BAB 2
LANDASAN TEORI

2.1 Wayang

Wayang adalah salah satu puncak seni budaya bangsa Indonesia yang paling menonjol di antara banyak karya budaya lainnya. Budaya wayang meliputi seni peran, seni suara, seni musik, seni tutur, seni sastra, seni lukis, seni pahat, dan juga seni perlambang. Budaya wayang, yang terus berkembang dari zaman ke zaman, juga merupakan media penerangan, dakwah, pendidikan, hiburan, pemahaman filsafat, serta hiburan.

2.1.1 Jenis-jenis Wayang

Ada bermacam-macam wayang yang dapat di jumpai di Jawa. Wayang tersebut terdiri dari bermacam-macam bahan pula/ beberapa jenis wayang tersebut dapat disebutkan sebagai berikut[4] :

1. Wayang Purwa

   Pada umumnya lakon yang dibawakan dalam Wayang Purwa diambil dari Ramayana dan Mahabarata. Awalnya bentuk Wayang Purwa didasarkan pada bentuk relief candi, lambat laun bentuk itu mengalami perubahan sedemikian rupa sehingga sesuai dengan pribadi masyarakat Indonesia (Jawa).

2. Wayang Madya

   Wayang Madya merupakan ciptaan Mangkunegara IV Surakarta. Cerita wayang yang dipergelarkan melanjutkan cerita Wayang Purwa. Pada Wayang Madya umumnya tokoh-tokoh raja tidak memakai praba (sinar atau nimbus), suatu perhiasan yang diapakai pada setiap raja sebagai lambang kedudukannya.

3. Wayang Klithik

   Boneka wayang klithik ini wujudnya pipih. Walaupun tidak setipis dan dibuat dari kayu. Lengan atau tangannya dibuat dari kulit sapi atau kerbau. Wayang klithik ini menceritakan tanah Jawa, khususnya kerajaan Majapahit dan Pajajaran.

4. Wayang Beber

   Wayang Beber merupakan gambar wayang yang dilukis pada kain putih. Wayang Beber biasanya terdiri dari 4 gulung yang berisikan 16 adegan. Setiap kali diceritakan lalu gambar wayang itu direntangkan agar diketahui oleh penonton bagaimana benuk lukisan certia tersebut.

5. Wayang Gedog

   Wayang Godeng diciptakan oleh Sunan Giri, untuk digunakan dalam cerita Panji, yang merupakan cerita raja-raja Jenggala, yaitu mulai dari Prabu Sri Ghataya (Subrata) sampai dengan Panji Kudalaleyan. Bentuk wayang Gedog ini mirip dengan bentuk wayang purwa, tetapi tidak menggunakan gelung "sapit urang" pada tokoh rajanya.

6. Wayang Golek

   Wayang Golek kebanyakan berpakaian jubah atau baju panjang tanpa digerakan secara bebas dan terbuat dari kayu yang bentuknya bulat seperti lazimnya boneka. Cerita wayang Golek bersumber pada serat Menak, yang berisikan cerita Arab. Tetapi ada beberapa daerah yang menggunakan cerita Ramayana dan Mahabrata.

Pada *game* yang akan dibangun karakter wayang akan menggunakan karakter pada wayang purwa. Dimana tokoh-tokoh wayang purwa yang akan dipakai [5] adalah :

1. Arjuna

   Arjuna seorang satria yang gemar berkelana, bertapa dan berguru menuntut ilmu. Ia dijadikan jago kadewatan membinasakan Prabu Niwatakawaca, raja raksasa dari negara Manimantaka. Atas jasanya itu, Arjuna dinobatkan sebagai raja di Kahyangan Kaindran bergelar Prabu Karitin. Arjuna memiliki 15 istri yang salah satunya adalah Dewi Srikandhi. Senjata pamungkas Arjuna adalah Panah Pasopati, Arjuna memiliki sifat perwatakan ; Cerdik, pandai, pendiam, teliti, sopan-santun, berani dan suka melindungi yang lemah.

2. Srikandi

   Dewi Srikandi sangat gemar dalam olah keprajuritan dan mahir dalam mempergunakan senjata panah. Kepandaiannya tersebut didapatnya ketika ia berguru pada Arjuna, yang kemudian menjadi suaminya. Dewi Srikandi menjadi suri tauladan prajurit wanita. Ia bertindak sebagai penanggung jawab keselamatan dan keamanan kesatrian Madukara dengan segala isinya.

3. Adipati Karna

   Adipati Karna yang nama lengkapnya Basukarna, adalah putra Dewi Kunti. Karna lahir diluar perkawinan yang syah akibat kesalahan Dewi Kunti membaca mantera Aji Pameling ajaran Resi Druwasa. Bayi Karna kemudian dihanyutkan/dilarung ke dalam sungai Gangga dan ditemukan oleh Rada, sais/ kusir kereta kerajaan Astina. Dari garis keturunan Ibu (Dewi Kunti), Karna mempunyai tiga saudara lelaki lain ayah. Mereka adalah ; Puntedewa, Bima dan Arjuna. Karna bersahabat baik dengan Prabu Duryudana, raja Negara Astina. Ia kemudian diangkat menjadi raja negara Awangga, sehingga lebih dikenal dengan sebutan Adipati Karna. Karna memiliki sifat perwatakan; Pemberani, tahu harga diri, setia, prajurit ulung, teguh dalam pendirian. Akhir riwayatnya diceritakan, gugur dalam perang Bharatayuda oleh panah Pasopati milik Arjuna.

4. Jayadrata

   Arya Jayadrata nama sesungguhnya adalah Arya Tirtanata atau Bambang Sagara. Karena ingin memperdalam pengetahuannya dalam bidang tata pemerintahan dan tata kenegaraan, Prabu Sinduraja pergi ke negara Astina untuk berguru pada Prabu Pandu Dewanata. Untuk menjaga kehormatan dan harga diri, ia menukar namanya dengan nama patihnya, Jayadrata. Jayadrata mempunyai sifat perwatakan; berani, penuh kesungguhan dan setia. Ia mahir mempergunakan panah dan sangat ahli bermain gada. Jayadrata tewas oleh Arjuna di medan perang Bharatayuda sebagai senapati perang Kurawa. Kepalanya terpangkas lepas dari badannya oleh panah sakti Pasopati.

5. Bisma

   Resi Bisma yang waktu mudanya bernama Dewabrata. Dengan kesaktiannya, Bisma memenangkan sayembara di negara Kasi, dan memboyong Dewi Amba, Dewi Ambika dan Dewi Ambiki/Ambalika. Dewi Ambika dikawinkan dengan Citragada, sedangkan Dewi Ambiki dengan Wicitrawirya. Sayang, akhirnya Bisma berutang pati dengan Dewi Amba, karena tanpa sengaja telah membunuh wanita tersebut. Sumpah Dewi Amba yang akan membalas dendam pada Bisma, dibuktikan dalam perang Baratayuda. rwah Dewi Amba manunggal dalam tubuh Dewi Srikandi, yang menjadi perantara kematian Resi Bisma.

2.1.2 Perang Baratayuda

Perang Baratayuda merupakan kisah perang yang terhebat dalam pewayangan. Perang ini adalah perang antar saudara yaitu Pandhawa(melambangkan kekuatan baik) dan sepupu mereka Kurawa(melambangkan kekuatan jahat) akibat perebutan takhta kerajaan yang seharusnya jatuh ke tangan Puntadewa, yang malah diberikan pamannya(Destrarestra) ke anaknya yaitu Duryudana, anak sulung dari para Kurawa. Padahal Destraresta berjanji akan mengembalikan takhta ke tangan Puntadewa saat mereka sudah cukup dewasa.

Di saat mereka sudah mulai berkuasa, datanglah Korawa dan Sengkuni yang mencurangi mereka dalam permainan judi sehingga para Pandawa pun harus pergi ke hutan sebagai akibat dari kekalahan mereka. Pandawa yang terusir tidak tinggal diam perlahan-lahan mereka membuat sebuah kerajaan yang bisa menandingi astina, mereka beri nama “Indaprasta”. Karena merasa tertandingi degan adanya kerajaan iIndaprasta pihak korawa mulai meyerang. Balatentara Korawa menyerang laksana gelombang lautan yang menggulung-gulung, sedang pasukan Pandawa menyerang dengan dahsyat seperti senjata yang menusuk langsung ke pusat kematian. Bisma dapat menewaskan Resi Seta. Baratayuda babak pertama diakhiri dengan sukacita pihak Korawa karena kematian pimpinan perang Pandawa.

Setelah Resi Seta gugur, Pandawa kemudian mengangkat Drestadyumna (Trustajumena) sebagai pimpinan perangnya dalam perang Bharatayuddha. Sedangkan Bisma tetap menjadi pimpinan perang Korawa. Dalam babak ini kedua kubu berperang dengan siasat yang sama yaitu Garudanglayang (Garuda terbang). Dalam pertempuran ini dua anggota Korawa, Wikataboma dan kembarannya, Bomawikata, terbunuh setelah kepala keduanya diadu oleh Bima. Sementara itu beberapa raja sekutu Korawa juga terbunuh dalam babak ini. Diantaranya Prabu Sumarma, raja Trigartapura tewas oleh Bima, Prabu Dirgantara terbunuh oleh Arya Satyaki, Prabu Dirgandana tewas di tangan Arya Sangasanga (anak Setyaki), Prabu Dirgasara dan Surasudirga tewas di tangan Gatotkaca, dan Prabu Malawapati, raja Malawa tewas terkena panah Hrudadali milik Arjuna.

Setelah melihat komandan pasukannya berguguran kemudian Bisma maju ke medan pertempuran, mendesak maju menggempur lawan. Atas petunjuk Kresna, Pandawa kemudian mengirim Dewi Srikandi untuk maju menghadapi Bisma. Dengan tampilnya prajurit wanita tersebut di medan pertempuran menghadapi Bisma. Bisma merasa bahwa tiba waktunya maut menjemputnya, sesuai dengan kutukan Dewi Amba yang tewas di tangan Bisma. Bisma gugur dengan perantaraan panah Pasopati milik Arjuna yang dilepaskan oleh istrinya, Srikandi.

Pada saat pandawa sibuk bertarung dengan laskar Samsaptaka. Abimanyu turun untuk melawan pihak korawayang dipimpin oleh Jayadarta, tetapi dalam pertarungan tersebut Abimanyu gugur karena pihak korawa yang tidak mematuhi aturan perang. Karna menghancurkan semua senjata Abimanyu dan mencoba mengalahkannya dengan tangan kosong tetapi pasukan korawa menghiraukan peraturan dan menyerang secara serentak hingga abimanyu gugur. Arjuna yang mendengar berita kematian Abimanyu merasa sangat sedih dan sakit hati, bersumpah akan membunuh Jayadrata dan Karna pada hari berikutnya sebelum matahari tenggelam. Menanggapi hal itu, pihak Korawa menempatkan Jayadrata dan Karna sangat jauh dari Arjuna. Ribuan prajurit dan ksatria mengelilingi dan melindungi Jayadrata. Arjuna berusaha menjangkau Jayadrata dan Karna, namun ribuan pasukan Korawa mengahalanginya. Hingga matahari hampir terbenam,

Jayadrata masih jauh dari jangkauan Arjuna. Melihat hal ini, Kresna menggunakan kecerdikannya. Ia membuat gerhana matahari, sehingga suasana menjadi gelap seolah-olah matahari sudah tenggelam. Pihak Korawa maupun Pandawa mengira hari sudah malam, dan sesuai aturan, mereka menghentikan peperangan dan kembali ke kubu masing-masing. Dengan demikian, pihak Korawa tidak melanjutkan pertarungan sehingga Jayadrata dan Karna tidak dalam perlindungan mereka lagi. Saat matahari muncul lagi, Kresna menyuruh Arjuna agar menggunakan kesempatan tersebut untuk membunuh Jayadrata dan Karna. Arjuna mengangkat busurnya dan meluncurkan panah, memutus leher Jayadrata dan Karna.

Pihak pandawa melakukan peperangan melawan pihak korawa dan akirnya Sengkuni pun gugur dan senopati korawa pun akhirnya maju ke medan perang pihak pandawa nenekan musuh menekan hingga terdesak, pertempuranpun berlangsung seimbang sampai akirnya pasukan senopati terakhir korawa pun gugur.

2.2 *Game*

Dalam bagian ini akan dibahas pengertian *game*, klasifikasi *game* berdasarkan *platform*, berdasarkan *genre*, berdasarkan sudut pandang permainan (*game view point*) dan berdasarkan pembatasan usia dari pemainnya serta komponen penyusun dari *game*.

2.2.1 Pengertian *game*

Game adalah kegiatan yang memiliki tujuan untuk dicapai dan dibatasi dengan peraturan bagi para pemainnya yang bersifat menyenangkan dan untuk membuat *game* maka haruslah memahami teknik dan metode animasi [6]. *Game* bertujuan untuk menghibur, biasanya *game* banyak disukai oleh anak – anak hingga orang dewasa. *Game* sebenarnya penting untuk perkembangan otak, untuk meningkatkan konsentrasi dan melatih untuk memecahkan masalah dengan tepat dan cepat karena dalam *game* terdapat berbagai konflik atau masalah yang menuntut untuk dilesaikan dengan cepat dan tepat.

Game merupakan salah satu teknologi yang perkembangannya sangat pesat pada masa kini di mulai dari *game* sederhana seperti permainan kartu, catur sampai permainan yang di mainkan di beberapa sarana atau perangkat teknologi seperti Playstation 3, Xbox 360, Nintendo Wii, PSP, Nintendo DS, maupun PC dari yang berbasis individu ataupun *multiplayer* yang lebih kompleks.

2.2.2 Klasifikasi *game*

Game bisa diklasifikasikan kedalam beberapa hal, diantaranya klasifikasi berdasarkan *platform*, berdasarkan *genre*, berdasarkan sudut pandang permainan (*game view point*) dan berdasarkan pembatasan usia dari pemainnya.

2.2.2.1 Berdasarkan Platform

Tiga kategori utama *platform* pada *game* [7] adalah personal computer (PC), console, dan mobile.

1. *Personal Computer (PC)*

   PC game adalah *game* yang dibuat untuk computer baik berbasis Windows, Mac, ataupun Linux. *PC* menyediakan kekuatan grafis dan pemrosesan yang kuat yang memungkinkan pengembang untuk membuat *game* yang mutakhir. Tetapi kelemahan *game PC* adalah mahal karena pengguna harus menggunakan *hardware* yang *up-to-date* untuk memainkan *game PC* dengan baik. Selain itu, kelemahan *game PC* bagi pengembang adalah banyaknya konfigurasi *PC* membuat sulit bagi pengembang untuk memastikan *game* tersebut berjalan dengan benar pada semua *setup PC*.

2. *Console*

   Console adalah *hardware* yang dibuat oleh pihak ketiga seperti *Sony*, *Microsoft*, dan *Nintendo*. *Console* terhubung ke televisi dan tujuan utamanya adalah untuk bermain *game*. *Game Console* sangat menarik bagi *game* pengembang karena mereka hanya perlu memikirkan satu konfigurasi *hardware* ketika membuat *software* untuk konsol. Sangat kontras dengan *PC* yang memiliki opsi konfigurasi yang tak terbatas.

3. *Mobile*

   Mobile platform terdiri atas sesuatu yang *portable* dan bisa digenggam, termasuk ponsel, PDA, iPods, dan *handheld game* seperti *Nintendo* DSi atau *Sony* PSP. *Game mobile* memiliki kontrol yang sederhana jika dibandingkan dengan PC.

2.2.2.2 Berdasarkan Genre

Menurut buku "*Fundamentals of Game Design*", *genre game* bisa dibagi menjadi 9 **[8]**, yaitu:

1. *Action*

   Action game adalah *game* dimana kebanyakan dari tantangan yang disajikan merupakan dari tes *physical skill* dan koordinasi pemain. Salah satu sub-*genre action game* adalah *shooters game*, baik yang 2D maupun 3D seperti *First Person Shooters* (FPS).

2. *Strategy*

   Strategy game menantang pemain untuk mencapai kemenangan dengan perencanaan, khususnya melalui perencanaan serangkaian tindakan yang dilakukan melawan satu lawan atau lebih. Kemenangan diraih dengan perencanaan matang dan pengambilan keputusan yang optimal.

3. *Role Playing Game (RPG)*

   RPG adalah *game* dimana pemain mengontrol satu atau lebih karakter yang biasanya di desain oleh pemain itu sendiri, dan memandu mereka melewati berbagai rintangan yang diatur oleh komputer. Perkembangan karakter dalam hal kekuatan dan kemampuannya adalah kunci dari *game* jenis ini.

4. *Sports*

   Sports game mensimulasikan berbagai aspek dari olahraga atletik nyata maupun imajiner, apakah itu memainkan pertandingan, me-*manage* tim dan karir, atau keduanya. Salah satu contoh *game* jenis ini adalah *Pro Evolution Soccer* 2012 (PES 2012), dimana pemain bisa memainkan pertandingan, menjadi manajer tim, maupun menjadi pemain dan mengembangkan karirnya sendiri.

5. *Vechicle Simulation*

   Vechicle simulation membuat feeling mengendarai kendaraan, baik kendaraan nyata maupun kendaraan imajiner. Performa dan karakteristik kasar mesin harus menyerupai kenyataan, kecuali jika yang didesain adalah kendaraan imajiner.

6. *Construction and Management Simulation*

   CMS *game* adalah *game* tentang proses. Tujuan pemain bukan untuk mengalahkan musuh, tetapi membangun sesuatu dengan konteks proses yang sedang berjalan. Semakin pemain mengerti dan mengontrol proses, semakin sukses sesuatu yang ia bangun. *Game* seperti ini biasanya menyediakan dua jenis permainan, yaitu mode bebas dimana pemain bebas membangun sesuatu, dan mode misi dimana terdapat skenario hal apa yang harus dibangun oleh pemain.

7. *Adventure/ Platform*

   Adventure atau *platform game* adalah cerita interaktif tentang karakter protagonis yang dimainkan oleh pemain. Penyampaian cerita dan eksplorasi adalah elemen inti dari *game* ini. Penyelesaian teka-teki dan tantangan konseptual adalah bagian besar dari permainan.

8. *Artificial Life and Puzzle Game*

   Artificial Life game adalah *game* yang membuat tiruan dari kehidupan sebenarnya. Biasanya ada dua jenis *game* ini, tiruan kehidupan manusia, contohnya The SIMS, dan tiruan kehidupan binatang, contohnya Tamagochi.

9. *Online Game*

   Istilah *online game* disini mengacu kepada *multiplayer game* dimana mesin dari para pemain terhubung dengan jaringan.

2.2.2.3 Berdasarkan Sudut Pandang Pemain (*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*

   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*

   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*

   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*).

2.2.2.4 Berdasarkan Pembatasan Usia Pemain

Menurut ESRB (*Entertainment Software Rating Board*) ada 7 pembatasan usia dari pemainnya [9] adalah sebagai berikut:

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.
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).

2.2.3 Elemen-elemen *Game*

Di dalam sebuah *game,* terdapat elemen-elemen dasar yang menyusunnya [10] antara lain:

1. *Game Rule*

   Game rule merupakan aturan perintah, cara menjalankan, fungsi objek dan karakter di dunia permainan. Dunia *game* bisa berupa pulau, dunia khayal, dan tempat-tempat lain yang sejenis yang dipakai sebagai setting tempat dalam permainan *game*.
2. Plot

   Plot biasanya berisi informasi tentang hal-hal yang akan dilakukan oleh *player* dalam *game* dan secara *detail*, perintah tentang hal yang harus dicapai dalam *game*.
3. Tema

   Di dalam biasanya ada pesan moral yang akan disampaikan.
4. *Character*

   Pemain sebagai karakter utama maupun karakter yang lain yang memiliki ciri dan sifat tertentu.
5. *Object*

   Merupakan sebuah hal yang penting dan biasanya digunakan pemain untuk memecahkan masalah, adakalanya pemain harus punya keahlian dan pengetahuan untuk bisa memainkannya.

6. *Text, Grafik dan Sound*

   Game biasanya merupakan kombinasi dari media teks, grafik maupun suara, walaupun tidak harus semuanya ada dalam permainan *game*.

7. Animasi

   Animasi ini selalu melekat pada dunia *game*, khususnya untuk gerakan karakter, karakter yang ada dalam *game*, properti dari objek.

8. *User Interface*

   Merupakan fitur-fitur yang mengkomunikasikan *user* dengan *game*.

2.3 *Game* Action *Adventure*

Game action adventure dikenal juga sebagai *game* petualangan *arcade* adalah video *game* yang menggabungkan unsur *genre game* petualangan dengan berbagai unsur *game* aksi. *Game action adventure* memerlukan banyak banyak keterampilan fisik yang sama sebagai *game action*, tetapi juga menawarkan alur cerita, karakter yang banyak, sistem inventori dan fitur lainnya dari *game adventure*. *Game action adventure* memiliki waktu yang lebih cepat dibandingkan *game* petualangan murni, karena *game* action *adventure* meliputi tantangan fisik dan konseptual tidak teraku pada petualangan untuk memecahkan misi saja. *Game action adventure* biasanya mencakup kombinasi unsur-unsur cerita yang kompleks yang mana ditampilkan untuk pemain menggunakan audio dan video[8].

Game adventure sangat susah di klasifikasikan, namun ada beberapa sub-genre yang populer pada *game adventure* antara lain:

1. *First Person Action Adventure*, adalah *game* yang menggunakan *First Person shooter gameplay*. *Genre* ini sering juga disebut *Real Time Adventure* atau RTA.
2. *Third-person action-adventure*, adalah *game adventure* yang *game*playnya dilihat dari orang ketiga. Conthnya: Tomb Raider, The Legend of Zelda dan masih banyak lagi.

3. *Isometric platform games*, adalah *game adventure* yang memiliki *enviroment* yang bebas dengan *gameplay* 3 dimensi dan grafis 2 dimensi dengan menggunakan proyeksi isometrik.
4. *Sandbox/Open World action-adventure*, adalah *game adventure* yang berfokus pada nonlinier *game*play contohnya seperti Assassin's Creed.
5. *Platform-adventure games*, adalah *game adventure* yang menekankan kepada eksplorasi dan memecahkan teka-teki, *genre* ini juga menekankan pada fitur tradisional *game conventions*.
6. *Stealth games*, adalah *game adventure* yang *game*playnya lebih pada menghindari musuh dari pada menyerangnya dan lebih mengarah pada penjelajahan dan memecahkan teka-teki.
7. *Survival horror games*, adalah *game adventure* yang *game*playnya menekankan pada "*inventory management*" dan memastikan pemain memiliki cukup amunisi dan item pemulihan untuk bertahan hidup. Contoh dari *game* genre ini adalah The Resident Evil.

2.4 *Game* Mobile

Mobile game merupakan *game* yang tidak hanya dapat dimainkan melalui telepon seluler, namun dapat dikembangkan dalam berbagai macam *mobile handset* seperti PDA, symbian OS dan Microsoft *Smartphone*. *Mobile game* dimainkan dengan menggunakan teknologi yang ada di dalam alat itu sendiri.

Dalam pembuatan *game mobile* ada beberapa hal yang harus diperhatikan agar *game* berjalan dengan baik pada *device mobile*. Hal yang harus diperhatikan adalah antara lain ukuran panjang dan lebar *(width and height) game*, ukuran *game*, dan kontrol *game*. Ukuran panjang dan lebar sangat berpengaruh pada grafis yang penting untuk tampilan *game* ukuran panjang dan lebar bisa disesuaikan ukurannya. Ukuran *game* sebaiknya tidak terlalu besar karena sangat berpengaruh pada saat *game* dijalankan, karena saat *game* dijalankan maka akan ada memori yang termakan, performa *game* akan melambat jika memori yang dipakai banyak dan kapasitas memori sedikit. Kontrol *game* pada *game* mobile ada dua cara yaitu dengan *touchscreen* dan *keypad* kontrol *game* tergantung pada jenis *device* mobile yang dipakai, kontrol *game* yang menggunakan *touchscreen*

tidak dapat dimainkan pada *device* yang menggunakan *keypad* karena input yang berbeda.

Dalam pembuatan *game* berbasis *mobile* ini akan digunakan *device mobile* yang sistem operasinya berbasis Android. Android adalah sistem operasi yang digunakan di *smartphone* dan juga tablet PC. Android dibuat dengan basis kernel Linux yang telah dimodifikasi, dan untuk setiap release-nya diberi kode nama berdasarkan nama hidangan makanan. Keunggulan utama Android adalah gratis dan *open source*. Keuntungan *open source*, banyak pengembang *software* yang bisa melihat dan memanfaatkan kode itu serta bisa membuat aplikasi baru di dalamnya. Berbagai aplikasi android diwadahi dalam sebuah portal, yaitu *Play Store*, sehingga pengguna tinggal meng-*install* aplikasi pilihannya.

2.5 AI (Artificial Intelligence)

Kecerdasan buatan atau dalam bahasa Inggrisnya *Artificial Intelligence* sering disingkat AI merupakan cabang terpenting dalam dunia komputer. Akhir-akhir ini teknologi AI telah begitu banyak mempengaruhi kehidupan 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.

AI berhubungan dengan penggunaan komputer untuk melaksanakan tugas atau menyelesaikan masalah yang memerluakan kecerdasan atau kepintaran. Pada umumnya komputer memerlukan aplikasi atau software untuk melaksanakan tugas atau menyelesaikan masalah, maka aplikasi atau software yang berdasarkan teknologi AI diperlukan.

Para ahli mendefinisikan AI secara berbeda-beda tergantung pada sudut pandang mereka masing-masing. Ada yang focus pada logika berpikir manusia saja, tetapi ada juga yang mendefinisikan AI secara lebih luas pada tingkah laku manusia. Stuart Russel dan Peter Norvig mengelompokkan definisi AI, yang diperoleh dari beberapa textbook berbeda, ke dalam empat kategori [11], yaitu :

1. *Thinking humanly : the cognitive modeling approach*
2. *Acting humanly : the Turing test approach*
3. *Thinking rationally : the laws of thought approach*
4. *Acting rationally : the rational agent approach*

Thinking humanly dan *acting humanly* adalah dua definisi dalam arti yang sangat luas. Sampai saat ini, pemikiran manusia yang diluar rasio, yakni refleks dan intuitif (berhubungan dengan perasaan), belum dapat ditirukan sepenuhnya oleh komputer. Dengan demikian, kedua definisi ini dirasa kurang tepat untuk saat ini. Jika kita menggunakan definisi ini, maka banyak produk komputasi cerdas saat ini yang tidak layak disebut sebagai produk AI.

Definisi *thinking rationally* terasa lebih sempit daripada *acting rationally*. Oleh karena itu, definisi AI yang paling tepat untuk saat ini adalah *acting rationally* dengan pendekatan *rational agent*. Hal ini berdasarkan pemikiran bahwa komputer bisa melakukan penalaran secara logis dan juga bisa melakukan aksi secara rasional berdasarkan hasil penalaran tersebut.

2.5.1 AI yang Digunakan pada *Game*

Algoritma A* (A *Star*) merupakan algoritma *Best First Search* yang menggabungkan *Uniform Cost Search* dan *Greedy Best First Search*. Algoritma A* (A Star) adalah *graph search algorithm* yang mencari *path* (jalur) dari titik awal yang diberikan menuju titik tujuan. Algoritma A* pada *game* umumnya diterapkan pada NPC untuk pencarian jalur player dalam area permainan. Metode A* adalah metode yang merupakan hasil pengembangan dari metode dasar *Best First Search*, hanya saja dengan dua faktor tambahan.

1. Setiap sisi mempunyai "*cost*" yang berbeda-beda, seberapa besar *cost* untuk pergi dari suatu simpul ke simpul yang lain.
2. *Cost* dari setiap simpul ke simpul tujuan bisa diperkirakan. Ini membantu pencarian, sehingga lebih kecil kemungkinan mencari ke arah yang salah.

Metode A* mengevaluasi setiap simpul dengan mengkombinasikan biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan.

$$f(n) = g(n) + h(n) \quad (2.1)$$

f(n) adalah jumlah dari g dan h yang merupakan estimasi terbaik dari biaya / *cost* jalan yang akan melalui simpul saat ini. Intinya semakin rendah nilai f maka akan lebih efisien. g(n) adalah biaya / *cost* untuk mencapai simpul n dari simpul awal, dan h(n) merupakan singkatan dari heuristik yang merupakan perkiraan biaya dari simpul n ke simpul tujuan. Untuk menghitung h digunakan fungsi heuristik, ada beberapa fungsi heuristik yang umumnya digunakan didalam algoritma A* yaitu:

1. *Manhattan Dinstance*

   Manhattan Distance adalah fungsi standar untuk algoritma A* yang digunakan untuk aplikasi yang memiliki 4 arah gerakan.

   $$h(n)= abs(nx-goal.x) + abs (ny-goal.y) \quad (2.2)$$

   dimana:

   nx adalah koordinat x dari simpul pertama pada grid.

   Goal.x adalah koordinat X pada simpul tujuan.

   ny adalah koordinat dari simpul pertama pada grid.

   Goal.y adalah koordinat Y pada simpul tujuan.

2. *Straight Line Distance*

   Straight Line Distance adalah fungsi heuristik yang digunakan pada aplikasi yang dapat bergerak ke segala arah/sudut.

   $$h(n)= sqrt ((nx-goal.x + abs (ny-goal.y) \quad (2.3)$$

   dimana:

   nx adalah koordinat x dari simpul pertama pada grid.

   Goal.x adalah koordinat X pada simpul tujuan.

   ny adalah koordinat dari simpul pertama pada grid.

   Goal.y adalah koordinat Y pada simpul tujuan.

3. *Diagonal Distance*

   Diagonal Distance adalah fungsi heuristik yang digunakan pada aplikasi yang yang memiliki delapan arah gerakan (dapat bergerak diagonal).

   $$h(n)= d* max (abs (nx-goal.x), abs (ny-goal.y)) \quad (2.4)$$

   dimana:

   d adalah nilai biaya. Nilai d didapat antara nilai minimum *cost* di pindah antar simpul

nx adalah koordinat x dari simpul pertama pada grid.

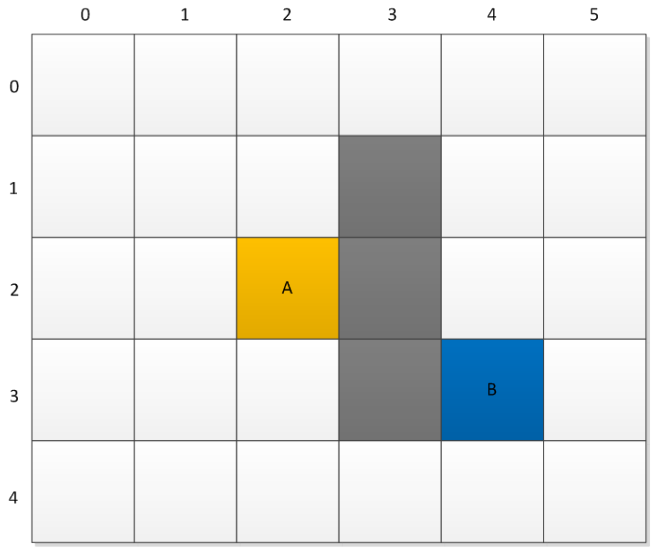
Goal.x adalah koordinat X pada simpul tujuan

ny adalah koordinat dari simpul pertama pada grid.

Goal.y adalah koordinat Y pada simpul tujuan.

Fungsi heuristik yang akan digunakan dalam pembanguna *game* Adventure lailah arjuna adalah fungsi heuristik *manhattan distance*, sebagai penghitung biaya perkiraan yang akan dihitung.

Cara kerja algoritma A* dapat digambarkan dengan sesorang yang ingin berjalan dari simpul A ke simpul B, dimana diantaranya terdapat hambatan. Dimana simpul A ditunjukan dengan kotak bewarna jingga, sedangkan simpul B ditunjukkan dengan kotak bewarna biru dan korak bewarna biru mewakili hambatan yang memisahkan kedua simpul tersebut. Area pencarian terdiri grid atau kumpulan tile yang memilikin orde x×y. Tiap *grid*-nya memiliki koordinat x dan y atau berbentuk seperti array dua dimensi seperti yang ditunnjukan pada gambar berikut.



Gambar 2-1 Posisi Simpul A dan Simpul B

Pencarian dengan menggunakan algoritma A* ada beberapa istilah-istilah yang akan digunakan yaitu *open list* dan *closed list*. *Open list* adalah *list* yang menyimpan kemungkinan jalur yang akan diperiksa. *open list* dibuat berturut berdasarkan nilai f. *Open list* merupakan list yang berisi semua simpul yang akan diperiksa yang bertetangga dengan simpul awal. *Closed list* adalah *list* untuk

menyimpan simpul-simpul yang sudah pernah diperiksa dan sudah terpilih sebagai simpul terbaik atau *list* yang menyimpan jalan yang sudah diperiksa dari *open list. Open list* dan *closed list* ini bertujuan untuk menghindari penelusuran berkali-kali jalan yang memang sudah diidentifikasi agar tidak masuk kembali kedalam *open list*.

Pencarian jalur A*, dimulai dari simpul A, memeriksa simpul-simpul yang berdekatan, dan mencari sampai tujuan ditemukan. Pencarian dilakukan dengan tahap sebagai berikut:

1. Simpul A yang merupakan simpul awal dimasukkan ke *open list.*
2. Simpul-simpul yang dapat dilalui dan yang berhubungan dengan simpul A akan diperiksa dengan cara di hitung nilai f-nya dan dimasukkan ke dalam *open list.*
3. Buang simpul A dari *open list* dan masukkan ke *closed list*
4. Setelah simpul-simpul yang berhubungan dengan simpul A diperiksa, maka pilih satu simpul yang memiliki nilai biaya terkecil sebagai *bestnode*
5. Simpul *bestnode* yang terpilih dan simpul-simpul yang telah diperiksa akan dimasukkan ke dalam *closed list* dan simpul-simpul yang berhubungan dengan simpul *bestnode* akan diperiksa untuk menjadi *bestnode* selanjutnya
6. Tahapan tersebut akan dilakukan berulang-ulang sampai simpul A akhirnya mencapai simpul B.

Tahapan pencarian lebih jelasnya akan dijelaskan dengan Gambar 2-1, dimana simpul A dengan koordinat (2,2) akan melakukan pencarian jalur untuk mencapai simpul B dengan koordinat (4,3). Langkah selanjutnya akan dilakukan pemeriksaan terhadap simpul-simpul yang bertetangga yang memiliki nilai *cost* / biaya terkecil sebagai *bestnode* sebagai rute yang dipilih simpul A dengan melakukan langkah perhitungan dengan fungsi 2.1. Dimana g akan diberi nilai 1 untuk setiap pergerakannya dan untuk menghitung h akan digunakan fungsi heuristik manhattan distance dimana hanya bisa melakukan pergerakan horizontal dan vertikal.

Setelah diperiksa didapatkan tiga simpul yang dapat dilalui oleh simpul A yaitu simpul dengan koordinat (2,1), (1,2) dan (2,3), simpul dengan koordinat

(3,2) tidak dapat dilalui karena terdapat penghalang yang tidak dapat dilalui musuh, langkah perhitungan untuk mendapatkan *bestnode* sebagai berikut:

1. Langkah pertama
    a. Koordinat (2,1)

    Nilai nx = 2

    Nilai ny = 1

    Simpul tujuan (4,3) sehingga goal.x = 4 dan goal.y = 3

    g(2,1) = 1

    h(2,1) = abs(nx-goal.x) + abs (ny-goal.y)
    = abs(2-4) + abs(1-3)
    = abs(-2) + abs(-2)
    = 2 + 2
    = 4

    f(2,1) = g(2,1) + h(2,1)
    = 1 + 4
    = 5

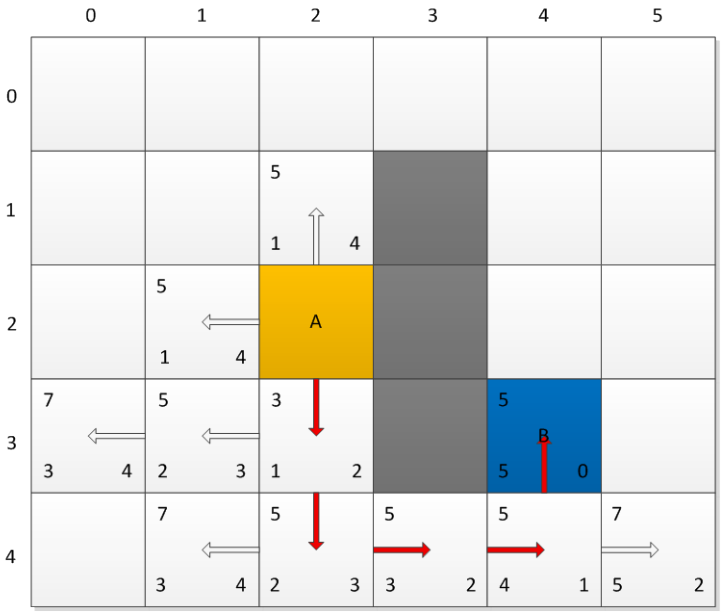
Pergitungan dengan cara yang sama akan dilakukan simpul (1,2) dan (2,3) dan hasilnya bisa dilihat pada tabel berikut.

Tabel 2-1 Langkah Pertama Perhitungan f

Iterasi	Titik	g	h	f
0	2,1	1	4	5
	1,2	1	4	5
	2,3	1	2	**3**
	3,2	-	-	-

Dari hasil perhitungan yang dilakukan terhadap tiga simpul, yang memiliki nilai cost / biaya terkecil adalah simpul dengan koordinat (2,3) maka dipilihlah simpul (2,3) sebagai bestnode dan dimasukkan ke closed list agar tidak diperiksa ulang.

Proses yang dilakukan diatas dilakukan berulang kali atau looping sehingga ditemukan simpul tujuan yang dicari. Hasil akhir pencarian jalur terpendek yang akan dilalui simpul A untuk menuju simpul b terlihat pada Gambar 2-2.



Gambar 2-2 Hasil Akhir Pencarian Algoritma A*

2.6 *Object Oriented Analysis and Design* (OOAD)

Object Oriented Analysis (OOA) adalah metode analisis yang memeriksa requirement (syarat/keperluan yang harus dipenuhi suatu sistem) dan sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup permasalahan. Yang menjadi sasaran OOA adalah mengembangkan sederatan model yang menggambarkan perangkat lunak komputer pada saat perangkat itu bekerja untuk memenuhi serangkaian persyaratan yang ditentukan oleh pelanggan. Seperti metode analisis konvensional lainnya .

Object Oriented Design (OOD) adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem. OOD mnerjemahkan model OOA dari dunia nyata kedalam model implementasi spesifik yang dapat direalisasikan di dalam perangkat lunak. Proses OOD dapat digambarkan sebagai piramida yang terdiri dari 4 (empat) lapisan. Lapisan dasar berfokus pada design subsistem yang mengimplementasikan fungsi-fungsi sitem mayor, lapisan kelas mengkhususkan keseluruhan arsitektur objek dan hirarki kelas yang diperlukan untuk mengimplementasi suatu sistem, lapisan pesan menunjukan bagaimana kolaborasi antar objek akan direalisasikan, dan

lapisan tanggung jawab mengidentifikasikan atribut dan operasi yang menandai masing-masing kelas[12].

2.6.1 Konsep-konsep OOAD

OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaiut analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memerika *requirement* (syarat/keperluan) yang harus dipenuhi sebuah sistem) dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup perusahaan. Sedangkan OOD adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem. Terdapat beberapa konsep dalam OOAD [12], yaitu :

1. *Object*

   Objek adalah suatu benda, entitas, biasanya berupa kata benda (*noun*), sesuatu yang dapat diangkat maupun ditendang atau pukul. Beberapa objek berupa benda hidup, dan beberapa tidak. Contoh objek dalam kehidupan nyata sehari-hari adalah sebuah mobil, orang, rumah, meja, seekor anjing, kucing ataupun beruang. Objek memiliki atribut (*attributes*), misalnya: sebuah mobil memiliki atribut manufaktur, nomor model, warna, dan harga, seekor anjing memiliki atribut umur, warna, tinggi maupun berat. Objek juga memiliki kelakuan (*behavior*), misalnya: sebuah mobil dapat bergerak dari satu tempat ke tempat lainnya dan seekor anjing dapat menggongong.

2. Enkapsulasi

   Enkapsulasi merujuk pada sebuah objek yang menyembunyikan atau mengenkapsulasi atribut-atributnya terhadap operasi yang dikenakan (analoginya objek tersimpan dan terkunci dalam sebuah kapsul, dan operasi berada di tepian luar kapsul tersebut). Atribut yang tersembunyi ini disebut dengan *private*. Beberapa bahasa pemograman (sebagai contoh, *Smalltalk*) secara otomatis membuat atribut-atributnya sebagai *private* dan beberapa tidak (sebagai contoh, Java) membiarkan programmernya menentukan sendiri.

3. Asosiasi dan Agregasi

   Setiap objek memiliki kaitan dengan objek lainnya, baik itu secara langsung maupun tidak langsung, kuat maupun lemah. Dengan mengaitkan tiap objek, membuat objek semakin kuat. Kaitan antar objek mengijinkan untuk membantu menemukan informasi dan kelakuan (*behavior*) ekstra. Ketika melakukan pemodelan objek, dapat mengaitkan ke dalam 2 prinsip: Asosiasi atau Agregasi. Terkadang memang sulit untuk mencari letak perbedaan antara asosiasi dan agregasi ini. Berikut sedikit penjelasannya :

   a. Asosiasi merupakan kaitan yang lemah, objek dapat berupa grup atau kelompok, namun objek ini tidak secara komplit tergantung satu dengan lainnya. Sebagai contoh, bayangkanlah sebuah mobil, seorang supir, seorang penumpang dan satu penumpang lainnya. Ketika supir dan dua penumpang berada dalam mobil, mereka berasosiasi dan mereka semua menuju ke arah atau alamat yang sama, mereka menempati tempat yang sama, yakni di dalam mobil. Namun kaitan asosiasi ini sifatnya lemah, supir dapat menurunkan salah satu atau kedua penumpang pada arah atau alamat yang terpisah, jadi penumpang tersebut tidak lagi berasosiasi dengan objek lainnya.

   b. Agregasi berarti menempatkan objek bersama-sama untuk menjadikannya objek yang lebih besar. Barang-barang atau benda-benda manufaktur biasanya merupakan contoh dari bentuk agregasi. Sebagai contoh, sebuah *microwave* terbentuk dari sebuah *cabinet* (semacam lemari kaca), sebuah pintu, sebuah panel indikator, sejumlah tombol, sebuah alat pemanas dan lain sebagainya. Agregasi mengimplikasikan kedekatan ketergantungan.

4. *Class*

   Sebuah kelas mengenkaspsulasi karakteristik bawaan ke dalam sebuah kelompok objek (sekumpulan objek). Beberapa programmer mengatakan “objek merupakan *instance* (diumpamakan) dari sebuah kelas”, oleh karena itu istilah *instance* bisa dikatakan merupakan persamaan kata (*synonym*) dari objek. Kelas merupakan sebuah model, pola, atau *blueprint* (cetak biru) yang digunakan untuk menciptakan sebuah objek. Ambil contoh sebuah rumah

yang dibuat oleh seorang arsitek. Ada banyak sekali rumah di dunia ini, kesemuanya memiliki karakteristik yang sama. Sebuah cetak biru menggambarkan semua karakteristik yang dimiliki oleh sebuah rumah, misalnya dinding, jendela, saluran listrik, dan lain sebagainya. Dengan menggunakan cetak biru tersebut, maka dapat dibuat banyak rumah. Rumah-rumah yang dibangun dari cetak biru itu disebut dengan *instance* dari kelas. *Instance* dari kelas adalah objek yang dibentuk dengan menggunakan kelas sebagai dasar pembentukannya. Rumah-rumah tersebut yang terbentuk dari cetak biru yang ada dapat berbeda satu dengan lainnya, misalnya: lokasi, bahan yang digunakan untuk membangun rumah, bentuk dan ukuran, dan lain sebagainya. Tetepi mereka dapat dikatakan memiliki karakteristik yang sama.

5. Pewarisan *(Inheritance)*

   Pewarisan mengijinkan untuk menspesifikasikan atau mengkhususkan bahwa sebuah kelas bisa memiliki atau mendapatkan beberapa karakteristik dari kelas induknya (*parent class*) dan menambahkan fitur atau kegunaan khusus atas dirinya sendiri. Kelas anak (*subclass*) mewariskan (*inherits*) semua *fields, messages, dan methods* dari kelas induk utama (*superclass*). Pewarisan menjadikan kelas dapat di-grupkan atau dikelompokkan ke dalam kelas yang lebih general (umum), oleh sebab itu pewarisan sering disebut juga sebagai bentuk spesialisasi atau generalisasi atas objek. Dalam perspektif pemograman, pewarisan diperlukan karena:

   a. Pewarisan mendukung perluasan dan kekuatan yang lebih dalam proses pemodelan perangkat lunak.
   b. Pewarisan mengijinkan untuk mendefenisikan informasi atau atribut dan kelakuan (*behavior*) pada sebuah kelas dan membagikannya (*share*) kepada kelas anak (*subclass*) yang berhubungan, sehingga *code* yang dimiliki akan lebih efisien dan *reuseable*.
   c. Pewarisan merupakan hal yang natural, yang gampang dimengerti dan didefenisikan dalam kehidupan nyata

6. Polimorfisme *(Polymorphism)*

Polimorfisme diambil dari bahasa Yunani yang diturunkan dari kata *poly*, yang berarti banyak, dan *morph* yang berarti bentuk. Maka dari itu polimorfisme berarti ‘memiliki banyak bentuk/rupa’. Dalam kehidupan sehari-hari, antara satu objek dengan objek lainnya sering terjadi komunikasi. Satu objek mengirmkan pesan ke objek lainnya untuk mendapatkan sesuatu yang dinginkan, namun tanggapan dari objek-objek tersebut dapat beragam walaupun pesan yang disampaikan adalah pesan yang sama.

2.6.2 OOP (*Object* Oriented Programming)

OOP (*Object Oriented Programming*) atau yang dikenal dengan Pemrograman Berorientasi Objek merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah tidak harus melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh suatu departemen yang memiliki *manager*, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bagian administrasi maka *manager* tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang *manager* tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri **[13]**.

2.6.2.1 Istilah-istilah OOP

Istilah OOP sudah sering didengar dalam ilmu pemrograman. Berikut ini istilah-istilah yang sering kali terdengar dalam teknik pemrograman OOP :

1. *Object*

   Objek memiliki atribut sebagai status (*state*) dan tingkah laku sebagai *behavior*. Di dalam OOP, state disimpan pada variabel dan tingkah laku disimpan pada method. Dalam bahasa teoretis OOP, Objek berfungsi untuk membungkus data dan fungsi bersama menjadi satu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

2. *Class*

   Class adalah suatu frame yang merupakan definisi yang memuat data dan metod pengolah data. *Class* juga dapat diartikan sebagai tempat untuk membuat obyek. Di dalam *class* dideklarasikan variable dan method yang dimiliki oleh obyek. Proses pembuatan obyek dari sebuah *class* disebut dengan instantiation. Jadi obyek merupakan hasil instansiasi dari *class*. Obyek disebut juga dengan instance. *Class* memiliki anggota yang disebut Anggota *Class* (*class member*) yaitu atribut dan *method*.

3. *Attributes*

   Atribut adalah data yang membedakan antara objek satu dengan yang lainnya. Contoh Objek : Volcano Robot (*a volcanic exploration vehicle*), mempunyai atribut sebagai berikut :

   Status → exploring, moving, returning home

   Speed → in miles per hour

   Temperature → in Fahrenheit degrees

   Dalam *class*, atribut sering disebut sebagai variabel. Atribut dibedakan menjadi dua jenis yaitu *Instance Variable* dan *Class Variable*. *Instance variable* adalah atribut untuk tiap objek dari kelas yang sama. Tiap objek mempunyai dan menyimpan nilai atributnya sendiri. Jadi, tiap objek dari *class* yang sama boleh mempunyai nilai yang sama atau berbeda. *Class Variable* adalah atribut untuk semua objek yang dibuat dari *class* yang sama.

Semua objek mempunyai nilai atribut yang sama. Jadi semua objek dari *class* yang sama mempunyai hanya satu nilai yang *value* nya sama.

a. *Behavior*

Behavior/tingkah laku adalah hal-hal yang bisa dilakukan oleh objek dari suatu *class*. *Behavior* dapat digunakan untuk mengubah nilai atribut suatu objek, menerima informasi dari objek lain, dan mengirim informasi ke objek lain untuk melakukan suatu tugas (*task*).

b. Abstraksi

Abstraksi adalah kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

2.6.2.2 Konsep-konsep OOP

Konsep dasar dari Pemrograman Berorientasi Objek Pemrograman orientasi-objek menekankan konsep berikut:

1. Enkapsulasi

   Enkapsulasi berfungsi untuk memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam/dari sebuah objek dengan cara yang tidak layak, hanya metode dalam objek tersebut yang diberi izin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

2. Pewarisan *(Inheritance)*

   Pewarisan merupakan pewarisan atribut dan method dari sebuah *class* ke *class* lainnya. *Class* yang mewarisi disebut *superclass* dan *Class* yang diwarisi disebut *subclass*. *Subclass* bisa berlaku sebagai *superclass* bagi *class* lainya, disebut sebagai *multilevel inheritance*. Prinsip dasar *inheritance* yaitu

persamaan-persamaan yang dimiliki oleh beberapa kelas dapat digabungkan dalam sebuah *class* induk sehingga setiap kelas yang diturunkannya memuat hal-hal yang spesifik untuk kelas yang bersangkutan. Keuntungan pewarisan:

a. *Subclass* menyediakan *state/behaviour* yang spesifik yang membedakan dengan *superclass*, sehingga memungkinkan *programmer* untuk menggunakan ulang *source code* dari *superclass* yang telah ada.
b. *Programmer* dapat mendefinisikan *superclass* khusus yang bersifat generik, yang disebut *abstract class* (abstraksi), untuk mendefinisikan *class* dengan tingkah laku dan *state* secara umum.

3. Polimorfisme

Poolimorfisme adalah kemampuan suatu obyek untuk mempunyai lebih dari satu bentuk. Polimorfisme tidak bergantung kepada pemanggilan subrutin. Metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesan tersebut dikirim. Sebuah variabel tunggal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama, hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

2.7 UML (Unified Modeling Language)

Pada perkembangan teknik pemrograman berorientasi objek, muncul sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modelling Language* (UML). Adapun pengertian dari UML adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek [14].

UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. Dalam hal ini UML merupakan suatu bahasa visual untuk

melakukan pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

UML menggunakan berbagai macam diagram dengan fungsi masing-masing untuk menggambarkan setiap proses dari sistem berorientasi objek. UML memiliki 13 jenis diagram. Dalam pembangunan *game* ini dipakai 4 diagram UML diantaranya :

1. *Use Case Diagram*

Use Case atau diagram *use case* merupakan pemodelan yang digunakan untuk menggambarkan kelakuan (*behavior*) dari sistem yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case* [14].

a. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antarunit atau aktor.

Sebuah *use case* dapat meninclude fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang me-*include* dieksekusi secara normal. *Use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. *Use case* juga dapat me-e*xtend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. Contoh dari *use case diagram* dapat dilihat pada gambar berikut.

2. *Activity Diagram*

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

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal[14].

3. *Sequence Diagram*

Diagram *sequence* adalah diagram yang menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek [14]. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram *sequence* yang harus digambarkan adalah sebanyak pendefinisian *use case* yang memiliki proses itu sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.

4. *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

a. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
b. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

2.8 Tools Yang Digunakan Untuk Pembuatan *Game*

Tools yang digunakan dalam pembuatan *Game Adventure* "Larilah Arjuna" diantaranya adalah *ActionScript* 3.0 dan Adobe Flash.

2.8.1 ActionScript

ActionScript adalah bahasa pemrograman yang dibuat berdasarkan *ECMAScript,* yang digunakan dalam pengembangan situs web dan perangkat lunak menggunakan *platform* Adobe Flash Player. *ActionScript* juga dipakai pada beberapa aplikasi basis data, seperti *Alpha Five*. Bahasa ini awalnya dikembangkan oleh Macromedia, tapi kini sudah dimiliki dan dilanjutkan perkembangannya oleh Adobe, yang membeli Macromedia pada tahun 2005.

ActionScript terbaru saat ini adalah *ActionScript* 3.0 dari edisi yang sebelumnya dikenal dengan *ActionScript* 2.0. *ActionScript* 3.0 adalah bahasa berorientasi objek untuk membuat aplikasi dan konten multimedia ber-*script* untuk pemutaran di *runtime* Flash *client* (seperti Flash Player dan Adobe AIR). Dengan sintaks yang mirip dengan Java dan C#. *ActionScript* 3.0 memiliki beberapa kelebihan dibanding pendahulunya, antara lain fitur yang ditawarkan adalah file pada *ActionScript* 3.0 dapat dibuat terpisah saat *runtime* [15].

2.8.2 Adobe Flash

Adobe Flash merupakan sebuah program yang didesain khusus oleh Adobe dan program aplikasi standar authoring tool professional yang digunakan untuk membuat animasi dan bitmap yang sangat menarik untuk keperluan pembangunan situs web yang interaktif dan dinamis. Flash didesain dengan kemampuan untuk membuat animasi 2 dimensi yang handal dan ringan sehingga flash banyak digunakan untuk membangun dan memberikan efek animasi pada *website,* CD

Interaktif dan yang lainnya. Selain itu aplikasi ini juga dapat digunakan untuk membuat animasi logo, *movie*, *game*, pembuatan navigasi pada situs web, tombol animasi, banner, menu interaktif, interaktif form isian, *e-card*, *screen saver* dan pembuatan aplikasi-aplikasi web lainnya.

Dalam Flash, terdapat teknik-teknik membuat animasi, fasilitas *action script, filter, custom easing* dan dapat memasukkan video lengkap dengan fasilitas *playback* FLV. Keunggulan yang dimiliki oleh Flash ini adalah ia mampu diberikan sedikit *code* pemograman baik yang berjalan sendiri untuk mengatur animasi yang ada didalamnya atau digunakan untuk berkomunikasi dengan program lain seperti HTML, PHP, dan Database dengan pendekatan XML, dapat dikolaborasikan dengan web, karena mempunyai keunggulan antara lain kecil dalam ukuran *file output*-nya.

Flash bekerja dengan bahasa pemprograman yang dinamakan *ActionScript*. Bahasa ini telah mengalami beberapa kali perkembangan sejak pertama kali diperkenalkan. Versi terakhir adalah *ActionScript* 3.0 atau disingkat AS3, dengan struktur bahasa yang sangat menyerupai *javascript*.

Berkas yang dihasilkan dari perangkat lunak ini mempunyai *file extension* .swF. SWF adalah kependekan dari "*Shock Wave Flash*", kata lain yang biasanya memiliki ekstensi .swf dipergunakan untuk menyebutkan file swf adalah *Flash Movie*. *File Flash Movie* dan dapat diputar di penjelajah web yang telah dipasangi Adobe Flash Player. *File* swf juga didapatkan pada *file game* yang dibuat oleh flash. Dalam Adobe Flash terbaru terdapat fungsi *export* yang dapat meng-*export file* kedalam bentuk .apk eksensi yang biasa digunakan dalam android

Flash juga dapat digunakan untuk mengembangkan secara cepat aplikasi-aplikasi web yang kaya dengan pembuatan *script* tingkat lanjut. Di dalam aplikasinya juga tersedia sebuah alat untuk men-*debug script*, dengan menggunakan *code hint* untuk mempermudah dan mempercepat pembuatan dan pengembangan isi *ActionScript* secara otomatis [16].

2.9 Metode Pengujian Sistem

Metode pengujian sistem untuk mengetahui efektifitas dari *software* yang digunakan selain memberikan kesempatan kepada pengguna untuk mengoperasikan dan melakukan pengecekan terhadap laporan yang dihasilkan melalui *software*. Metode pengujian sistem terdiri dari Pengujian *Whitebox* dan Pengujian *Blackbox* [17].

2.9.1 Pengujian *Whitebox*

Pengujian *whitebox* dilakukan untuk menguji prosedur-prosedur yang ada. Lintasan lojik yang dilalui oleh setiap bagian prosedur diuji dengan memberikan kondisi/loop spesifik. Pengujian *whitebox* menjamin pengujian terhadap semua lintasan yang tidak bergantungan minimal satu kali, mencoba semua keputusan lojik dari sisi '*true*' dan '*false*', eksekusi semua *loop* dalam batasan kondisi dan batasan operasionalnya dan pengujian validasi data internal. Konsep Pengujian Basis *Path* sebagai berikut*:*

1. Merupakan bagian dari pengujian *whitebox* dalam hal pengujian prosedur-prosedur.
2. Mempergunakan notasi aliran *graph* (*node*, *link* untuk merepresentasikan *sequence*, *if*, *while*, *until* dan sebagainya).
3. Konsep kompleksitas *cyclomatic* antara lain cara perhitungan daerah tertutup pada graph planar dimana dapat menghubungkan batas atas jumlah pengujian yang harus direncanakan dan dieksekusi untuk menjamin pengujian seluruh *statement* program.
4. Memunculkan kasus-kasus yang akan diuji dengan membuat daftar lintasan kasus pengujian berdasarkan kompleksitas dan *cyclomatic* yang didapat.
5. Membuat alat bantu graph matriks yang membantu pengawasan pengujian.

2.9.2 Pengujian *Blackbox*

Pengujian yang dilakukan untuk antarmuka perangkat lunak, pengujian ini dilakukan untuk memperlihatkan bahwa fungsi-fungsi bekerja dengan baik dalam arti masukan yang diterima dengan benar dan keluaran yang dihasilkan benar-benar tepat, pengintegrasian dari eksternal data berjalan dengan baik.

Metode pengujian *blackbox* memfokuskan pada *requirement* fungsi dari perangkat lunak, pengujian ini merupakan komplenetari dari pengujian *whitebox*. Pengujian *whitebox* dilakukan terlebih dahulu pada proses pengujian, sedangkan pengujian *blackbox* dilakukan pada tahap akhir dari pengujian perangkat lunak. Proses yang terdapat dalam proses pengujian *blackbox* antara lain sebagai berikut:

1. Pembagian kelas data untuk pengujian setiap kasus yang muncul pada pengujian *whitebox*.
2. Analisis batasan nilai yang berlaku untuk setiap data.

2.10 Teknik Pehitungan Kuisioner

Menurut Sugiyono, kuesioner atau angket merupakan teknik pengumpulan data yang dilakukan dengan cara memberi seperangkat pertanyaan atau pernyataan tertulis kepada responden untuk dijawabnya.

Dalam mengolah data kuesioner dalam penelitian ini menggunakan Skala Likert dengan rumus sebagai berikut:

Untuk menghitung data hasil kuesioner diberikan tiga pilihan jawaban dan skornya, dengan ketentuan seperti Tabel 2-2 :

Tabel 2-2 Kriteria Skor

Kriteria	Skor
S (Setuju)	3
SD (Sedang)	2
TD(Tidak Setuju)	1

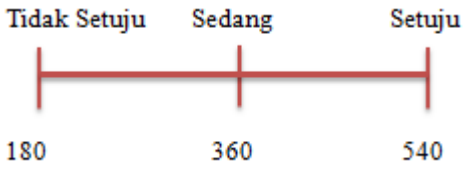
Untuk menghitung jumlah skor ideal (kriterium) dari seluruh item, digunakan persamaan (2.5) berikut.

Skor Kriterium=(Nilai tertinggi x Jumlah soal) x jumlah responden (2.5)

Dengan skor tertinggi adalah 3 (seandainya seluruh responden menjawab“S”) dengan jumlah pertanyaan 6 dan jumlah responden 30, maka dapat dirumuskan dengan persamaan (2.6) berikut.

$$(3\times6)\times30=540 \quad (2.6)$$

Selanjutnya, skor yang telah diperoleh kemudian dimasukkan kedalam *rating scale* seperti yng terlihat pada gambar 2-3



Gambar 2-3 Rating Scale

Rating scale berfungi untuk mengetahui hasil data angket secara umum dan keseluruhan yang didapat dari penilaian angket.

Sedangkan untuk mengetahui jumlah jawaban dari para responden melalui prosentase, yaitu digunakan rumus sebagai berikut :

$$P = \frac{f}{n} \times 100\% \qquad (2.7)$$

Keterangan :

P : Prosentase

f : Frekuensi dari setiap jawaban angket

n : Jumlah responden

100: Bilangan tetap