# BAB II
# TINJAUAN PUSTAKA

## II.1 Permainan

### II.1.1 Definisi Permainan

Permainan terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk mengoptimalkan kemenangan sendiri atau meminimalkan kemenangan lawan. Peraturan menentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai aturan dalam bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi [4] (J. Von Neumann and O. Morgenstern, 1953).

Permainan video umumnya menyediakan sistem penghargaan, misalnya *score* (nilai) 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*).

1. Permainan komputer (*Computer Game*)

   Permainan computer 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 *online* 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* (klien). Server adalah penyedia layanan permainan yang merupakan dasar agar klien yang terhubung dapat memainkan permainan dan dapat melakukan komunikasi dengan baik. Suatu server pada prinsipnya hanya pelaksana permainan dan menghubungkan klien-klien. Sedangkan klien adalah pengguna permainan dan terhubung dengan klien lain melalui

Server. Contoh permainan online adalah Dragon nest, Gunbound, Ragnarok Online, Seal dan lain-lain.

2. Permainan Konsol (*Game Console*)

Permainan konsol adalah sebuah bentuk media interaktif yang berfungsi sebagai hiburan. Permainan ini pada umumnya berisi gambar-gambar dan suara-suara yang dihasilkan oleh sebuah alat permainan konsol, dan ditampilkan pada sebuah televisi atau peralatan audio visual yang sejenis. Permainan ini sendiri dapat dikendalikan dengan menggunakan alat kendali yang dipegang yang terhubung pada konsol yang dinamakan *controller* (pengendali). Pengendali ini pada umumnya terdiri dari sejumlah tombol-tombol dan kendali arah yang masing-masing bagian memiliki fungsi tersendiri. Fungsi dari alat pengendali tersebut untuk mengendalikan maupun berinteraksi dengan gambar yang ada di layar.

### II.1.2 Jenis Permainan

Permainan dari tahun pertama dibuat sampai sekarang semakin banyak jenis dan teknologi yang digunakan semakin canggih. Dari semua permainan yang telah dibuat di dunia, permainan tersebut dikelompokkan menjadi beberapa jenis. Beberapa contoh jenis-jenis permainan yang sering dimainkan dikelompokkan sebagai berikut:

**a. *Action Game* (Permainan Aksi)**

Jenis permainan ini memiliki fitur utama berupa banyaknya aksi di mana pemain harus memiliki keterampilan dengan reaksi yang cepat untuk menghindari musuh atau menghindari rintangan. Pengembang permainan jenis ini perlu memastikan permainan yang dibuat dimaksimalkan sehingga pemain memiliki pengalaman dalam bermain yang baik, dan tidak terganggu oleh waktu proses yang lama.

Contoh: Metal Gear Solid

**b. *Adventure Game* (Permainan Bertualang)**

Jenis permainan ini pada umumnya membuat pemain harus berjalan mengelilingi suatu tempat dengan kondisi tertentu, seperti sebuah istana

atau gua yang berkelok dan planet yang jauh. Pemain melakukan navigasi terhadap suatu tempat untuk mencari pesan-pesan rahasia dan memperoleh objek yang memiliki kemampuan yang bervariasi. Untuk membuat permainan ini membutuhkan perencanaan yang akurat sehingga memiliki alur cerita yang menarik bagi pemain.

Contoh: Tomb Raider

**c. *Fighting Game* (Permainan Bertarung)**

Jenis permainan ini pada umumnya pemain bertarung dengan lawan tandingnya dengan cara memukul, menendang, atau menggunakan jurus sampai lawan tandingnya kalah.

Contoh: Tekken, Mortal Kombat, dan lain-lain.

**d. *Sport Game* (Permainan Olahraga)**

Permainan jenis ini berupa kompetisi antara dua pemain atau lebih, di mana pemain dapat berupa individual atau kelompok. Contoh permainan jenis ini adalah sepakbola, bola basket, tenis, bilyard, dan lain-lain. Permainan ini tergantung seberapa lama waktu yang dibutuhkan dalam satu pertandingan.

Contoh: Pro Evolution Soccer

**e. *Role Playing Game* (Permainan Beralur)**

Permainan ini seringkali berupa *multi-player* (Lebih dari satu pemain) di mana setiap pemain memiliki karakter dengan kemampuan, kekuatan, dan kelemahan yang berbeda. Pemain saling berkompetisi, berinteraksi, dan bertempur satu sama lain. Tampilan grafik yang berbeda untuk setiap karakter pemain dan memiliki *storyline* (alur cerita) yang mendebarkan sehingga memberikan pengalaman yang menarik dan berbeda di dalam bermain.

Contoh: Final Fantasy

**f. *Strategy Game* (Permainan Strategi)**

Jenis permainan ini pada umumnya pemain menggunakan taktik dan strategi untuk melawan dan menghancurkan musuhnya dengan membangun

tentara dan alat perang menggunakan sumber daya yang ada secara bijaksana dan teliti.

Contoh: Command & Conquer

**g. *Platform Game***

Permainan ini mengharuskan pemain mengarahkan suatu obyek dengan melalui berbagai tahap atau tingkatan area untuk menyerang musuh dan menghindar terhadap serangan. Jenis permainan ini hampir serupa dengan permainan aksi, tetapi aksi yang dilakukan tidak secepat permainan aksi.

Contoh: Sonic The Hedgehog & Mario Bros

**h. *Puzzle Game* (Permainan Teka-teki)**

Permainan ini pada umumnya membuat pemain menggunakan kemampuan berpikirnya sebagai pengganti keterampilan reaksi yang cepat karena terdapat rahasia yang perlu dipecahkan. Permainan ini lebih bersifat statis dibanding permainan aksi. Pembuatan permainan jenis ini seringkali ditunjang dengan algoritma kecerdasan buatan.

Contoh: 7 Wonders of The Ancient World, Zuma, dan lain-lain.

**i. *Simulation Game* (Permainan Simulasi)**

Jenis Permainan ini memberikan pengalaman atau interaksi secara mirip dengan kendaraan aslinya, meskipun terkadang kendaraan tersebut masih berupa contoh atau bahkan fiktif, tapi ada penekanan khusus pada detail dan pengalaman nyata menggunakan kendaraan tersebut.

Contoh: Gran Turismo

### II.1.3 Permainan Edukasi

Permainan edukasi merupakan permainan yang memberikan pendekatan kepada pengguna untuk belajar. Pengguna dari permainan ini ditujukan kepada anak taman kanak-kanak hingga sekolah dasar. Permainan edukasi ini biasanya berisi permainan berhitung dan permainan mengenal kata, namun masih ada juga edukasi yang lain.

Permainan edukasi online adalah permainan edukasi yang memanfaatkan fasilitas *online* (terhubung dengan internet) dalam mengaplikasikannya. Berikut salah satu contoh dari permainan edukasi secara *online* [5]

(http://www.arcademicskillbuilders.com/). Terdapat beberapa jenis permainan edukasi dengan berbagai kategori, diantaranya:

1. Penjumlahan
2. Pengurangan
3. Perkalian
4. Pembagian
5. Aljabar
6. Desimal
7. Pecahan
8. Bilangan Bulat
9. Uang
10. Waktu
11. Geografi
12. Seni Berbahasa
13. Mengetik

Permainan yang digunakan dalam proses pembelajaran dan dalam permainan harus mengandung unsur mendidik atau nilai-nilai pendidikan. Untuk pemilihan permainan diusahakan agar seluruh aspek yang dimiliki anak dapat berkembang dengan baik, seperti dari segi kognitif, afektif dan juga psikomotorik. Karena itu perlu ditunjang alat bantu yang tepat saat bermain.

### II.1.4 Permainan Memasak

Permainan memasak merupakan permainan yang memberikan pendekatan kepada pengguna untuk belajar tentang memasak. Pengguna dari permainan ini ditujukan kepada anak yang memiliki kegemaran memasak. Permainan memasak biasanya berisi tentang tata cara memasak, bumbu yang digunakan, cara mengolah masakan yang benar. Contoh untuk permainan memasak *online* dapat dijumpai pada [6] (http://www.games.co.id/).

### II.1.5 Kriteria Permainan

Sudah banyak pihak Indonesia yang telah serius mengembangkan permainan pembelajaran karena efektif dalam menghubungkan kegiatan bermain

dan kegiatan pendidikan. Beberapa kriteria pembuatan permainan pembelajaran yang baik sebagai berikut :

**a. Menarik dan Menantang**

Dari judul, awalan, dan isi berikan kesan yang menarik dan menantang bagi pemain agar pemain merasa sedang melakukan misi tertentu.

**b. Tujuan Pembelajaran Jelas**

Pemain haru mengetahui tujuan setiap langkah dari permainan tersebut sehingga menciptakan motivasi belajar bagi pemain. Tujuan pembelajaran harus ditentukan sehingga tidak terlalu mudah dan tidak terlalu sulit.

**c. Persepsi yang baik**

Buatlah kondisi yang nyaman dan dapat dimengerti pemain pada saat memulai permainan.

**d. Komunikatif**

Setiap yang ditampilkan di dalam permainan harus menjadi komunikasi yang baik dengan pemain agar tujuan pembelajaran dipahami pemain.

**e. Ilustrasi dan Animasi Jelas**

Diusahakan ilustrasi dan animasi yang terdapat dalam permainan mendekati kenyataan, sehingga pemain akan merasa dalam kondisi yang nyata dan mudah dalam membayangkan kembali.

**f. Evaluasi dan Umpan Balik Kreatif**

Berikan kepada pemain tes yang kreatif berupa misi, sehingga pemain tidak sadar bahwa sedang dalam proses pembelajaran dan berikan umpan balik yang kreatif, dapat berupa nilai atau hadiah sehingga pemain memiliki motivasi untuk memainkan kembali.

## II.2 Kecerdasan Buatan

### II.2.1 Pengertian Kecerdasan Buatan

Kecerdasan buatan adalah ilmu dan rekayasa, yang membuat mesin mempunyai kecerdasan 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 memerlukan kecerdasan atau kepintaran. Pada umumnya komputer memerlukan aplikasi atau perangkat lunak untuk melaksanakan tugas atau menyelesaikan masalah, maka aplikasi atau perangkat lunak yang berdasarkan teknologi AI di perlukan. Kecerdasan perlu dilihat dari berbagai sudut pandang, pemahaman, inferensi, daya penjelasan, pembelajaran, komunikasi dan lain-lain.

Menurut Stuart, Russel, Peter dan Norvig mengelompokkan definisi kecerdasan buatan dalam empat kategori [7], yaitu :

1. Sistem yang dapat berfikir seperti manusia
2. Sistem yang dapat bertingah laku seperti manusia
3. Sistem yang dapat berfikir secara rasional
4. Sistem yang dapat bertingkah laku rasional

### II.2.2 Tujuan Akhir Kecerdasan Buatan

Menurut Lenat dan Feigenbaum (1992) [8], terdapat sembilan tujuan AI yaitu:

**a.** Memahami kognisi manusia.

Bagaimana manusia dalam menyelesaikan masalah, memiliki ingatan pengetahuan yang mendalam, belajar, membuat keputusan, dan lain-lain.

**b.** Otomatisai biaya-efektif

Menggantikan manusia dalam tugas-tugas intelegensi.

**c.** Penguatan intelegensi biaya dan efektif

Membangun sistem untuk membantu manusia berpikir lebih baik, lebih cepat, lebih dalam, dan lain-lain.

**d.** Intelegensi manusia super

Membangun program yang mempunyai kemampuan untuk melebihi intelegensi manusia.

**e.** Problem-solving umum

Sistem penyelesaian berbagai masalah yang luas.

**f.** Wacana koheren

Komunikasi dengan manusia mengguanakan bahasa, alami.

**g.** Otonomi

Mempunyai sistem intelegensi yang beraksi atas inisiatif sendiri.

**h.** Belajar (induksi)

Sistem sebaiknya dapat untuk memperoleh data sendiri, dan tau bagaimana memperolehnya.

**i.** Informasi

Simpan informasi dan mengetahui bagaimana untuk mengambil informasi.

### II.2.3 Kecerdasan Buatan yang digunakan

Algoritma A* (A *Star*) merupakan algoritma *Best First Search* yang menggabungkan *Uniform Cost Search* dan *Greedy Best First Search*. Biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi matematika dituliskan sebagai : f(n) = g(n) + h(n). Dengan perhitungan biaya seperti ini, algoritma A* (A *Star*) adalah *complete* dan optimal.

Sama dengan algoritma dasar *Best First Search*, algoritma A* (A *Star*) ini juga menggunakan *OPEN* dan *CLOSED*. Terdapat tiga kondisi bagi setiap suksesor yang dibangkitkan, yaitu : sudah berada di *OPEN*, sudah berada di *CLOSED*, dan tidak berada di *OPEN* maupun *CLOSED*.

Pada ketiga kondisi tersebut diberikan penanganan yang berbeda-beda. Jika suksesor sudah berada di *OPEN*, maka dilakukan pengecekan apakah perlu pengubahan *parent* atau tidak tergantung pada nilai g nya melalui *parent* lama atau *parent* baru. Jika melalui *parent* baru memberikan nilai g yang lebih kecil, maka dilakukan pengubahan *parent*. Jika pengubahan *parent* dilakukan, maka dilakukan pula perbaruan (*update*) nilai g dan f pada suksesor tersebut. Dengan perbaruan ini, suksesor tersebut memiliki kesempatan yang lebih besar untuk terpilih sebagai simpul terbaik (*best node*).

Jika suksesor sudah pernah berada di *CLOSED*, maka akan dilakukan pengecekan apakah perlu pengubahan parent atau tidak. Jika ya, maka dilakukan

perbaruan nilai g dan f pada suksesor tersebut serta pada semua "anak cucunya" yang sudah pernah berada di *OPEN*. Dengan perbaruan ini, maka anak cucunya tersebut memiliki kesempatan lebih besar untuk terpilih sebagai simpul terbaik (*best node*).

Jika suksesor tidak berada *di OPEN* maupun *CLOSED*, maka suksesor tersebut dimasukkan ke dalam *OPEN*. Tambahkan suksesor tersebut sebagai suksesornya *best node*. Hitung biaya suksesor tersebut dengan rumus f = g + h [9] (Dr. Suyoto, Intelegensi Buatan: Teori dan Pemrograman).

## II.3 OOP (*Object Oriented Programming*)

Pemrograman berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Pemrograman berorientasi objek merupakan suatu cara bagaimana system perangkat lunak dibangun melalui pendekatan objek secara sistematis. Pemrograman berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Pemrograman berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek dan pengujian berorientasi objek.

Pada saat ini, pemrograman berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat transformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan saat ini sangat beragam (aplikasi bisnis, *real-time*, *utility*, dan sebagainya) dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut [10] (Rosa A.S, Modul Pembelajaran Rekayasa Perangkat Lunak).

Keuntungan menggunakan pemrograman berorientasi objek adalah sebagai berikut:

1. Meningkatkan produktivitas
2. Kecepatan pengembangan

3. Kemudahan pemeliharaan
4. Adanya konsistensi
5. Meningkatkan kualitas perangkat lunak

Berikut ini beberapa bahasa pemrograman yang sudah menggunakan konsep pemrograman berbasis objek adalah:

1. C++
2. Visual C++
3. Visual Basic
4. C#
5. PHP
6. Java

## II.4 Java

Java adalah bahasa pemrograman yang sederhana, berorientasi-objek, bertipe statik, terkompilasi, *multi-threaded*, *garbage collected*, bebas arsitektur, *robust* (tangguh), aman, dan dapat di perluas.

**a.** Sederhana

Pengembangan java banyak membuang fitur-fitur yang tidak di perlukan seperti yang dimiliki oleh bahasa pemrograman tingkat tinggi lainnya.

**b.** Berorientasi-objek

Seperti halnya C++, java menggunakan kelas-kelas untuk mengelola kode ke dalam modul-modul logika.

**c.** Bertipe statik

Semua objek yang digunakan dalam program harus di deklarasikan sebalum digunakan.

**d.** Terkompilasi

Sebelum dapat menjalankan program yang di tulis dengan bahasa pemrograman java, program harus dikompilasi oleh kompilator java.

**e.** *Multi-treaded*

Program java dapat berisi eksekusi multiple treadeds, yang memungkinkan program mengulas citra pada layar pada satu thread, sementara tetap menerima masukan dari keyboard pada thread utamanya.

**f.** *Garbage collected* (pembuang sampah)

Program java menyingkirkan sendiri ‘sampah-sampah’ yang tak berguna.

**g.** Bebas arsitektur

Pada dasarnya, java tidak dirancang untuk prosesor ataupun sistem operasi tertentu.

**h.** *Robust* (tangguh)

Karena interpreter java memeriksa seluruh akses sistem yang dilakukan program, maka program java tidak akan membuat sistem menjadi crash.

**i.** Aman

Sistem java hanya memverifikasi seluruh akses memori tetapi juga menjamin bahwa tidak ada virus yang ikut pada applet yang sedang berjalan.

**j.** Dapat diperluas

Program java mendukung metode native yakni fungsi-fungsi yang dituliskan dalam bahasa lain, biasanya C++ [11] (Tim Penelitian dan Pengembangan Wahana Komputer, Pemrograman Java).

## II.5 UML (*Unified Modeling Language*)

*Unified Modelling Language* (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa

berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

UML menyediakan macam diagram yang dapat digunakan untuk memodelkan aplikasi berorientasi objek yang dibuat (Rosa A.S, Modul Pembelajaran Rekayasa Perangkat Lunak), yaitu :

1. *use case* diagram untuk memodelkan proses bisnis.
2. *class diagram* untuk memodelkan struktur kelas.
3. *statechart diagram* untuk memodelkan perilaku *objects* di dalam sistem.
4. *activity diagram* untuk memodelkan perilaku *Use Cases* dan *objects* di dalam *system*.
5. *sequence diagram* untuk memodelkan pengiriman pesan (*message*) antar *objects*.
6. *collaboration diagram* untuk memodelkan interaksi antar *objects*.
7. *component diagram* untuk memodelkan komponen *object*.
8. *deployment diagram* untuk memodelkan distribusi aplikasi.

## II.6 *Greenfoot*

*Greenfoot* merupakan *software* untuk pemrograman OOP (*Object Oriented Programming*), didukung dengan aplikasi grafis berbasiskan pemrograman Java. *Greenfoot* diprakarsai oleh Michael Kolling pada 2003. Tujuan dibuatnya *Greenfoot* ini awalmulanya adalah menarik minat anak-anak untuk belajar memprogram.

Menggunakan *Greenfoot* sama seperti memprogram biasa berbentuk teks. Namun, *software* ini lebih mudah digunakan karena ada perangkat bantu

seperti class browser, editor, dan compiler. *Greenfoot* juga mendukung bahasa Java secara penuh dan perangkat tersebut membuat agan lebih mudah dalam memahami konsep OOP. Dalam Greenfoot terdapat proyek yang dinamakan scenario. Dalam proyek tersebut, kita bisa membuat dua macam tipe *class* dari *library Greenfoot*, yaitu *class world dan actor*. *World* merupakan latar dari game yang akan dibuat. *Class actor* adalah kelas yang berfungsi sebagai karakter pada game yang akan kita buat [12] (http://www.greenfoot.org/doc).

*Tabel II.1 Greenfoot*

| | |
|---|---|
| World | Metode World tersedia untuk world. |
| Actor | Metode actor tersedia untuk semua turunan actor. |
| Greenfoot | Digunakan untuk berkomunikasi dengan lingkungan Greenfoot sendiri. |
| MouseInfo | Memberikan informasi mouse terakhir. |
| GreenfootImage | Untuk presentasi dan manipulasi gambar. |
| GreenfootSound | Untuk mengendalikan pemutaran suara. |

*Tabel II.2 Penjelasan Kelas Greenfoot*

| **Class Actor** | |
|---|---|
| **Actor**() | Konstruktor Actor. |
| void **act**() | Metode act ini dipanggil oleh kerangka Greenfoot memberikan kesempatan untuk melakukan beberapa tindakan objek. |
| protected void **addedToWorld** (World world) | Metode ini dipanggil oleh system Greenfoot ketika objek telah dimasukkan ke dalam world. |

| | |
|---|---|
| GreenfootImage **getImage**() | Mengembalikan gambar yang digunakan untuk mewakili actor ini. |
| protected List **getIntersectingObjects**(Class cls) | Mengembalikan semua objek yang berpotongan objek ini. |
| protected List **getNeighbours**(int distance, boolean diagonal, Class cls) | Para tetangga mengembalikan ke objek ini pada jarak yang diberikan. |
| protected List**getObjectsAtOffset**(int dx, intdy, Class cls) | Mengembalikan semua objek yang berpotongan dengan lokasi yang diberikan (relative terhadap lokasi obyek ini). |
| protected List **getObjectsInRange**(int r, Class cls) | Mengembalikan semua objek dalam jangkauan 'r' di sekitar objek ini. |
| protected Actor **getOneIntersectingObject**(Class cls) | Mengembalikan sebuah objek yang bersimpangan objek ini. |
| protected Actor **getOneObjectAtOffset**(int dx, intdy, Class cls) | Mengembalikan satu benda yang terletak di sel tertentu (relative terhadap lokasi objek ini). |
| int**getRotation**() | Mengembalikan rotasi yang ada pada objek. |
| World **getWorld**() | Mengembalikan world pada objek ini berada. |
| int**getX**() | Mengembalikan x koordinat pada objek lokasi saat ini. |
| int**getY**() | Mengembalikan x koordinat pada objek lokasi saat ini. |
| protected boolean**intersects**(Actor | Memeriksa apakah objek ini |

| | |
|---|---|
| other) | bersimpangan dengan yang diberikan objek. |
| void **move**(int distance) | Memindahkan actor pada jarak tertentu ke arah saat dia menghadap. |
| void **setImage**(GreenfootImage image) | Mengatur gambar untuk objek ini untuk gambar tertentu. |
| void**setImage**(String filename) | Mengatur gambar untuk objek ini dari file gambar. |
| void**setLocation**(int x, int y) | Menetapkan lokasi baru untuk objek ini. |
| void**setRotation**(int rotation) | Set rotasi objek. |
| void **turn**(int amount) | Memutar actor ini dengan jumlah yang ditentukan (dalam derajat). |
| void **turnTowards**(int x, int y) | Memutar actor untuk menghadap menuju lokasi tertentu. |
| **Class World** | |
| **World**(intworldWidth, intworldHeight, intcellSize) | Konstruktor world baru. |
| **World**(intworldWidth, intworldHeight, intcellSize, boolean bounded) | Membangun sebuah world baru dengan pilihan untuk membuat world yang tak terbatas di mana actor dapat meninggalkan dunia. |
| void **act**() | Act pada metode world. Dipanggil sekali per tindakan. |
| void **addObject**(Actor object, int x, int y) | Tambahkan actor kepada world. |
| GreenfootImage**getBackground**() | Mengembalikan gambar world latar belakang. |

| | |
|---|---|
| int**getCellSize**() | Mengembalikan ukuran sel (dalam pixel). |
| Color **getColorAt**(int x, int y) | Mengembalikan warna di pusat sel. |
| int**getHeight**() | Mengembalikan tinggi dunia (dalamjumlahsel). |
| List **getObjects**(Class cls) | Mendapatkan semua benda di dunia. |
| List **getObjectsAt**(int x, int y, Class cls) | Mengembalikan semua benda di sel yang diberikan. |
| int**getWidth**() | Mengembalikan lebar world (dalam jumlah sel). |
| int**numberOfObjects**() | Mendapatkan jumlah actor saat ini di dunia. |
| void **removeObject**(Actor object) | Menghapus objek dari seluruh dunia. |
| void **removeObjects**(Collection objects) | Menghapus daftar bendadari seluruh dunia. |
| void **repaint**() | Mengecatulang dunia. |
| void **setActOrder**(Class... classes) | Mengatur urutan tindakan objek di dunia. |
| void **setBackground**(GreenfootImage image) | Mengatur gambar latar belakang untuk dunia. |
| void **setBackground**(String filename) | Mengatur gambar latar belakang untuk dunia dari file gambar. |
| void**setPaintOrder**(Class... classes) | Mengatur urutan cat objek di dunia. |
| void **started**() | Disebut oleh system Greenfoot ketika eksekusi telah dimulai. |
| void **stopped**() | Disebut oleh system Greenfoot ketika eksekusi telah berhenti. |
| **Class MouseInfo** | |

| | |
|---|---|
| Actor **getActor**() | Kembali aktor (jika ada) bahwa perilaku mouse saat ini berhubungan dengan. |
| int**getButton**() | Kembali pemeran (jika ada) bahwa perilaku mouse saat ini berhubungan dengan. |
| int**getClickCount**() | Jumlah mouse klik pada mouse ini. |
| int**getX**() | X posisi saat ini pada kursor mouse. |
| int**getY**() | Y posisi saat ini pada kursor mouse. |
| String**toString**() | Mengembalikan representasi string info acara mouse ini. |
| **Class Greenfoot** | |
| **Greenfoot**() | Konstruktor. |
| static void **delay**(int time) | Penundaan eksekusi oleh sejumlah langkah-langkah di waktu. Ukuran satu langkah waktu ditentukan oleh kecepatan slider. |
| static String **getKey**() | Mendapatkan kunci paling baru-baru ini ditekan sejak terakhir kali metode ini disebut. |
| static String **getMicLevel**() | Mendapatkan tingkat masukan mikrofon. |
| static MouseInfo**getMouseInfo**() | Kembali sebuah objek dengan informasi tentang keadaan mouse. |
| static int**getRandomNumber**(int limit) | Kembali nomor acak antara 0 (inklusif) dan limit (eksklusif). |
| static boolean**isKeyDown**(String keyName) | Periksa apakah diberikan kunci saat ini ditekan ke bawah. |
| static boolean**mouseClicked**(Object | Periksa apakah diberikan kunci saat ini |

| | |
|---|---|
| obj) | ditekan ke bawah. |
| static boolean**mouseDragEnded**(Object obj) | Benar jika mouse drag telah berakhir. |
| static boolean**mouseDragged**(Object obj) | True jika mouse telah menyeret pada objek tertentu. |
| static boolean**mouseMoved**(Object obj) | True jika mouse telah dipindahkan pada objek tertentu. |
| static boolean**mousePressed**(Object obj) | Benar mouse ditekan pada objek tertentu. |
| static void **playSound**(String soundFile) | Memutar suara dari file. |
| static void **setSpeed**(int speed) | Atur kecepatan pelaksanaan simulasi. |
| static void **setWorld**(int speed) | Menetapkan dunia ke orang yang diberikan. |
| static void **start**() | Menjalankan (atau melanjutkan) simulasi. |
| static void **stop**() | Menghentikan simulasi. |
| **Class GreenfootImage** | |
| **GreenfootImage**(GreenfootImage image) | Membuat GreenfootImage dari GreenfootImage lain. |
| **GreenfootImage**(int width, int height) | Membuat gambar (transparan) kosong dengan ukuran tertentu. |
| **GreenfootImage**(String filename) | Membuat gambar dari file gambar. |
| **GreenfootImage**(String string, int size,<br>Color foreground, Color background) | Membuat gambar dengan string tertentu diambil sebagai teks menggunakan ukuran font, warna latar depan dan warna latar belakang. |

| | |
|---|---|
| void **clear**() | Menghapus gambar. |
| void**drawImage**(GreenfootImage image,<br>int x, int y) | Menarik gambar yang diberikan ke gambar ini. |
| void **drawLine**(int x1, int y1, int x2, int y2) | Menggambar garis, menggunakan warna gambar saat ini, antara poin (x 1, y1) dan (x 2, y2). |
| void **drawOval**(int x, int y, int width, int height) | Menggambar oval dibatasi oleh persegi panjang tertentu dengan warna gambar saat ini. |
| void **drawPolygon**(int[] xPoints, int[] yPoints, intnPoints) | Menarik sebuah poligon tertutup yang didefinisikan oleh array x dan y koordinat. |
| void **drawRect**(int x, int y, int width, int height) | Menarik garis dari persegi panjang tertentu. |
| void **drawShape**(Shape shape) | Menggambar bentuk langsung pada gambar. |
| void **drawString**(String string, int x, int y) | Menggambar teks yang diberikan oleh string tertentu, menggunakan saat ini font dan warna. |
| void **fill**() | Mengisi seluruh gambar dengan warna gambar saat ini. |
| void **fillOval**(int x, int y, int width, int height) | Isi oval dibatasi oleh persegi panjang tertentu dengan warna gambar saat ini. |
| void **fillPolygon**(int[] xPoints, int[] yPoints, intnPoints) | Mengisi poligon tertutup yang didefinisikan oleh array x dan y koordinat. |
| void **fillRect**(int x, int y, int width, | Mengisi persegi panjang tertentu. |

| | |
|---|---|
| int height) | |
| void **fillShape**(Shape shape) | Menggambar bentuk diisi langsung pada gambar. |
| BufferedImage**getAwtImage**() | Kembali BufferedImage yang punggung GreenfootImage ini. |
| Color**getColor**() | Kembali warna gambar saat ini. |
| Color **getColorAt**(int x, int y) | Kembali warna di pixel tertentu. |
| Font **getFont**() | Mendapatkan font yang sekarang. |
| int**getHeight**() | Kembali tinggi gambar. |
| int**getTransparency()** | Kembali transparansi gambar (kisaran 0 hingga 255). |
| int**getWidth**() | Kembali lebar gambar. |
| void **mirrorHorizontally**() | Cermin gambar horizontal (Sandal di sekitar sumbu-x). |
| void **mirrorVertically**() | Cermin gambar vertikal (Sandal di sekitar sumbu y). |
| void **rotate**(int degrees) | Berputar gambar ini di sekitar pusat. |
| void **scale**(int width, int height) | Skala gambar ini untuk ukuran yang baru. |
| void **setColor**(Color color) | Mengatur warna gambar saat ini. |
| void **setColorAt**(int x, int y, Color color) | Menyetel warna di pixel diberikan dengan warna tertentu. |
| void **setFont**(Font f) | Menyetel font yang sekarang. |
| void **setTransparency**(int t) | Mengatur transparansi gambar (kisaran 0 hingga 255). |
| String**toString**() | Mengembalikan sebuah string representasi |

| | |
|---|---|
| | dari gambar ini. |
| **Class GreenfootSound** | |
| **GreenfootSound**(String filename) | Membuat suara baru dari file tertentu. |
| int**getVolume**() | Mendapatkan volume saat ini suara, antara 0 (off) dan 100 (paling keras). |
| boolean**isPlaying**() | True jika suara sedang diputar. |
| void **pause**() | Berhenti suara saat ini jika ia sedang bermain. |
| void **play**() | Mulai bermain suara ini. |
| void **playLoop**() | Memainkan suara ini berulang kali dalam satu lingkaran. |
| void **setVolume**(int level) | Mengatur volume saat ini suara antara 0 (off) dan 100 (paling keras). |
| void **stop**() | Berhenti bermain suara ini jika ia sedang bermain. |
| String **toString**() | Mengembalikan representasi string suara ini berisi nama file dan apakah itu saat ini bermain atau tidak. |

### II.7 *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.

Adobe 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 [12] (Blues Kim, Photodhop CS3 Accelerated).

## II.8 *Corel Draw*

*Corel Draw* merupakan salah satu aplikasi pengolah gambar berbasis *vector* yang banyak dipakai oleh pengguna PC. Karena berbagai kemudahan dan keunggulan yang dimiliki oleh *corel draw*, maka *corel draw* sering dimanfaatkan untuk *desktop publishing*, percetakan, dan bidang lain yang memerlukan pemrosesan visual.

Keunggulan mengolah gambar berbasis *vector* adalah ukuran hasil akhir yang dapat ditekan seminimal mungkin namun dengan kualitas yang tidak kalah dengan gambar berbasis *raster* atau *bitmap*.

Sebenarnya ada banyak sekali *software* desain grafis dan pengolah gambar berbasis *vector* yang bisa kita gunakan,misalnya saja diantaranya adalah *corel draw*, *photoshop*, *freehand*, *adobe ilustrator* dll. Namun *corel draw* merupakan program paling populer yang sering dimanfaatkan. Selain karena tampilannya yang *user friendly* dan mudah dipelajari, *corel draw* juga mempunyai berbagai keunggulan - keunggulan lain yang sangat bermanfaat.

## II.9 *Enterprise Architect*

Spark sistem mengkhususkan diri dalam kinerja tinggi dan alat-alat pemodelan visual scalable untuk perencanaan, desain dan konstruksi sistem perangkat lunak intensif.

Dengan pelanggan di industri mulai dari Teknik otomotif dan kedirgantaraan untuk keuangan, pertahanan, pemerintah, hiburan dan telekomunikasi, sistem Sparx adalah vendor terkemuka solusi inovatif berdasarkan Unified Modeling Language (UML) dan spesifikasinya terkait.

Anggota memberikan kontribusi dari objek manajemen grup (OMG), sistem Sparx berkomitmen untuk mewujudkan potensi pembangunan berbasis model yang didasarkan pada standar terbuka.

Produk unggulan dari perusahaan, arsitek Enterprise, telah menerima berbagai penghargaan sejak dirilis komersial pada Agustus 2000. Sekarang pada versi 10, arsitek Enterprise adalah alat desain pilihan untuk 300.000 pengguna terdaftar di seluruh dunia.

SparX arsitek Enterprise sistem yang digunakan oleh perusahaan-perusahaan dari semua ukuran dari organisasi multi-nasional besar, terkenal untuk perusahaan independen yang lebih kecil dan konsultan. Arsitek Enterprise juga digelar di berbagai industri termasuk: dirgantara, perbankan, pengembangan web, rekayasa, keuangan, Kedokteran, militer, penelitian, akademisi, transportasi, ritel, utilitas (gas, listrik dll), teknik elektro dan banyak lagi. Hal ini digunakan secara efektif untuk UML dan tujuan pelatihan arsitektur bisnis di banyak perguruan tinggi terkemuka dan Universitas di seluruh dunia [13] (http://www.sparxsystems.com/about.html).

### II.10 *Visual Paradigm*

Visual paradigma internasional terus-menerus berupaya memberdayakan organisasi untuk mengembangkan aplikasi berkualitas — lebih cepat, lebih baik dan lebih murah. Kami membantu pelanggan kami untuk memahami organisasi mereka dan menerjemahkan kebutuhan ke dalam perangkat lunak kualitas dengan alat yang kaya Terpilih dan mudah digunakan.

Seperti kata pepatah, a picture is bernilai seribu kata. Kita melihat visual pemodelan cara yang bagus untuk berkomunikasi secara efisien, khususnya selama masa krisis proyek. Itu membuat kolaborasi antara orang-orang terampil dalam berbagai bidang efektif dan bermanfaat. Jadi mengapa tidak membiarkan Deskripsi tekstual mengambil kursi belakang dan dukungan visual lebih ramah waktu untuk melakukan pekerjaan yang sama, jika tidak lebih baik?

Kita juga melihat bahwa ada model perangkat lunak seringkali kurang ideal. Mereka mungkin terlalu mahal. Atau mereka tidak memiliki kemudahan penggunaan dan memerlukan investasi substansial dalam pelatihan. Jadi kami

yakin ada pengguna yang dapat memilih untuk pemodelan visual dalam Roh tetapi belum tentu dalam praktek. Ini adalah di mana alat-alat kami datang untuk mengisi ceruk ini dengan hemat inovasi.

Kami memiliki keyakinan yang tanpa kompromi — perangkat lunak harus user-friendly, bahkan ketika hal ini dimaksudkan untuk memecahkan masalah yang cukup kompleks. Kita menaatinya dan semua produk kami dibangun di sekitar ide ini [14] (http://www.visual-paradigm.com/aboutus).