari penelitian ini adalah *game edukasi* yang dikembangkan sebagai media pembelajaran untuk meningkatkan semangat belajar anak dan kemampuan dalam

mempelajari tentang Aksara Lontara untuk pengguna *smartphone* di Indonesia, termasuk di kalangan anak-anak. (Nugraha & Mansoor, 2022)

9. Penelitian **Andi, Juan Charles, Octara Pribadi, Carles Juliandy dan Robet** dengan judul ***Game Development "Kill Corona Virus" for Education About Vaccination Using Finite State Machine and Collision Detection*****.** Hasil dari penelitian ini adalah game edukasi yang dikembangkan dengan menggunakan *Game Development Life Cycle* (GDLC). *Game* ini bertujuan untuk masyarakat mempelajari tentang pentingnya vaksinasi dalam mencegah infeksi virus corona, sehingga perlu mengembangkan sebuah permainan yang memberikan edukasi agar masyarakat tertarik untuk melakukan vaksinasi. (Charles, Pribadi, & Juliandy, 2022)
10. Penelitian **Cinde Yuliga dan Tito Pinandita** dengan judul ***The Development of Android-Based Plane Figure Educational Game using Unity 2D with Fisher Yates Shuffle Algorithm (A Case Study at* Sekolah Dasar Negeri 1 Brobot)**. Hasil dari penelitian ini adalah permainan edukatif berbasis *android* untuk bangunan datar menggunakan *Unity 2D* dengan algoritma *Fisher Yates shuffle* telah dikembangkan secara sistematis dengan menggunakan model *ADDIE* sesuai dengan cabang yang terdiri dari lima tahap. Permainan edukatif ini mendapatkan tanggapan positif dari siswa dan guru kelas IV SD Negeri 1 Brobot. (Cinde & Pinandita, 2023)

2.4 Kerangka Pemikiran



Gambar 2.8 Kerangka Pemikiran
Sumber: Data Penelitian

Kerangka pemikiran adalah struktur atau pola pikir yang menjadi dasar pemikiran dan merupakan landasan bagi penulisan suatu karya ilmiah, seperti skripsi, tesis, atau laporan penelitian. Kerangka pemikiran merupakan bagian penting dari suatu karya ilmiah karena membantu menjelaskan secara sistematis dan logis bagaimana penelitian atau analisis tersebut dilakukan dan mengapa peneliti melakukannya.

Dari kerangka pemikirian diatas dimulai dengan permasalahan memahami dan menyelesaikan puzzle. Setelah itu masalah tersebut diproses dengan menggunakan UML dan Metode GDLC, kemudian menggunakan *Unity*, *Adobe Illustrator*, dam Visual Paradigm untuk menghasilkan *Game Puzzle Word* Search berbasis *Android*. Aplikasi yang dihasilkan berupa permainan teka-teki mencari kata yang praktis dan mudah dimainkan.