BAB 2
LANDASAN TEORI

2.1 Permainan Kukuyaan

Kukuyaan yang artinya Kura-Kura, permainan Kukuyaan adalah jenis permainan tradisional khas Jawa Barat dimana dalam permainan ini menggunakan media air layaknya arung jeram, tetapi permainan tradisional khas Jawa Barat ini menggunakan ban bukan perahu karet. Orang yang bermain Kukuyaan harus terlentang di atas ban untuk mengalun di sungai, dengan tangan mengayuh seperti dayung, seperti kura-kura [1].

Permainan ini dilakukan sebagai media sosialisai untuk melakukan pembersihan sampah disungai. pembersihan sungai dilakukan karena kurangnya kesadaran masyarakat sekitar akan kebersihan lingkungan daerah sungai Cikapundung karena selalu membuang sampah dan limbah rumah tangga langsung kesungai, maka mereka mengadakan kegiatan Cikapundung Rehabilitasi Program yang awalnya dilakukan hanya didaerah sungai Cikapundung saja yang merupakan program dari perhimpunan Penempuh Rimba dan Pendaki Gunung, namun sekarang ini kegiatan ini sudah mulai didukung dari berbagai pihak maupun pemerintahan. Kegiatan kukuyaan ini pun dilakukan bukan hanya di daerah sungai Cikapundung namun akan diterapkan kegiatan yang serupa di sungai Citarum dan Ciliwung yang dianggap sungainya telah tercemar. Cara memainkan permainan ini adalah dengan cara mengarungi sungai menggunakan ban dalam bekas sambil memungut sampah disepanjang jalur yang dilewati [3].

Demi menjaga kebersihan sungai. Pemerintah Kota Bandung menerapkan Peraturan Daerah Kota Bandung no 11 tahun 2004. Bagi yang membuang sampah ke Cikapundung, maka akan dikenakan denda Rp 5 juta bagi individu, Rp 50 juta untuk perusahaan dan dikenakan ancaman kurungan 3 bulan penjara [3].

2.2 Permainan (*Game*)

Permainan merupakan sebuah aktivitas rekreasi dengan tujuan bersenang-senang ataupun untuk mengisi waktu luang. Permainan biasanya dilakukan secara sendiri-sendiri ataupun secara kelompok.

2.2.1 Klasifikasi Permainan

Sejarah *game* tidak lepas dengan namanya *platform*, dari yang awalnya hanya mesin *Arcade* (Mesin ding dong) hingga sampai sekarang ini yang sudah mencapai era *Mobile Games*. Untuk lebih jelasnya, berikut ini adalah klasifikasi *game* berdasarkan jenis *Platform* yang di gunakan [6]:

1. *Arcade Games* yaitu yang sering disebut ding-dong di Indonesia, biasanya berada di daerah / tempat khusus dan memiliki box atau mesin yang memang khusus didesain untuk jenis *video games* tertentu dan tidak jarang bahkan memiliki fitur yang dapat membuat pemainnya lebih merasa "masuk" dan menikmati. Seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil (beserta transmisinya).
2. *PC Games* (*personal computer*) yaitu *video game* yang dimainkan menggunakan *Personal Computers*.
3. *Console Games* yaitu *video game* yang dimainkan menggunakan *console* tertentu seperti *playstation XBOX, Nitendo*, dan lan-lain.
4. *Handheld Game* yaitu *game* yang dimainkan di *console* khusus *video game* yang dapat dibawa kemana-mana, contoh *nitendo DS dan Sony PSP*.
5. *Mobile Games* yaitu *game* yang biasa dimainkan khusus untuk *handphone atau PDA*.

Klasifikasi permainan berdasarkan pembatasan usia dari pemainnya, menurut ***ESRB*** (***Entertainment Software Rating Board***) adalah sebagai berikut [7]:

1. ***Early Childhood (3+)*** memiliki isi yang mungkin cocok untuk usia 3 keatas. Tidak mengandung materi yang tidak pantas seperti adegan kekerasan.

2. ***Everyone (6+)*** memiliki konten yang mungkin cocok untuk usia 6 keatas. Judul-judul dalam kategori ini mungkin berisi sedikit kartun, fantasi atau kekerasan ringan dan tidak jarang menggunakan bahasa yang ringan.
3. ***Everyone (10+)*** memiliki konten yang mungkin cocok untuk usia 10 keatas. Judul-judul dalam kategori ini mungkin berisi lebih banyak kartun, fantasi atau kekerasan ringan, bahasa ringan dan juga minimal tema sugestif.
4. ***Teen (13+)*** memiliki konten yang mungkin cocok untuk usia 13 keatas. Judul-judul dalam kategori ini mungkin mengandung kekerasan, tema sugestif, humor kasar, sedikit darah, simulasi perjudian, dan tidak jarang menggunakan bahasa yang kuat.
5. ***Mature (17+)*** memiliki konten yang mungkin cocok untuk orang usia 17 dan lebih tua. Judul-judul dalam kategori ini mungkin berisi *intens* kekerasan, darah, konten seksual dan bahasa yang kuat. *Mature* (17+) memiliki konten yang mungkin cocok untuk orang usia 17 dan lebih tua. Judul judul dalam kategori ini mungkin berisi *intens* kekerasan, darah, konten seksual dan bahasa yang kuat.
6. ***Adults Only (18+)*** memiliki konten yang hanya boleh dimainkan oleh orang-orang 18 tahun keatas. Judul-judul dalam kategori ini mungkin termasuk adegan yang *intens*, kekerasan dan juga mengandung konten seksual.
7. ***RP* (*Rating Pending*)** telah disampaikan kepada ESRB dan sedang menunggu penilaian akhir. (Simbol ini hanya muncul dalam iklan sebelum permainan dibebaskan).

2.2.2 Permainan Video (Video Game)

Permainan *video* (*video game*) adalah permainan yang menggunakan interaksi antarmuka pengguna melalui gambar yang dihasilkan oleh piranti *video*. Permainan *video* umumnya menyediakan sistem penghargaan, misalnya *score* yang dihitung berdasarkan tingkat keberhasilan yang dicapai dalam

menyelesaikan tugas-tugas yang ada di dalam permainan *video* tersebut. Sistem elektronik yang digunakan untuk menjalankan permainan *video* disebut *platform*, contohnya adalah komputer pribadi (*personal computer*) ataupun konsol permainan (*game console*) [5].

1. Permainan komputer (*Computer Game*)

Permainan komputer adalah permainan *video* yang di mainkan pada komputer pribadi, dan bukan pada konsol permainan. Permainan komputer telah berevolusi dari sistem grafis sederhana menjadi sangat kompleks dan mutakhir.

Permainan *onlin*e (*online game*) adalah jenis permainan *video* atau permainan komputer dengan menggunakan jaringan internet sebagai media interaksinya.

Permainan *online* terdiri dari dua unsur utama, yaitu *server* dan *client*. *Server* adalah penyedia layanan gaming yang merupakan basis agar *client –client* yang terhubung dapat memainkan permainan dapat melakukan komunikasi dengan baik. Suatu *server* pada prinsipnya hanya melakukan administrasi permainan dan menghubungkan *client-client*. Sedangkan *client* adalah pengguna permainan dan memakai kemampuan *server*. Contoh permainan *online* adalah *Ragnarok Online, Risk Your Life, Xian* [5].

2. Konsol Permainan (*Game Console*)

Konsol permainan adalah sebuah mesin elektronik yang dirancang khusus untuk memainkan permainan *video*. Perangkat keluarannya biasa berupa monitor komputer atau televisi dan alat masukan utamanya berupa sebuah pengendali. Konsol permainan yang pertama kali dibuat adalah *Atari*, kemudian dilanjutkan dengan *Nintendo* yang sukses pada tahun 1985-1989. Konsol permainan modern sekarang ini antara lain adalah *Playstation* buatan perusahaan *Sony*. Untuk *Console* yang berbentuk kecil dan dapat dibawah kemana-mana biasa disebut *Portable Console* [5].

2.2.3 Sudut Pandang Permainan (*Game View Point*)

Sebuah *game* biasanya mempunyai sudut pandang permainan tersendiri disesuaikan berdasarkan *genre game* yang diambil. Berikut beberapa macam sudut pandang permainan yang biasa digunakan [8]:

1. ***Side Scrolling***

   Adalah sudut pandang permainan yang terlihat dari samping dan memungkinkan karakter utama untuk bergerak dari kiri ke kanan serta memungkinkan *background* pada *game* seolah-olah bergeser mengikuti pergerakan karakter utama.

2. ***Top Down***

   Adalah sudut pandang permainan yang memungkinkan karakter utama bermanuver ke empat arah namun cara permainannya sendiri bergeser dari bawah ke atas, dan biasanya *game* yang menggunakan sudut pandang permainan jenis ini adalah *shooter game*.

3. ***Isometric***

   Adalah sudut pandang permainan yang memungkinkan permainan terlihat diantara sisi *Side Scrolling* dan juga *Top Down*, dan biasanya diterapkan pada *game* dengan *genre RTS* (*Real Time Strategy*).

2.2.4 Jenis Permainan (*Genre Game*)

Game dari tahun-tahun semakin banyak saja jenisnya, teknologi yang dipakaipun semakin canggih. Dari ratusan atau bahkan ribuan *game* yang telah bermunculan di dunia, *game-game* tersebut dikelompokkan menjadi beberapa *genre*. Pada dasarnya, *genre-genre game* yang beredar di pasaran dapat dikelompokkan sebagai berikut [8]:

a. ***Advergames***

   Permainan yang dikembangkan untuk tujuan periklanan.

b. ***Action Game***

   Adalah tipe *game* dengan fitur utama berupa banyaknya aksi di mana pemain harus memiliki keterampilan reaksi yang cepat untuk menghindari musuh atau menghindari rintangan. Pengembang *game* tipe ini

perlu memastikan *game* yang dibuat dioptimasi sehingga pemain memiliki pengalaman bermain yang baik, yang tidak terganggu oleh *delay* proses yang lama, contohnya : *Metal Gear Solid*.

c. Adventure Game

Adalah tipe *game* yang umumnya membuat pemain harus berjalan mengelilingi suatu tempat yang terkondisi, seperti sebuah istana, gua yang berkelok, dan planet yang jauh. Pemain melakukan navigasi suatu area mencari pesan-pesan rahasia, memperoleh obyek yang memiliki kemampuan yang bervariasi, bertempur dengan musuh, dan lain-lain. Untuk membuat *game* ini, diperlukan perencanaan yang akurat sehingga memiliki alur cerita yang menarik bagi pemain, contohnya : *Tomb Raider, Prince of Persia, Hitman*, dan lain-lain.

d. Fighting Game

Adalah tipe *game* yang intinya harus menjatuhkan lawan tandingnya, entah itu dengan pukulan, tendangan, combo, maupun dengan jurus special, contohnya : *Tekken, Mortal Kombat, Naruto*, dan lain-lain.

e. FPS (First Person Shooter)

Adalah tipe *game* yang mengunakan sudut pandang orang pertama untuk membidik atau membunuh musuh, sehingga kita hanya melihat tangannya saja dan tidak melihat tubuh karakter yang dimainkan, contohnya : *Call of Duty, Half Life, Crysis*, dan lain-lain.

f. RPG (Role Playing Game)

Adalah tipe *game* yang seringkali berupa multi karakter *game* di mana setiap pemain memiliki karakter dengan kemampuan, kekuatan, dan kelemahan yang spesifik. Para pemain saling berkompetisi, berinteraksi, dan bertempur satu sama lain. Tampilan grafis yang khas untuk setiap karakter pemain ditambah dengan *storyline* yang mendebarkan akan sangat menarik dan memberikan pengalaman yang berbeda di dalam bermain, contohnya : *Final Fantasy, Warcraft*, dan lain-lain.

g. Platform Game

Adalah tipe *game* yang mengharuskan pemain mengarahkan suatu obyek dengan melalui berbagai tahap atau tingkatan area untuk menyerang musuh dan menghindar terhadap serangan. Tipe *game* ini sedikit serupa dengan *action game*, tetapi aksinya tidak secepat *action game*. Teknik *collision detection* sangat sering dimanfaatkan pada tipe *game* ini, contohnya : *Sonic The Hedgehog, Mario Bros, Metal Slug*.

h. Puzzle Game

Adalah tipe *game* yang umumnya membuat pemain menggunakan kemampuan berpikirnya sebagai pengganti keterampilan reaksi yang cepat karena terdapat rahasia yang perlu dipecahkan. *Game* ini lebih bersifat statis dibanding *action game*. Pembuatan *game* tipe ini seringkali ditunjang dengan AI (*Artificial Intelligence*), contohnya : *7 Wonders of The Ancient World.*

i. Racing Game

Adalah tipe *game* yang tujuannya adalah mencapai garis finish dari suatu *race*, dalam *game* ini biasanya pemain dapat memilih dan membeli kendaraan, mendandani, melakukan *upgrade* mesin , contohnya : *Need For Speed.*

j. Simulation Game

Adalah tipe *game* yang memberikan pengalaman atau interaksi sedekat mungkin dengan keadaan yang aslinya, meskipun terkadang keadaan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detail dan pengalaman realistik menggunakan keadaan tersebut, contohnya : *Gran Turismo.*

k. Sandbox Game

Adalah tipe *game* yang umumnya ditujukan untuk menjelajahi suatu kota atau tempat dan bebas berinteraksi dengan objek disekitarnya. Mungkin *game* ini bisa di bilang mirip *RPG* tapi *Sandbox* sangat berbeda dengan *RPG game*, perbedaannya adalah tidak adanya peningkatan *level* karakternya, contohnya : *GTA dan Bully*.

l. RTS (Real Time Strategy)

Adalah tipe *game* yang mengharuskan pemainnya menggunakan taktik dan strategi untuk jeli dalam melihat setiap peluang, kelemahan musuh dan bijaksana dalam menggunakan sumber daya yang ada, contohnya : *Command and Conquer.*

m. Sport Game

Adalah tipe *game* yang berupa kompetisi antara dua pemain atau lebih, di mana pemain dapat berupa individual atau tim. Contoh *game* tipe ini antara lain sepakbola, bola basket, tenis, dan *bilyard*. Tergantung seberapa cepat permainan yang terjadi, aplikasi *game* perlu dioptimalkan, contohnya : *Pro Evolution Soccer.*

2.3 Kecerdasan Buatan / AI (*Artificial Intelligence*)

Kecerdasan buatan atau dalam bahasa Inggrisnya *Artificial Intelligence* sering disingkat AI merupakan salah satu bagian ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia.

AI ialah ilmu dan rekayasa, yang membuat mesin mempunyai intelegensi tertentu khususnya program komputer yang ‘cerdas’. Intelegensi merupakan bagian kemampuan komputasi untuk mencapai tujuan di dalam dunia. Ada beberapa macam jenis dan derajat intelegensi untuk manusia, binatang dan beberapa mesin. Pengertian kecerdasan buatan juga dapat dipandang dari berbagai sudut pandang [9], antara lain:

1. Sudut pandang kecerdasan.

   Kecerdasan buatan akan membuat mesin menjadi ‘cerdas’ (mampu berbuat seperti apa yang dilakukan manusia).

2. Sudut pandang penelitian

   Kecerdasan buatan adalah suatu studi bagaimana membuat agar komputer dapat melakukan sesuatu sebaik yang dikerjakan manusia.

2.3.1 Karakteristik AI

Karakteristik ini didasarkan pada pandangan bahwa AI ada 4 kategori yaitu [10]:

a. Sistem yang dapat berpikir seperti manusia.
b. Sistem yang dapat beraksi seperti manusia.
c. Sistem yang dapat berpikir secara rasional.
d. Sistem yang dapat beraksi secara rasional.

2.3.2 Perkembangan AI dan Aplikasinya

Ai merujuk pada satu bidang komputer yang berupaya melakukan tugas sama seperti manusia. Dalam keadaan ini, komputer tersebut boleh dikatakan cerdas. Banyak bidang yang melibatkan AI antara lain. Penyelesaian masalah, logika kabur (*fuzzy*), pengenalan bahasa alamiah, robotik, dan sistem pakar. AI boleh di identifikasikan sebagai kajian yang menyeluruh bagaimana sebuah komputer berupaya melakukan tugas dengan terapan yang lebih baik jika di bandingkan manusia. Kajian tersebut meliputi kebolehan komouter belajar, memahami, membuat keputusan, dan meningkatkan prestasi atas hasil kesalahan yang telah dilakukan sebelumnya [10].

2.3.3 Metode Pencarian (*Searching*)

Teknik *searching* digunakan untuk pencarian *rute* optimum untuk memandu seseorang diperjalanan. Berikut cara pemecahan masalah dengan teknik *searching* atau pencarian [11]:

1. Mendefinisikan ruang masalah untuk suatu masalah ynag dihasapi, ruang masalah ini dapat digambarkan sebagai himpunan keadaan (*state*) atau bisa juga sebagai himpunan *rute* dari keadaan awal (*initial state*) menuju keadaan tujuan (*goal state*).
2. Mendefinisikan aturan produksi yang digunakan untuk mengubag suatu *state* ke *state* lainnya.
3. Memilih metode pncarian yang tepat sehingga dapat menemukan solusi terbaik dengan usaha yang minimal.

Pada dasarnya teknik pencarian (*searching*) dapat dibagi menjadi 2 kelompok besar, yaitu pencarian *blind* atau *un-informed search* (pencarian buta atau tidak berbekal informasi) dan *heuristic* atau *informed search* (pencarian dengan panduan atau berbekal informasi). Untuk mengukur performa metode pencarian, terdapat empat kriteria yang dapat dilakukan [11], yaitu :

a. *Completeness* : apakah metode tersebut menjamin adanya solusi jika solusinya ada.
b. *Time Complexity* : berapa lama waktu yang diberikan untuk menemukan solusi tersebut.
c. *Space complexity* : berapa banyak memori yang dibutuhkan untuk menemukan solusi tersebut.
d. *Optimaly* : apakah metode tersebut menjamin menemukan solusi terbaik jika terdapat beberapa solusi yang berbeda.

2.3.4 Pencarian Blind/Un-informed Search

Pada teknik pencarian ini tidak terdapat informasi awal yang digunakan dalam proses pencarian. 6 metode yang tergolong *blind search* [11], yaitu :

1. *Breadth-First Search* (BFS)

   Pencarian dilakukan pada semua simpul dalam setiap *level* secara berurutan dari kiri ke kanan. Jika pada satu *level* belum ditemukan solusi, maka pencarian dilakukan pada *level* selanjutnya. Demikian seterusnya sampai ditemukan solusi.

2. *Depth-First Search* (DFS)

   pencarian dilakukan pada suatu simpul dalam setiap *level* dari yang paling kiri. Jika pada *level* yang terdalam solusi belum ditemukan, maka pencarian pada simpul sebelah kanan dan simpul yang kiri dapat dihapus dari memori. Jika pada *level* yang paling dalam tidak ditemukan solusi, maka pencarian dilanjutkan pada *level* sebelumnya. Demikian selanjutnya sampai ditemukan solusi.

3. *Depth-Limited Search* (DLS)

   Metode ini berusaha mengatasi kelemahan DFS (tidak *complete*) dengan membatasi kedalaman maksimum dari suatu jalur solusi. DLS bisa menjadi tidak *complete* jika batasan kedalamannya lebih kecil dibandingkan dengan *level* solusinya.

4. *Uniform Cost Search* (UCS)

   Kosepnya hampir sama dengan BFS, bedanya adalah bahwa BFS menggunakan urutan *level* dari yang paling rendah sampai yang paling tinggi, sedangkan UCS menggunakan urutan biaya dari yang paling kecil sampai yang terbesar. UCS berusaha menemukan solusi dengan total baiya terendah yang dihitung berdasarkan biaya dari simpul asal menuju simpul tujuan.

5. *Iteractive-Deepening Search* (IDS)

   IDS merupakan metode yang menggabungkan kelebihan BFS (*complete* dan *optimal*) dengan kelebihan DFS (*space complexity* rendah atau membutuhkan sedikit memori). Tapi konsekuensinya adalah *time complexity*-nya menjadi tinggi.

6. *Bi-Directional Search* (BDS)

   Pada setiap iterasi pencarian dilakukan dari dua arah : pencarian maju (dari *start* ke *goal*) dan pencarian mundur (dari *goal* ke *start*). Ketika dua arah pencarian telah membangkitkan simpul yang sama, maka solusi telah ditemukan, yaitu dengan cara menggabungkan kedua jalur yang bertemu.

2.3.5 Pencarian *Heuristic* (Terbimbing)

Pencarian buta biasanya tidak efisien karena waktu akses dan memori yang di butuhkan cukup besar. Untuk mengatasi hal ini maka perlu ditambahkan suatu informasi pada domain yang bersangkutan sehingga proses pencarian yang baru akan dihasilkan. Pencarian seperti ini disebut sebagai *informed search* atau pencarian *heuristic* (terbimbing) yaitu pencarian berdasarkan panduan.

Teknik pencarian *heuristic* (*heuristic searching*) merupakan suatu strategi untuk melakukan proses pencarian secara selektif dan dapat memandu proses pencarian yang memiliki kemungkinan sukses paling besar, namun dengan kemungkinan mengorbankan kelengkapan (*completeness*).

Untuk menerapkan pencarian *heuristic* diperlukan suatu fungsi *heuristic*. Fungsi *heuristic* adalah aturan-aturan yang digunakan untuk mendapatkan solusi yang diinginkan. Pencarian *Heuristic* terdiri dari : *Generate And Tes* (Pembangkitan dan Pengujian), *Hill Climbing* (Pendakian Bukit) : *Simple Hill Climbing, Steepest-Ascent Hill Climbing, Best First Search,* Algoritma A* (A *Star*), *Simulated Anneliang* [11].

2.3.6 AI yang digunakan

Algoritma *Greedy* adalah algoritma yang untuk membentuk solusi langkah perlangkah. Pada setiap langkah tersebut akan dipilih keputusan yang paling optimal. Keputusan tersebut tidak perlu memperhatikan keputusan selanjutnya yang akan diambil, dan keputusan tersebut tidak dapat diubah lagi pada langkah selanjutnya. Definisi dari Algoritma *Greedy* secara umum adalah suatu algoritma pencarian solusi yang memecahkan suatu masalah langkah per langkah, dimana pada setiap langkah [11]:

1. Mengambil pilihan terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi kedepan.
2. Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

2.3.6.1 Prinsip Utama Algoritma *Greedy*

Prinsip utama algoritma *greedy* adalah "*take what you can get now*". Maksud dari prinsip tersebut adalah pada setiap langkah dalam algoritma *greedy*, diambil keputusan yang paling optimal untuk langkah tersebut tanpa memperhatikan konsekuensi pada langkah selanjutnya dinamakan solusi tersebut dengan optimum lokal. Kemudian saat pengambilan nilai optimum lokal pada setiap langkah, diharapkan tercapai optimum global, yaitu tercapainya solusi optimum yang melibatkan keseluruhan langkah dari awal sampai akhir [11].

2.3.6.2 Elemen Utama Algoritma *Greedy*

Elemen-elemen yang digunakan dalam penerapan algoritma *greedy* antara lain :

1. Himpunan Kandidat

   Himpunan yang berisi kemungkinan - kemungkinan yang bisa menjadi solusi dari permasalahan.
2. Himpunan Solusi

   Himpunan yang berisi kandidat yang telah terpilih sebagai solusi.
3. Fungsi Seleksi

   Fungsi untuk melakukan seleksi terhadap kandidat agar menghasilkan solusi yang diharapkan
4. Fungsi Kelayakan

   Fungsi untuk memastikan bahwa solusi yang dipilih memenuhi syarat.
5. Fungsi Objektif

   Memilih solusi paling optimal dari himpuna solusi.

2.3.1 Metode *Collision Detection*

Hampir setiap permainan *video* yang ada sudah menerapkan *collision detection* (deteksi tabrakan), baik itu dalam hal tabrakan antara *sprite* dengan *sprite* maupun antara *sprite* dengan peluru dan lain-lain. Dalam aplikasi permainan ada beberapa reaksi yang ditunjukkan dari *collision detection*, yaitu

reaksi dari *spirte* yang berupa berkurangnya nyawa, berkurangnya darah, meledak, dan lain-lain.

Proses *collision* dapat dibagi menjadi dua kategori dasar, yaitu *collision detection* dan *collision response*, dengan jarak respon yang telah diaplikasikan secara spesifik. Selain itu pada sumber lain disebutkan bahwa *collision detection* terbagi menjadi dua jenis [12], yaitu :

1. *Collision Detection Object To Object*

   Tubrukan pada *object to object* yang terjadi adalah adanya tubrukan antara satu objek dengan objek lainnya yang berada di dalam satu *frame*, baik itu objek tersebut bertubrukan secara langsung ataupun bertubrukan antara kotak maupun lingkaran yang mengandung objek tersebut.
2. *Collision Detection Object To World*.

   Sedangkan pada *object to world* yang terjadi adalah adanya tubrukan antara objek dengan *frame* itu sendiri, contohnya adalah tubrukan antara *sprite* dengan dinding pada aplikasi permainan itu sendiri sehingga apabila terjadi, maka *sprite* tersebut tidak dapat melewati dinding tersebut.

2.4 *Tools* yang digunakan

Pada sub bab ini dijelaskan *tools* apa saja yang digunakan dalam pembangunan *game* Petualangan Kukuyaan di Jawa Barat.

2.4.1 OOP (*Object Oriented Programming*)

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan pernagkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya, yang merupakan suatu acara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pasa penerapan prinsip-prinsip pengolahan kompleks meliputi rangkaian aktivitas

analisis berorientasi objek, perencanaan berorientasi objek, pemograman berorientasi objek, dan pengujian berorientasi objek [13].

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi dan lainnya) yang akan mendukung sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nyata. Setiap komponen dalam sistem tersebut dapat mewarisi atribut dan sifat komponen lainnya, dan dapat berinteraksi satu sama lain [13].

Enam keuntungan menggunakan OOP : Alami (Natural), dapat diandalkan (*Reliable*), dapat digunakan kembali (*Reusable*), mudah untuk dalam perawatan (*Maintainable*), dapat diperluas (*Extendable*), efisiensi waktu.

Berikut ini beberapa bahasa pemrograman yang sudah menggunakan konsep OOP [13], adalah :

1. *C++*.
2. *Visual C++*.
3. *Visual Basic*.
4. *Java*.

Berikut adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek [13] :

a. Kelas (*class*)

Kelas adalah kumpulan objek – objek dengan karakteristik yang sama. Kelas merupakan definisi statis himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, di mana atribut dan kelas semula dapat diwariskan kekelas yang baru. Secara teknis, kelas adalah sebuah struktur tertentu dalam pembuatan perangkat lunak.

b. Objek (*Object*)

Objek adalah abstraksi dan sesuatau yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

c. Metode (*Method*)

Operasi atau metode pada suatu kelas hampir sama dengan fungsi prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek. Metode atau operasi dapat berasal dari *event*, aktivitas atau aksi keadaan, fungsi, atau kelakuan dunia nyata.

d. Atribut (*attribute*)

Atribut dari sebuah kelas adalah variabel global yang memiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama dsb.

e. Abstraksi (*abstraction*)

Prinsip untuk mempresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

f. Pembungkusan (*Encapsulation*)

Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

g. Pewarisan (*Inheritance*)

Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

h. Antarmuka (*interface*)

Antarmuka atau *interface* sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuat *interface* dapat diimplementasikan oleh kelas lain.

i. *Reusability*

Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

j. Generalisasi dan Spesialisasi

Menunjukan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus.

k. Komunikasi Antarobjek

Komunikasi antarobjek dilakuka lewat pesan (*message*) yang dikirim dan satu objek ke objek lainnya.

l. Polimorfisme (*polymorphism*)

Kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

m. *Package*

Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

2.4.2 UML (Unified Modeling Language)

Unified Modelling Language (UML) adalah salah satu standar bahasa yang banyak digunakan didunia industri untuk mendefinisikan *requirement,* membuat analisis dan desain, serta menggambarkan arsitektur dalam pemograman berorientasi objek [13].

UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk [13] :

a. Merancang perangkat lunak.
b. Sarana komunikasi antara perangkat lunak dengan proses bisnis.

c. Menjabarkan *system* secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
d. Mendokumentasi sistem yang ada, proses-proses dan organisasinya.

Blok pembangun utama UML adalah diagram. Beberapa diagram ada yang rinci, dan ada yang bersifat umum. Dengan demikian UML mrupakan alat komunikasi yang konsisten dalam mensuport para pengembang sistem saat ini [14].

A. Evolusi UML

Pada bulan oktober 1994, Jim Rumbaugh, penemu *Object Modelling Technique (OMT)* dan Grady Booch, penemu *Booch Methode* (Metode Booch) bersama-sama menyamakan notasi mereka. Di tahun yang sama Ivar Jacobson (penemu *Objectory Method*) ikut bergabung. Ketiga orang itu bersama-sama membangun notasi standar OOP untuk *software Rational* IBM. Jim Rumbaugh banyak member masukan dalam pembuatan notasi dan analisa UML. Grady Booch merancang secara detail kapsitas UML, sedang Ivar Jacobson berusaha membuat UML cocok dengan model bisinis dan mencoba mengembangkan use case diagram lebih lanjut .

B. Diagram-diagram UML

Jenis-jenis Diagram UML yaitu :

a. Diagram kelas. Bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antar muka, kolaborasi, serta relasi-relasi. Pemodelan ini umum dijumpai pada pemodelan sistem berorientasi objek.
b. Diagram Paket. Bersifat statis. Diagram ini memperlihatkan kumpulan kelas-kelas, merupakan bagian dari diagram komponen.
c. Diagram *Use-case*. Bersifat statis. Diagram ini memperlihatkan himpunan *use case* dan aktor-aktor. Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu *system* yang dibutuhkan serta diharapkan pengguna.

d. Diagram interaksi dan *Sequence*. Bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu.
e. Diagram komunikasi. Bersifat dinamis. Diagram yang menekankan organisasi struktural dari objek-objek yang menerima serta mengirim pesan.
f. Diagram *Statechart*. Bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (*state*), transisi, kejadian sera aktifitas.
g. Diagram aktivitas. Bersifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainya dalam suatu sistem.
h. Diagram komponen. Bersifat statis. Diagram komponen ini memperlihatkan organisani serata kebergantungan *system*/perangkat lunak pada komponen-komponen yang telah ada sebelumnya.
i. Diagram *Deployment*. Bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi di jalankan .

2.4.3 Pengujian Kotak Putih (*White-Box)*

White Box Testing merupakan cara pengujian dengan melihat ke dalam modul untuk meneliti kode-kode program yang ada, dan menganalisis apakah ada kesalahan atau tidak. Jika ada modul yang menghasilkan output yang tidak sesuai dengan proses bisnis yang dilakukan, maka baris-baris program, variabel, dan parameter yang terlibat pada unit tersebut akan dicek satu persatu dan diperbaiki, kemudian di-compile ulang [15].

Dengan menggunakan *white box* akan didapatkan kasus uji yang :

1. Menguji semua keputusan logikal
2. Menguji seluruh Loop yang sesuai dengan batasannya
3. Menguji seluruh struktur data internal yang menjamin validitas

Kelebihan *White Box Testing* adalah :

1. Kesalahan Logika

   Digunakan pada sintaks ‘if’ dan pengulangan. Dimana *White Box Testing* akan mendeteksi kondisi-kondisi yang tidak sesuai dan mendeteksi kapan proses pengulangan akan berhenti.
2. Ketidaksesuaian asumsi

   Menampilkan asumsi yang tidak sesuai dengan kenyataan, untuk di analisa dan diperbaiki.
3. Kesalahan ketik

   Mendeteksi bahasa pemrograman yang bersifat *case sensitive.*

2.4.4 Pengujian Jalur Dasar

Pengujian jalur dasar (*basis path testing)* adalah teknik pengujian kotak putih dimana metode ini memungkinkan perancang *test case* untuk menurunkan ukuran kompleksitas logis dari suatu rancangan prosedural dan menggunakan ukuran ini sebagai pedoman untuk menentukan rangkaian dasar jalur eksekusi. *Test case* siturunkan untk menguji rangkaian dasar yang dijamin untuk mengeksekusi setiap pernyataan dalam program, setidaknya satu kali selama pengujian [15].

2.4.4.1 Notasi Grafik Alir (*Flow Graph*)

Notasi Grafik Alir (*flow graph*) menggambarkan arus kontrol logis dengan menggunakan notasi [15].

2.4.4.2 Jalur Program Independen

Jalur independen adalah setiap jalur yang melalui progrma yang memperkenalkan setidaknya satu kumpulan pernyataan-pernyataan pemrosesan atau kondisi baru, bila dinyatakan dalam grafik alir, jalur independen harus bergerak sepanjang setidaknya satu *edge* yang belum dilintasi sebelum jalur tersebut didefinisikan [15].

2.4.4.3 Matrik-Matrik Grafik

Matriks grafik adalah matriks persegi yang ukurannya (jumlah, baris, dan kolom) sama dengan jumlah *node* pada grafik alir. Setiap baris dan kolom terkait dengan *node* yang diidentifikasi, dan masukan matriks terkait dengan koneksi-koneksi (*edge*) yang berada diantara *node-node* [15].

2.4.5 Pengujian *Black-Box*

Metode ujicoba *blackbox* memfokuskan pada keperluan fungsional dari *software*. Karna itu ujicoba blackbox memungkinkan pengembang *software* untuk membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program. Ujicoba *black box* bukan merupakan alternatif dari ujicoba *whitebox*, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode *whitebox* [15].

Ujicoba *blackbo*x berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan interface
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan performa
5. Kkesalahan inisialisasi dan terminasi

2.4.6 StarUML

StarUML adalah *software* permodelan yang mendukung UML *(Unified Modeling Language).* Berdasarkan pada UML version 1.4 dan dilengkapi 11 macam diagram yang berbeda, mendukung notasi UML 2.0 dan juga mendukung pendekatan MDA *(Model Driven Architecture)* dengan dukungan konsep UML. StarUML dapat memaksimalkan pruduktivitas dan kualitas dari suatu *software project.* Berikut konsep Dasar pada StarUML [16] :

a. Model, View and Diagram

StarUML membuat perbedaan konseptual yang lebih jelas antara *models, views and diagrams.* Model adalah elemen yang memuat informasi untuk model *software*. *View* adalah suatu ekspresi visual dari informasi di dalam model dan Diagram adalah suatu koleksi dari elemen yang memberikan pemikiran user di dalam mendesain secara spesifik.

b. Project and Unit

Project adalah unit manajemen dasar di dalam StarUML. Suatu *project* dapat mengatur satu atau lebih model *software*. *Project* merupakan top-*level package* yang selalu ada di dalam beberapa model *software*. Secara umum, satu *project* disimpan dalam satu *file*. Struktur Sebuah projek terdiri dari dan mengatur *sub-elements* [16], dapat dilihat pada Tabel 2.1 berikut ini :

Tabel 2.1 Struktur Objek

Project Sub-Element	Deskripsi
Model	Elemen yang mengatur suatu model software.
Subsystem	Elemen yang mengatur model-model yang tercangkup dalam satu sub sistem.
Package	Elemen yang paling umum untuk mengatur elemen.

File project disimpan ke dalam format XML dengan extension ".UML". Semua model, *views* dan diagrams yang dibuat dengan StarUML disimpan dalam satu *file project*. *File project* berisikan informasi [15], sebagai berikut :

1. UML *profile* yang digunakan dalam projek.

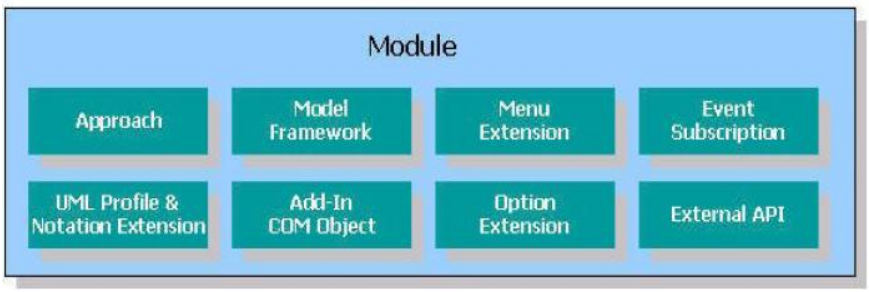
2. Unit *file* yang direferensi oleh projek.
3. Informasi untuk semua model yang ada di dalam *project*.
4. Informasi untuk semua diagrams dan *views* yang ada di dalam *project*.

Ada beberapa kasus dimana satu *project* perlu disimpan di dalam beberapa *file file* kecil sehingga para pengembang dapat bekerja di dalam satu *project* secara bersamaan. Di dalam kasus ini suatu *project* dapat mengatur bermacam – macam unit.

Suatu unit mempunyai struktur hirarki dan berisikan beberapa sub-unit. Unit disimpan sebagai ".UML " *file* dan beberapa mengacu pada *file project* (.UML) atau unit *file* lainnya (.UNT). Komposisi Unit Hanya *package, subsystem* dan elemen model yang dapat membentuk satu unit. Semua elemen di bawah jenis elemen *package* ini disimpan di dalam masing – masing *file* unit (.UNT) [16].

c. Module

Modul adalah suatu *package* yang menyediakan fungsi – fungsi baru dan feature sebagai perluasan dari StarUML. Modul dapat dibuat sebagai kombinasi dari beberapa elemen – elemen extension dan juga membuat beberapa jenis elemen – elemen di dalam suatu modul, dapat dilihat pada Gambar 2.1



Gambar 2.1 *Module*

2.4.7 Adobe flash

Adobe flash (dahulu bernama *Macromedia flash*) adalah salah satu perangkat lunak komputer yang merupakan produk unggulan *Adobe Systems*. *Adobe flash* digunakan untuk membuat gambar vektor maupun animasi gambar tersebut. Berkas yang dihasilkan dari perangkat lunak ini mempunyai extension .swf dan dapat diputar dipenjelajah web yang telah dipasang *Adobe flash* Player. *Flash* menggunakan bahasa pemograman yang bernama *Action Script* yang muncul pertama kalinya pada *Flash* 5 [17].

Adobe flash merupakan sebuah program yang didesain khusus oleh *Adobe* dan program aplikasi standar *authoring tool profesional* yang digunakan untuk membuat animasi dan bitmap yang sangat menarik untuk keperluan membangun situs web yang interaktif dan dinamis. *Flash* didesain dengan kemampuan untuk membuat animasi 2 dimensi yang andal dan ringan sehingga *Flash* banyak digunakan untuk membangun dan memberikan efek animasi pada *website, CD* interaktif, *game* dan yang lainnya. Dalam *flash* terdapat teknik-teknik membuat animasi, fasilitas *action script, filter, custom easing* dan dapat memasukan vidio lengkap dengan fasilitas *playback* FLV, dan dapat disisipkan *code* pemograman dan *database* dengan pendekatan *XML* [17].

2.4.8 Adobe photoshop

Adobe photoshop adalah *software* yang dibuat oleh perusahaan *Adobe System*, yang di khususkan untuk pengeditan foto atau gambar dan pembuatan effect. Perangkat lunak ini banyak digunakan oleh Fotografer Digital dan perusahaan iklan sehingga dianggap sebagai pemimpin pasar (*market leader*) untuk perangkat lunak pengolah gambar. Meskipun pada awalnya *Photoshop* dirancang untuk menyunting gambar untuk cetakan berbasis-kertas, *Photoshop* yang ada saat ini juga dapat digunakan untuk memproduksi gambar untuk *World Wide Web*. Beberapa versi terakhir juga menyertakan aplikasi tambahan, *Adobe ImageReady*, untuk keperluan tersebut.

Photoshop juga memiliki hubungan erat dengan beberapa perangkat lunak penyunting media, animasi, dan *authoring* buatan-*Adobe* lainnya. *File* format asli

Photoshop, PSD, dapat diekspor ke dan dari *Adobe ImageReady*. *Adobe Illustrator, Adobe Premiere Pro, After Effects dan Adobe Encore DVD* untuk membuat DVD profesional, menyediakan penyuntingan gambar non-linear dan layanan special *effect* seperti *background*, tekstur, dan lain-lain untuk keperluan televisi, film, dan situs web [17].

2.4.9 ActionScript

Adalah bahasa yang dikenali dan digunakan dalam *Flash* untuk berbagai hal dalam movie [17].

1. *ActionScript* 1.0 : *Flash* 5 dan *Flash MX*(6)
2. *ActionScript* 2.0 : *Flash MX* 2004(7) dan *Flash* (8)
3. *ActionScript* 3.0 : *Flash CS3* (9), *Flash CS4* (10), *Flash CS6* (11)