APLIKASI PERMAINAN ASTEROID SHOOTER MENGGUNAKAN MCRNG DAN A* SEBAGAI ALGORITMA *RANDOMING SPAWN* DAN PENCARIAN OBJEK BERBASIS MOBILE

SKRIPSI

Oleh:
JUNIARDI NUR FADILA
NIM. 10650008

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2014

APLIKASI PERMAINAN ASTEROID SHOOTER MENGGUNAKAN MCRNG DAN A* SEBAGAI ALGORITMA *RANDOMING SPAWN* DAN PENCARIAN OBJEK BERBASIS MOBILE

SKRIPSI

Diajukan kepada:
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:
JUNIARDI NUR FADILA
NIM. 10650008

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2014

HALAMAN PERSETUJUAN

APLIKASI PERMAINAN ASTEROID SHOOTER MENGGUNAKAN MCRNG DAN A* SEBAGAI ALGORITMA *RANDOMING SPAWN* DAN PENCARIAN OBJEK BERBASIS MOBILE

SKRIPSI

Oleh :

Nama : Juniardi Nur Fadila
NIM : 10650008
Jurusan : Teknik Informatika
Fakultas : Sains Dan Teknologi

Telah Disetujui, 7 April 2014

Dosen Pembimbing I	Dosen Pembimbing II
Hani Nurhayati, M.T	**Dr. Muhammad Faisal, M.T**
NIP. 19780625 200801 2 006	**NIP. 19740510 200501 1 007**

Mengetahui,

Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian

NIP. 19740424 200901 1 008

HALAMAN PENGESAHAN

APLIKASI PERMAINAN ASTEROID SHOOTER MENGGUNAKAN MCRNG DAN A* SEBAGAI ALGORITMA *RANDOMING SPAWN* DAN PENCARIAN OBJEK BERBASIS MOBILE

SKRIPSI

Oleh :

Juniardi Nur Fadila
NIM. 10650008

Telah Dipertahankan Di Depan Dewan Penguji Skripsi
Dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)

Tanggal 11 April 2014

Susunan Dewan Penguji:		Tanda Tangan
1. Penguji Utama	: Fressy Nugroho, M.T NIP. 19710722 201101 1 001	( )
2. Ketua Penguji	: Yunifa Miftachul Arif, M.T NIP. 19830616 201101 1 004	( )
3. Sekretaris	: Hani Nurhayati, M.T NIP. 19780625 200801 2 006	( )
4. Anggota Penguji	: Dr. M. Faisal, M.T NIP. 19740510 200501 1 007	( )

Mengetahui,
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian

NIP. 19740424 200901 1 008

HALAMAN PERNYATAAN

ORISINALITAS PENELITIAN

Saya yang bertandatangan di bawah ini:

Nama	:	Juniardi Nur Fadila
NIM	:	10650008
Fakultas/Jurusan	:	Sains Dan Teknologi / Teknik Informatika
Judul Penelitian	:	Aplikasi permainan asteroid shooter menggunakan mcrng dan A* sebagai algoritma *randoming spawn* dan pencarian objek berbasis *mobile*

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan atau pikiran oarang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka. Apabila di kemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut

Malang, 17 Maret 2014

Yang Membuat Pernyataan,

Juniardi Nur Fadila

10650008

HALAMAN MOTTO

"Space that has no boundaries, so does science. Don't ever give up studying because of the knowledge that we have is as small as a grain of sand compared to the existing knowledge in the universe"

HALAMAN PERSEMBAHAN

Dengan rasa syukur seraya mengharap ridho Ilahi
kupersembahkan karya ini kepada :
Ayahanda dan Ibunda tercinta
Turmudi** dan **Rosalia Siswanti
Atas Segalanya.

*Kepada teman seperjuanganku **Puspa Safitri, Muiz Lidinillah, Novi Anto, Nazzala Tia, Rizal Furqon, Rifa'ati Azizah Munaz, Catur Prio Wibowo** yang selalu saling bersama-sama dan saling mengingatkan jika lalai*

*Kepada sahabatku yang telah berpulang terlebih dahulu, **Alm. Muslih Adlan Agung** dan **Alm. Wicaksono Tegar K.***

Kepada para sahabat TI angkatan 2010, yang selalu ada untuk membantu sesama.

Dan kepada teman-temanku semua yang tidak bisa kusebutkan satu persatu yang selalu membantuku dan menyemangatiku di saat aku susah dan terpuruk.

Semoga Allah SWT melindungi, menyayangi dan menempatkan mereka semuanya pada surganya kelak dan melimpahkan rezeki kepada mereka semua...

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah melimpahkan rahmat serta karuniaNya kepada penulis sehingga bisa menyelesaikan skripsi dengan judul "Aplikasi permainan *Asteroid Shooter* menggunakan MCRNG dan A* sebagai algoritma *randoming spawn* dan pencarian objek berbasis *mobile* " dengan baik.

Shalawat serta salam semoga tercurah kepada Nabi Agung Muhammad SAW yang telah membimbing umatnya dari gelapnya kekufuran menuju cahaya Islam yang terang benderang.

Penulis menyadari keterbatasan pengetahuan yang penulis miliki, karena itu tanpa keterlibatan dan sumbangsih dari berbagai pihak, sulit bagi penulis untuk menyelesaikan skripsi ini. Maka dari itu dengan segenap kerendahan hati patutlah penulis ucapkan terima kasih kepada:

1. Ayah dan Ibu yang selalu memotivasi dan selalu mendoaakan yang terbaik dalam pengerjaan skripsi ini.
2. Hani Nurhayati, M.T, selaku dosen pembimbing I yang telah meluangkan waktu untuk membimbing, memotivasi, mengarahkan dan memberi masukan dalam pengerjaan skripsi ini.
3. Muhammad Faisal, M.Kom, selaku dosen pembimbing II, yang selalu memberikan masukan, nasehat serta petunjuk dalam penyusunan laporan skripsi ini.
4. Cahyo Crysdian, M.Kom selaku Ketua Jurusan Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang, yang mendukung dan mengarahkan dalam pengerjaan skripsi ini.

5. Segenap Dosen Teknik informatika yang telah memberikan bimbingan keilmuan kepada penulis selama masa studi.
6. Semua pihak yang tidak mungkin penulis sebutkan satu-persatu, atas segala yang telah diberikan kepada penulis dan dapat menjadi pelajaran.

Sebagai penutup, penulis menyadari dalam skripsi ini masih banyak kekurangan dan jauh dari sempurna. Semoga apa yang menjadi kekurangan bisa disempurnakan oleh peneliti selanjutnya. Apa yang menjadi harapan penulis, semoga karya ini bermanfaat bagi kita semua. Amin.

Malang, 07 April 2014

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Nur Fadila, Juniardi. 2014. **Aplikasi Permainan "Asteroid Shooter" Menggunakan MCRNG dan A* Sebagai Algoritma *Randoming Spawn* Dan Pencarian Objek Berbasis Mobile**. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing : (I) Hani Nurhayati, M.T (II) Dr. Muhammad Faisal, M.T

Kata Kunci: *Acak, Pembangkit,Multiplicative Congruential RNG(MCRNG), Path Finding, AStar (A*).*

Permainan *Asteroid Shooter* ini adalah sebuah permainan yang menganut tipe permainan *shoot them all* yaitu adalah sebuah game yang hanya diharuskan menembak semua musuh. Musuh yang berdatangan diharapkan selalu acak untuk kemunculannya dan kriterianya. Untuk menembak musuh sang pemain harus menjawab pertanyaan pada musuh tersebut. Jika benar sebuah rudal atau peluru akan muncul dan mengejar target tersebut. Namun bagaimana kita membangkitkan musuh tersebut secara acak terus menerus? Dan bagaimana rudal yang keluar tersebut dapat mencari targetnya? Saat ini terdapat beberapa metode pengacakan dan path finding. Salah satunya adalah MCRNG dan Astar(A*). Pada penelitian ini, kita akan mencoba menerapkan MCRNG sebagai metode pembangkit musuh agar memiliki kriteria yang acak dan algoritma AStar(A*) untuk kecerdasan rudal agar dapat mengejar targetnya. Pengujian dilakukan pada perangkat mobile dengan menggunakan *platform* Android.

ABSTRACT

Nur Fadila, Juniardi. 2014. **Asteroid Shooter Games Application with MCRNG and AStar (A*) as a Randoming Spawn and Object Search Algorithm Based on Mobile Device.** Thesis. Informatics Department of Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University, Malang, Adviser: (I) Hani Nurhayati, M.T (II) Dr. Muhammad Faisal, M.T

This Asteroid Shooter is a shoot them all game type which is the game that must shoot all of opponents spawned in front of player. Enemies that expected to be spawn is an enemy that always spawn from random location, and have a random criteria too. To shoot the enemy, the player must answer the question on the clicked enemy. If the answer is true, then a missile will appear and chase the clicked enemy. But, how do we generate the random location for the enemy continuously? And what about the missile that can seek out the clicked enemy? Currently there are several algorithms of randomization and path finding. One is Multiplicative Congruential Random Number Generator(MCRNG) and AStar(A*) Path Finding. In this study, we will try to apply this MCRNG algorithm for a method to generating a random enemy spawn location and enemy criteria and AStar(A*) algorithm for missile artificial intelligence to chase the clicked enemy. Tests carried out on mobile devices using the Android platform

Key Word: Random, Generator, Multiplicative Congruential RNG, Path Finding, AStar(A*).

BAB I

PENDAHULUAN

1.1 Latar Belakang

Sebuah game dalam masa kini sering dipandang sebelah mata oleh orang tua. Mengapa demikian? Hal ini karena game saat ini dianggap kurang memberikan kontribusi untuk pembelajaran anak-anak. Bermain game oleh orang tua dipandang sebagai pekerjaan yang tidak bermanfaat dan hanya membuang-buang uang saja. Berdasarkan pengalaman penulis, bermain game memang dapat membuat seseorang ketagihan dan rela membuang – buang uang untuk hal itu. Kita ambil contoh saja untuk seorang gamer yang sudah ketagihan sebuah game online. Gamer tersebut rela menghabiskan uangnya untuk berada seharian di depan komputer di sebuah warnet di sudut kota. Jika kita ambil rata-rata untuk sebuah harga paket internet di warnet tersebut seharga Rp10.000 untuk bermain selama 3 jam, dan gamer tersebut menghabiskan rata-rata waktunya di warnet tersebut selama 6 jam, maka untuk tiap harinya gamer tersebut selalu menghabis kan Rp 20.000 untuk bermain game. Untuk setiap minggunya sebesar Rp 140.000 dan setiap bulannya total gamer tersebut menghabiskan Rp 560.000. Harga tersebut hanya untuk bermain di warnet saja, belum lagi jika gamer tersebut membeli voucher untuk bermain game tersebut. Jika sudah benar-benar ketagihan, seorang gamer tersebut rela menghabiskan uang jutaan rupiah hanya untuk membeli sebuah ID pemain.

Betapa besar pengaruh game terhadap kehidupan seseorang. Bagi penulis pengaruh game tersebut ingin di adopsi untuk media pembelajaran. Pembelajaran yang ingin disampaikan oleh penulis adalah sebuah pembelajaran matematika. Matematika adalah pelajaran mendasar yang harus dipelajari mengingat matematika ialah induk dari segala materi perhitungan yang ada. Selain itu, menurut ilmuan galileo dalam salah satu ungkapannya menyatakan "Matematika adalah bahasa Tuhan ketika Dia menulis Alam Semesta". Allah mengenkripsikan Matematika dalam Al-quran, untuk memelihara komitmen isi dan bacaan serta kandungan yang ada didalamnya.

Di dalam QS. Jin ayat 28 dijelaskan:

لِّيَعۡلَمَ أَن قَدۡ أَبۡلَغُواْ رِسَٰلَٰتِ رَبِّهِمۡ وَأَحَاطَ بِمَا لَدَيۡهِمۡ وَأَحۡصَىٰ كُلَّ شَيۡءٍ عَدَدَۢا ٢٨

"Supaya Dia mengetahui, bahwa Sesungguhnya Rasul-rasul itu telah menyampaikan risalah-risalah Tuhannya, sedang (sebenarnya) ilmu-Nya meliputi apa yang ada pada mereka, dan ***Dia menghitung*** *segala sesuatu satu persatu."*

Juga pada QS. Maryam ayat 93 – 94:

إِن كُلُّ مَن فِي ٱلسَّمَٰوَٰتِ وَٱلۡأَرۡضِ إِلَّآ ءَاتِي ٱلرَّحۡمَٰنِ عَبۡدٗا ٩٣ لَّقَدۡ أَحۡصَىٰهُمۡ وَعَدَّهُمۡ عَدّٗا ٩٤

"Tidak ada seorangpun di langit dan di bumi, kecuali akan datang kepada Tuhan yang Maha Pemurah selaku seorang hamba. Sesungguhnya Allah telah ***menentukan jumlah mereka dan menghitung mereka dengan hitungan yang teliti****."*

Ayat-ayat tersebut menjelaskan bahwa Allah juga menciptakan segala sesuatu dengan perhitungan-perhitungan yang Maha teliti. Maka dari itu kita diharuskan mempelajari matematika semenjak usia dini. Akan tetapi kenyataannya, keberadaan matematika ini

dianggap sebagai momok dalam pelajaran dikarenakan persepsi bahwa matematika itu susah dan membingungkan. Dari pendapat beberapa siswa SD di kampung penulis, pelajaran yang paling susah bagi mereka adalah matematika. Padahal jika dari usia dini kita tidak mahir matematika, maka untuk kedepannya kita akan sulit melangkah. Hal tersebut dikarenakan dalam pelajaran matematika yang ada hanyalah berhitung, berhitung, dan berhitung sehingga hal tersebut dapat memberi kesan bahwa matematika itu hanyalah memeras otak saja dan tidak memiliki kesan menyenangkan bagi anak-anak. Sehingga anak-anak cenderung memiliki persepsi bahwa matematika itu membosankan.

Hal senada diungkap oleh seorang ahli pendidikan barat yang menyatakan bahwa ada rasa takut akan matematika, rasa takut tersebut mendekam dalam pikiran (Buxton, 1984:1). Masih menurut Buxton, rasa takut ini terjadi dikarenakan adanya *Mind in Chaos* (Buxton, 1984:85), yaitu suatu kesan negatif yang dibiarkan terjadi sejak mereka masih kecil bahwa matematika itu sulit yang pada akhirnya menjadikan mereka sampai dewasa berfikiran bahwa matematika sulit dan menakutkan. Kemudian ada juga yang beranggapan bahwa matematika itu terasa sulit dan menakutkan karena di pengaruhi beberapa faktor internal dan eksternal.

Faktor internal diantaranya minat siswa terhadap mata pelajaran Matematika. Matematika memang memiliki sesuatu yang berbeda dengan mata pelajaran yang lain. Secara normal, ketika kita belajar matematika maka secara otomatis banyak sekali berinteraksi dengan angka-angka dibanding dengan kata. Nah, bagi anak yang tidak memiliki kecerdasan logic-mathematics hal ini merupakan sesuatu yang sangat membosankan sehingga mengurangi minat mereka untuk belajar. Sehingga dibutuhkan cara yang tepat supaya mereka tertarik untuk belajar terutama bidang studi matematika.

Kemudian faktor eksternal yang mempengaruhi sulitnya belajar matematika ialah guru yang kurang bersahabat. Guru yang kurang bersahabat dan dianggap siswa

merupakan guru yang *killer* dapat menjadi faktor yang membuat siswa tersebut takut untuk belajar matematika. Selain itu juga metode penyampaian yang kurang tepat juga bisa menjadi hal yang mempengaruhi tingkat pembelajaran siswa tersebut, karena setiap orang memiliki gaya belajar yang berbeda-beda.

Sebuah ganjaran atau penguatan mempunyai peranan yang sangat penting dalam proses belajar. Ganjaran merupakan respon yang sifatnya menggembirakan dan merupakan tingkah laku yang sifatnya subjektif. Penguatan merupakan sesuatu yang mengakibatkan meningkatnya kemungkinan suatu respon dan lebih mengarah kepada hal-hal yang sifatnya dapat diamati dan diukur (Skinner, 1938). Dalam teori Skinner dinyatakan bahwa penguatan terdiri atas penguatan positif dan penguatan negatif. Contoh penguatan positif diantaranya adalah pujian yang diberikan pada anak setelah berhasil menyelesaikan tugas dan sikap guru yang bergembira pada saat anak menjawab pertanyaan. Skiner menambahkan bahwa jika respon siswa baik (menunjang efektivitas pencapaian tujuan) harus segera diberi penguatan positif agar respon tersebut lebih baik lagi,atau minimalnya perbuatan baik itu dipertahankan.

Dalam pembelajaran sebaiknya dikembangkan suatu proses pembelajaran yang menarik sehingga bisa meningkatkan minat siswa terhadap pelajaran matematika. Salah satunya ialah pembelajaran menggunakan game sebagai media pembelajarannya (Dienes, 2003)

Dari uraian diatas, menggagas penulis membuat sebuah program permainan yang mengaplikasikan beberapa teori diatas. Permainan ini dibuat agar dapat menjadi tutor bagi usernya untuk dapat berhitung matematika dengan lancar. Permainan ini juga mengaplikasikan teori Skinner, yang akan memberikan pujian atau ganjaran pada usernya saat mendapatkan hasil yang memuaskan. Permainan ini diharapkan akan menjadi solusi yang menarik minat siswa untuk belajar matematika.

Permainan yang saat ini banyak dikembangkan ialah permainan-permainan dengan *console* berupa *smartphone*. Keunggulan dari permainan berbasis *smartphone* ini ialah dapat dimainkan kapan saja dan dimana saja. Sistem operasi yang digunakan ialah android yang mana android ini adalah sistem operasi yang bersifat terbuka dan para *developer* dapat berkesempatan mengembangkan aplikasi-aplikasinya sesuai keinginannya sendiri. Permainan yang digemari oleh anak-anak saat ini ialah sebuah game ketangkasan dimana setiap bertambah level, bertambah sulit pula level tersebut diselesaikan. Pada penelitian ini, penulis ingin mencoba membuat permainan bernama *Asteroid Shooter.*

Permainan *Asteroid Shooter* ini merupakan sebuah permainan bergenre *Shoot Them 'All* akan tetapi di modifikasi menjadi sebuah game edukasi untuk pembelajaran matematika . Ide dari game ini ialah dimana terdapat sebuah pesawat luar angkasa yang berada di tengah layar dan diharuskan menembak setiap *asteroid* yang akan menabrak pesawat tersebut. Cara menembak asteroid tersebut ialah dengan menjabawab pertanyaan matematika yang ada di setiap *asteroid* yang datang. Kemunculan *asteroid* ini dari posisi yang acak sehingga tidak bisa ditebak darimana *asteroid* ini akan datang. Posisi acak ini didapatkan dari hasil algoritma *Multiplicative Congruential Random Number Generator* (MCRNG).

MCRNG ini digunakan sebagai pengganti fungsi acak pada bawaan bahasa pemrograman yaitu *Math.random* dikarenakan pada fungsi acak bawaan bahasa pemrograman tersebut dinilai kurang optimal oleh penulis. Karena saat penulis mencoba me-*looping* data yang banyak, terdapat perbedaan waktu yang cukup signifikan untuk perulangan menggunakan *Math.random* dibandingkan menggunakan rumus MCRNG

Dalam permainan, *asteroid* bergerak mendatar menuju pesawat pemain. Jika pemain menembak, maka akan muncul peluru. Peluru ini secara otomatis akan mengejar asteroid

yang menjadi sasaran. Bagaimana peluru ini bisa otomatis mencari sasaran? Maka peluru tersebut harus memiliki kecerdasan untuk mencari jalur agar sampai menuju sasaran. Pencarian jalur oleh peluru inilah yang akan mengaplikasikan algoritma A*. Algoritma A* digunakan karena mudah dalam penggunaannya dan memiliki hasil pencarian yang optimal.

Aplikasi permainan ini akan diimplementasikan pada *Android OS mobile* dengan harapan dapat diaplikasikan dimana saja dan kapan saja.

1.2 Rumusan Masalah

Dapatkah kita menerapkan algoritma pengacakan MCRNG dan algoritma *path finding* A-Star (A*) dalam sebuah permainan untuk pembelajaran matematika anak-anak ?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini sebagai berikut:.

a. Permainan ini dimainkan *single player*.
b. Pertanyaan berupa perhitungan matematika sederhana (hanya pertambahan, perkurangan, dan perkalian)
c. *Multiplicative CRNG* digunakan sebagai algoritma pengacakan kemunculan *asteroid.*
d. Algoritma *A** digunakan sebagai algoritma pencarian
e. Permainan ini diimplementasikan pada *Android OS mobile*.

1.4 Tujuan Penelitian

Dapat menerapkan algoritma pengacakan MCRNG dan A* dalam sebuah aplikasi permainan untuk pembelajaran matematika anak-anak.

1.5 Manfaat Penelitian

Manfaat pembuatan aplikasi permainan ini adalah memberikan metode pembelajaran hitung-menghitung yang menyenangkan untuk di pelajari bagi anak-anak usia dini.

BAB II

LANDASAN TEORI

1.1 Kajian Pustaka

1.1.1 *Game* (Permainan)

Game merupakan kata dalam bahasa inggris yang berarti permainan. Permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius atau dengan tujuan refreshing. Suatu cara belajar yang digunakan dalam menganalisa interaksi antara sejumlah pemain maupun perorangan yang menunjukkan strategi-strategi yang rasional. Teori permainan pertama kali ditemukan oleh sekelompok ahli Matematika pada tahun 1944. Permainan terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan. Peraturan-peraturan menentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi. (Neumann, 2007)

Pengertian game menurut beberapa ahli :

1. Menurut Agustinus Nilwan dalam bukunya Pemrograman Animasi dan Game Profesional terbitan Elex Media Komputindo, game merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Jika ingin mendalami pengunaan animasi haruslah memahami pembuatan

2. game. Atau jika ingin membuat game, maka haruslah memahami teknik dan metode animasi, sebab keduanya saling berkaitan.
3. Menurut Clark C. Abt, Game adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan dibatasi oleh konteks tertentu (misalnya, dibatasi oleh peraturan).
4. Menurut Bernard Suits Game adalah upaya sukarela untuk mengatasi rintangan yang tidak perlu .
5. Menurut Greg Costikyan, Game adalah sebentuk karya seni di mana peserta, yang disebut Pemain, membuat keputusan untuk mengelola sumberdaya yang dimilikinya melalui benda di dalam game demi mencapai tujuan .

Game ditinjau dari genre permainannya dapat dibagi menjadi beberapa pengklarifikasian. Diantaranya ialah :

1. *Shooting Game*, game ini merupakan sebuah simulasi pertarungan menggunakan senjata api atau sejenisnya, dimana inti dari permainan ini adalah menembak musuh atau sasaran. *Game* ini memerlukan *reflect*, koordinasi mata dan tangan, serta timing untuk menembak juga. Termasuk didalam genre ini yaitu permainan bertipe:
    - *First person shooting(FPS)* contohnya *counter-strike, call of duty*
    - *Drive n' shoot* meanfaatkan unsur simulasi kendaraan dalam gameplay-nya seperti permainan *Spy Hunter, Rock and Roll Racing, Burnout.*
    - *Shoot em' up* contohnya Raiden, 1942, dan Gradius

- *Light gun shooting* tipe ini menggunakan alat yang umumnya berbentuk seperti senjata. Contohnya *Virtual Cop* dan *Time Crisis*

2. *Fighting Game*, merupakan game pertarungan antara dua karakter atau lebih. Biasanya pada tipe ini, pemain diuji kecepatan refleksnya dan koordinasi mata dengan tangan. Tipe ini memfokuskan pemain untuk belajar penguasaan jurus dan cara mengeksekusinya. Selain itu *combo*-pun menjadi esensial untuk mengalahkan lawan secepat mungkin. Contohnya *Street Fighter, Teken, Mortal Combat*, dan lain sebagainya
3. *Adventure Game.* Genre ini merupakan sebuah cerita atau dongeng atau film yang diadopsi menjadi sebuah game. Pada tipe ini, game memiliki sebuah alur cerita dimana jika jalan ceritanya telah usai, maka game tersebut juga tamat. Pemain ditekankan untuk memahami jalan cerita dan kemampuan berpikirnya dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter. Termasuk dalam genre ini yaitu:
   - Petualangan dengan teks contohnya Kings Quest, Space Quest, Heroes Quest, dan lain sebagainya
   - Novel atau Film interaktif seperti game *dating* yang banyak beredar di jepang, Dragons Lair dan Night Trap.
4. *Action-Adventure Game.* Game jenis ini sudah berkembang menjadi genre campuran antara *action, beat'em up,* juga *shooting game* dan sekarang jenis ini cenderung memiliki visual 3D dan sudut pandang orang ke-3. Contohnya *Tomb Rider, Prince of Persia, Grand Theft Auto*

5. Simulasi, Konstruksi dan manajemen. Video Game jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. Dari mencari jodoh dan pekerjaan, membangun rumah, gedung hingga kota, mengatur pajak dan dana kota hingga keputusan memecat atau menambah karyawan. Dunia kehidupan rumah tangga sampai bisnis membangun konglomerasi, dari jualan limun pinggir jalan hingga membangun laboratorium cloning. Video Game jenis ini membuat pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas. Contoh: Sim City, The Sims, Tamagotchi.
6. *Role Playing Game*. Video game jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh/peran perwakilan pemain di dalam permainan, yang biasanya adalah tokoh utamanya, dimana seiring kita memainkannya, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain (biasanya menjadi semakin hebat, semakin kuat, semakin berpengaruh, dll) dalam berbagai parameter yang biasanya ditentukan dengan naiknya level, baik dari status kepintaran, kecepatan dan kekuatan karakter, senjata yang semakin sakti, ataupun jumlah teman maupun mahluk peliharaan.Secara kebudayaan, pengembang game Jepang biasanya membuat *Role Playing Game (RPG)* ke arah cerita linear yang diarahkan seolah karakter kita adalah tokoh dalam cerita itu, seperti *Final Fantasy*, *Dragon Quest* dan *Xenogears*. Sedangkan pengembang game RPG Eropa, cenderung membuat karakter kita bebas memilih jalan cerita

sendiri secara non-linear, seperti *Ultima, Never Winter Nights, baldurs gate, Elder Scroll, dan Fallout.*

7. Strategi. Kebalikan dari video game jenis action yang berjalan cepat dan perlu refleks secepat kilat, video game jenis strategi, layaknya bermain catur, justru lebih memerlukan keahlian berpikir dan memutuskan setiap gerakan secara hati-hati dan terencana. Video game strategi biasanya memberikan pemain atas kendali tidak hanya satu orang tapi minimal sekelompok orang dengan berbagai jenis tipe kemampuan, sampai kendaraan, bahkan hingga pembangunan berbagai bangunan, pabrik dan pusat pelatihan tempur, tergantung dari tema ceritanya. Pemain game strategi melihat dari sudut pandang lebih meluas dan lebih kedepan dengan waktu permainan yang biasanya lebih lama dan santai dibandingkan game action. Unsur-unsur permainannya biasanya berkisar sekitar, prioritas pembangunan, peletakan pasukan, mencari dan memanfaatkan sumberdaya (uang, besi, kayu,minyak,dll), hingga ke pembelian dan peng-upgrade-an pasukan atau teknologi. Game jenis ini terbagi atas:
    - *Real time Strategy,* game berjalan dalam waktu sebenarnya dan serentak antara semua pihak dan pemain harus memutuskan setiap langkah yang diambil saat itu juga berbarengan mungkin saat itu pihak lawan juga sedang mengeksekusi strateginya. Contoh: *Starcraft, Warcraft , dan Command and Conquer.*
    - *Turn based Strategy ,* game yang berjalan secara bergiliran, saat kita mengambil keputusan dan menggerakan pasukan, saat itu pihak lawan menunggu, begitu pula sebaliknya, layaknya catur. contoh: *Front*

Mission, Super robot wars, Final Fantasy tactics, Heroes of might and magic, Master of Orion.

8. *Puzzle*. Video game jenis ini sesuai namanya berintikan mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kota masuk ke tempat yang seharusnya, itu semua termasuk dalam jenis ini. Sering pula permainan jenis ini adalah juga unsur permainan dalam video game petualangan maupun game edukasi. *Tetris, Minesweeper, Bejeweled, Sokoban dan Bomberman*.
9. Simulasi kendaraan. Video Game jenis ini memberikan pengalaman atau interaktifitas sedekat mungkin dengan kendaraan yang aslinya, muskipun terkadang kendaraan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detil dan pengalaman realistik menggunakan kendaraan tersebut. Terbagi atas beberapa jenis:
    - Perang. Video game simulasi kendaraan yang sempat tenar di tahun 90-an ini mengajak pemain untuk menaiki kendaraan dan berperang melawan kendaraan lainnya. Dan kebanyakan diantaranya memiliki judul sama dengan nama kendaraannya. Contoh : *Apache 64, Comanche, Abrams, YF-23, F-16 fighting eagle*.
    - Balapan. Dari namanya sudah jelas, siapa sampai duluan di garis finish dialah pemenangnya! Terkadang malah pemain dapat memilih kendaraan, mendandani, upgrade mesin bahkan mengecatnya. Contoh: *Top Gear, Test Drive, Sega Rally Championship, Daytona, Grand Turismo, Need For Speed, Mario Cart, ManXTT*.

- Luar Angkasa. Walau masih dapat dikategorikan simulasi kendaraan perang, tetapi segala unsur fiksi ilmiah dan banyaknya judul yang beredar membuat subgenre ini pantas dikategorikan diluar simulasi kendaraan perang. Jenis ini memungkinkan pemain untuk menjelajah luar angkasa, berperang dengan mahluk alien, mendarat di planet antah berantah atau sekedar ingin merasakan bagaimana menjadi kapten di film fiksi ilmiah kesayangan kamu. Contoh: *Wing Commander, Freelancer , Star Wars X-Wing, Star Wars Tie Fighter, dll.*
- *Mecha.* Pendapat bahwa hampir tidak ada orang yang terekspos oleh film robot jepang saat kecilnya tidak memimpikan ingin mengendalikan robot, memang sulit dibantah. Dipopulerkan oleh serial *Mechwarrior* oleh *Activision,* subgenre Simulasi Mecha ini memungkinkan pemainnya untuk mengendalikan robot dan menggunakannya untuk menghancurkan gedung, helikopter dan tentu saja robot lainnya. Contoh: *Mechwarrior, Gundam Last war Chronicles, dan Armored Core.*

10. *Olahraga.* Singkat padat jelas, bermain sport di PC atau konsol anda. Biasanya permainannya diusahakan serealistik mungkin walau kadang ada yang menambah unsur fiksi seperti NBA JAM. Contohnya pun jelas, Seri *Winning Eleven*, seri NBA, seri FIFA, *John Madden* NFL, *Lakers vs Celtics*, *Tony hawk pro skater*, dll.

1.1.2 *MCRNG (Multiplicative Congruential Random Number Generator)*

Bilangan acak adalah suatu angka yang muncul tanpa bisa ditebak berapa nilai yang akan muncul. Biasanya bilangan acak ini digunakan untuk undian atau pun untuk enkripsi suatu password atau sejenisnya. Atau dengan kata lain bilangan acak adalah bilangan yang tidak dapat diprediksi kemunculannya. Pada zaman dahulu bilangan acak diperoleh dengan cara melempar dadu atau mengocok kartu. Pada zaman modern bilangan acak diperoleh dengan cara membentuk bilangan acak secara numerik/ aritmatik (menggunakan komputer), disebut "*Pseudo Random Number*" (bilangan pseudo acak).

Pembangkit bilangan acak harus (Riani L, 2010):

1. Berdistribusi uniform (0,1) dan tidak berkorelasi antar bilangan
2. Membangkitkan secara cepat dan *storage* tidak besar
3. Dapat di-"*reproduce*"
4. Periode besar, karena kemungkinan bilangan acak dibangkitkan berulang

Menurut Riani L. (2010:3), tidak ada komputasi yang benar-benar menghasilkan deret bilangan acak secara sempurna. Bilangan acak yang dibangkitkan oleh computer adalah bilangan acak semu (*Pseudo Random Number*), karena menggunakan rumus-rumus matematika. *Random Number Generator* adalah suatu algoritma yang digunakan untuk menghasilkan urutan-urutan atau *sequence* dari angka-angka sebagai hasil dari perhitungan dengan komputer yang diketahui distribusinya sehingga angka-angka tersebut muncul secara random dan digunakan terus-menerus.

- *Sequence* atau Urutan:

Yang dimaksud dengan sequence di sini adalah bahwa random number tersebut harus dapat dihasilkan secara urut dalam jumlah yang mengikuti algoritma tertentu dan sesuai dengan distribusi yang akan terjadi atau dikehendaki.

- *Distribution* atau distribusi

  Pengertian distribusi berhubungan dengan distribusi probabilitas yang dipergunakan untuk meninjau atau terlibat langsung dalam penarikan random number tersebut. Pada umumnya distribusi probabilitas untuk random number ini adalah *Uniform Variate* yang dikenal dengan Distribusi Uniform. Maksudnya adalah rentang dari nilai

- Muncul angka-angka secara random

  Pengertian random disini menunjukkan bahwa algoritma tersebut akan menghasil-kan suatu angka yang akan berperan dalam pemunculan angka yang akan keluar dalam proses komputer. Dengan kata lain suatu angka yang diperoleh merupakan angka penentu bagi angka random berikutnya. Demikian seterusnya. Walaupun random number ini saling berkaitan namun angka-angka yang muncul dapat berlainan.

  (MDP, 2013)

Banyak algoritma atau metode yang dapat digunakan untuk membangkitkan bilangan acak. Pada umumnya terdapat beberapa sumber yang dipergunakan, antara lain (Hidayat, 2010):

a. Metode Fisik

Metode paling pertama untuk menghasilkan angka secara acak adalah dengan menggunakan dadu, koin, rolet, dan lain sebagainya. Sampai saat ini, metode ini masih cukup sering digunakan, terutama di dalam game dan perjudian. Karena metode ini dianggap terlalu lambat, pengaplikasiannya untuk statistika dan kriptografi kurang begitu popular saat ini. Dasar dari metode fisik adalah fenomena fisika atomik atau subatomik acak yang tidak bisa diprediksi dapat dilacak dengan menggunakan mekanika kuantum.

b. Metode Distribusi Probabilitas

Metode ini menggunakan fungsi densitas probabilitas. Metode ini bekerja cukup baik untuk menghasilkan pseudo-random dan true random number. Salah satu metode, yaitu metode inverse, mengintegralkan area lebih dari sama dengan bilangan acak. Metode kedua, acceptance-rejection, memilih antara nilai x dan y, lalu membandingkan apakah fungsi x lebih besar dari nilai y. Apabila fungsi x lebih dari nilai y, maka nilai x akan diterima. Jika sebaliknya, maka nilai x akan ditolak dan algoritmanya akan mencoba ulang.

c. Congruential Pseudo Random Number Generator, Random Number Generator

Metode ini menggunakan algoritma bernama Pseudo-random number generator yang secara otomatis menghasilkan serangkaian angka acak yang memiliki kualitas baik. Nilai yang dihasilkan oleh algoritma tersebut secara umum ditentukan dengan sebuah konstanta yang disebut seed. Salah satu *Pseudo Random Number Generator* yang umum adalah

linear congruential generator. Algoritma ini dikemukakan oleh D. H. Lehmer pada tahun 1949. Algoritma *Linear Congruential Generator* ditentukan oleh 4 bilangan bulat antara lain (Charles N. Zeeb, 1984):

m	*modulus*	$m > 0$
a	faktor pengali	$0 \leq a < m$
c	*increment*	$0 \leq c < m$
Z_0	angka permulaan (*seed*)	$0 \leq Z_0$

Pseudo RNG Linear Congruential Generator, berbentuk:

$$Z_i = (aZ_{i-1} + c) \bmod m$$

$$R_i = Z_i / m$$

Dimana:

Z_i = bilangan acak ke-*i* dari deretnya

Z_{i-1} = bilangan acak sebelumnya

a = factor pengali

c = *increment*

m = *modulus*

R_i = Bilangan random ke i [0,1]

Operasi modulo merupakan sisa pembagian dari satu bilangan oleh bilangan yang lain. Jika diberikan dua bilangan a dan b, a modulo b (disingkat sebagai a mod b) dapat disamakan dengan sisa dari pembagiannya. Misalnya, "5 mod 4" akan menghasilkan 1, karena 5 dibagi dengan 4 bersisa 1, sedangkan "9 mod 3" akan menghasilkan 0 karena

pembagian 9 oleh 3 tidak meninggalkan sisa. Ketika a atau b adalah negatif, definisi ini menjadi memiliki celah dan banyak bahasa pemrograman memberikan definisi yang berbeda-beda. Meskipun biasanya a dan n keduanya adalah bilangan bulat, banyak sistem penghitungan yang memungkinkan penggunaan jenis operan numerik lainnya. (Hidayat, 2010)

Kunci pembangkit adalah Z_0 yang disebut **umpan** (*seed*). Apabila nilai *increment* (c) adalah 0, maka disebut algoritma tersebut *Multiplicative Congruential Generator*. Maka bentuk *Pseudo RNG* dari *Multiplicative Congruential Generator* sebagai berikut (Riani, 2010):

$$Z_i = (aZ_{i-1}) \bmod m$$

$$R_i = Z_i / m$$

Untuk membangkitkan bilangan bulat acak dari sebuah rentangan data, kita bisa menggunakan distribusi diskrit uniform. Dalam meninjau distribusi ini probabilitas untuk menyeleksi setiap bilangan integer diantara *a* dan *b* adalah sama saja (*Equally likely*). Dengan demikian

$$fx\ (x = r) = p$$

$$r = a, \ldots, b$$

Dan didapat :

$$(b - a + 1)\ p = 1,$$

maka :

$$p = \frac{1}{b - a + 1}$$

Untuk mendapatkan nilai acak X kita terlebih dahulu harus generate R dulu (Random Number) menggunakan generator dan menyusunnya dalam bentuk :

$$X = a + [R/p]$$

$$X = a + [R(b\text{-}a+1)]$$

[R(*b*-*a*+1)] menunjukkan angka terbesar integer. Dengan demikian tahap-tahap ini akan menunjuk pada :

$$X = \begin{cases} 1.a & untuk \quad 0 \leq R < P \\ 2.a+1 & untuk \quad P \leq R < 2P \\ 3.a+i & untuk \quad Pi \leq R < P(I=1) \\ 4.b \; untuk \; P(b-a) \leq R < P(b-a+1) = 1 \end{cases}$$

Hal ini menunjukkan probabilitas dari R berada dalam suatu interval khusus dari Pi sampai P(I + 1), dan ini adalah P, yang mana X adalah distribusi uniform pada permulaan bilangan integer. (MDP, 2013)

1.1.3 Algoritma A Star (A*)

Path Finding atau pencarian jalan adalah salah satu dasar algoritma dalam konsep menggerakkan karakter dalam game. Tanpa algoritma pecarian jalan ini, karakter dalam game yang dibuat tidak akan bisa bergerak dengan benar atau seseuai keinginan kita. Walaupun bisa jalan, belum tentu karakter tersebut bergerak sesuai keinginan kita (Ramadhani, 2008). Pengembangan yang paling terkenal dari bentuk pencarian menggunakan metode Best-first search ialah algoritma pencarian A*. Algoritma ini mengevaluasi titik dengan menjumlahkan antara *g(n)*, yaitu harga jalur terpendek dari *starting point* ke A

(A adalah simpul yang sedang dijalankan dalam algoritma pencarian jalur terpendek) dan *h(n)*, harga estimasi dari sebuah node hingga node tujuan. (Russell & Norvig, 2003)

$$G(n) = \sqrt{Xn^2 + Yn^2} \quad [1]$$

$$H(n) = |X(target) - X(n)| + |Y(target) - Y(n)| \quad [2]$$

$$F(n) = G(n) + H(n) \quad [3]$$

Prinsip algoritma ini adalah mencari jalur terpendek dari sebuah simpul awal (*starting point)* menuju simpul tujuan dengan memperhatikan harga (F) terkecil. Diawali dengan menempatkan A pada *starting point*, kemudian memasukkan seluruh simpul yang bertetangga dan tidak memilik atribut rintangan dengan A ke dalam *open list* (*open list* adalah tempat menyimpan data simpul yang mungkin diakses dari starting point maupun simpul yang sedang dijalankan.). Kemudian mencari nilai F terkecil dari simpul-simpul dalam *open list* tersebut. Kemudian memindahkan A ke simpul yang memiliki nilai F terkecil. Simpul sebelum A disimpan sebagai *parent* dari A dan dimasukkan ke dalam *closed list* (*closed list* adalah tempat menyimpan data simpul sebelum A yang juga merupakan bagian dari jalur terpendek yang telah berhasil didapatkan.).

Jika terdapat simpul lain yang bertetangga dengan A (yang sudah berpindah) namun belum termasuk kedalam anggota *open list*, maka masukkan simpul-simpul tersebut ke dalam *open list.* Setelah itu, bandingkan nilai G yang ada dengan nilai G sebelumnya (pada langkah awal, tidak perlu dilakukan perbandingan nilai G). Jika nilai G sebelumnya lebih kecil maka A kembali ke posisi awal. Simpul yang pernah dicoba dimasukkan ke dalam

closed list. Hal terebut dilakukan berulang-ulang hingga terdapat solusi atau tidak ada lagi simpul lain yang berada pada *open list*.

Menggunakan algoritma pencarian A* untuk menggerakkan karakter dalam game mengharuskan menginterpretasikan lingkungan game sebagai graf. Dengan kata lain, menentukan apa yang direpresentasikan oleh simpul dan sisinya. Sebagai contoh, dalam game 2D (dua dimensi), berbentuk ubin-ubin atau kotak-kotak seperti papan catur yang disebut grid , ini mudah. Simpul-simpul merepresentasikan ubin-ubin yang ada. Kita bisa menggerakkan karakter dari ubin satu ke ubin-ubin lain yang merupakan tetangga dari ubin tadi (kecuali dalam ubin tetanga terdapat penghalang).

Sejauh ini, dalam ilmu komputer (*computer science*), algoritma A* merupakan algoritma terbaik di antara algoritma-algoritma pencarian graf untuk mencari jalan dengan cost terkecil dari simpul awal menuju simpul akhir. (Ramadhani, 2008)

1.1.4 *Platform Android*

Android merupakan salah satu sistem operasi yang terkenal dikalangan perangkat *mobile* yang merupakan pesaing dari sistem operasi perangkat *mobile* lainnya seperti *Windows Phone*, *iOS*, *BlackBerry*, *MeeGo*, *Bada* dan *Symbian*. Namun berbeda dengan sistem operasi *mobile* lainnya, karena *Android* bersifat *Open Source* yang memungkinkan untuk dikembangkan lebih lanjut oleh pihak ketiga.

Definisi pertama yang diungkapkan oleh Wikipedia yang mendefinisikan Android adalah sistem operasi untuk telepon seluler yang berbasis Linux.

Lebih lengkap lagi, menurut Ajith Abraham, Jamie Lloret Mauri dan John Buford mengemukakan bahwa Android adalah sistem operasi milik Google. Sistem operasi ini berbeda dengan sistem operasi yang sebelumnya bisa digunakan pada mobile devices, notebook, dan komputer. Sejalan dengan pendapat dari Ajith Abraham, Jamie Lloret Mauri dan John Buford, Matamaya Studio menjelaskan bahwa Android merupakan sistem operasi dari Google yang bersifat open *source*, sehingga berbeda dengan windows dimana kita harus membeli lisensinya.

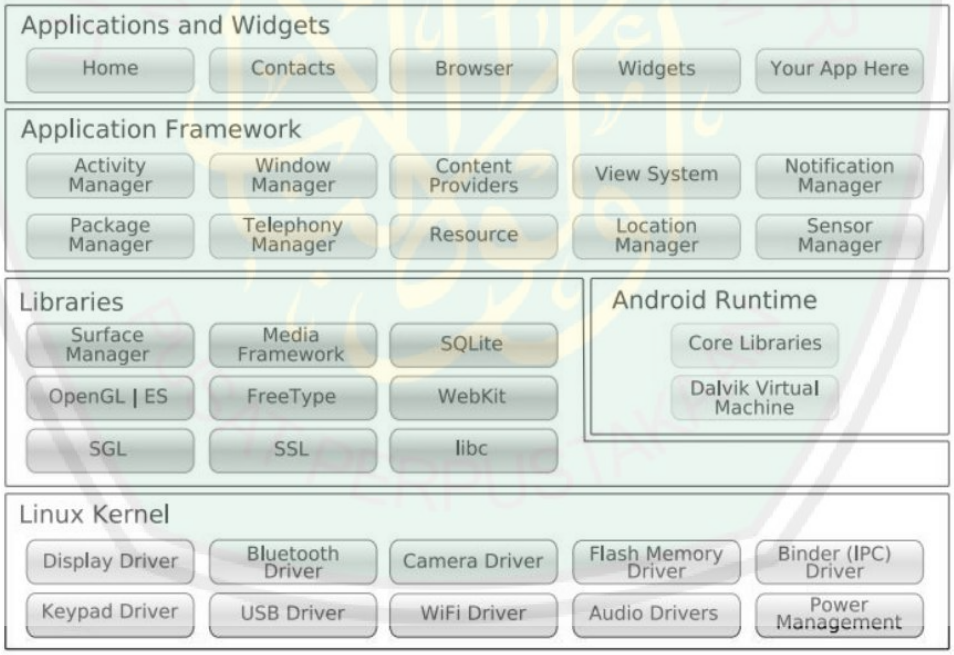
Sejarah singkat android yaitu, pada Juli 2000, Google bekerjasama dengan Android Inc., perusahaan yang berada di Palo Alto, California Amerika Serikat. Para pendiri Android Inc. bekerja pada Google, di antaranya Andy Rubin, Rich Miner, Nick Sears, dan Chris White. Saat itu banyak yang menganggap fungsi Android Inc. hanyalah sebagai perangkat lunak pada telepon seluler. Sejak saat itu muncul rumor bahwa Google hendak memasuki pasar telepon seluler. Di perusahaan Google, tim yang dipimpin Rubin bertugas mengembangkan program perangkat seluler yang didukung oleh kernel Linux. Hal ini menunjukkan indikasi bahwa Google sedang bersiap menghadapi persaingan dalam pasar telepon seluler. Pada tahun 2005 Google mengakusisi Android Inc yang pada saat itu dimotori oleh Andy Rubin, Rich Miner, Nick Sears dan Crish White. Yang kemudian pada tahun itu juga memulai membangun platform Android secara lebih intensif. Kemudian pada tanggal 12 November 2007 Google bersama Open Handset Aliance (OHA) yaitu konsorium perangkat lunak mobile terbuka, merilis Google Android SDK, setelah mengumumkannya seminggu sebelumnya.

Android built in pada Linux kernel (Open Linux Kernel), dengan sebuah mesin virtual yang telah didesain dan untuk mengoptimalkan penggunaan sumberdaya memori dan handware pada lingkungan perangkat mobile (Mobile Environment). Dalvik adalah nama dari Android Virtual Mesin, yang merupak interpreter virtual mesin yang akan mengeksekusi file kedalam format Dalvik Executable (*.dex). Sebuah format yang dirancang untuk ruang penyimpanan yang efisien dan eksekusi memori yang terpetakan (memory-mappable execution). Dlavik Virtual Mesun (Dalvik VM) berbasi register, dan dapat mengeksekusi kelas (class) yang telah terkompilasi pada compier bahasa java, kemudian di transformasikan kedalam native format dengan menggunakan Tool "dx" yang telah terintegrasi. (Safaat, 2001:1)

Android dirancang dengan arsitektur sebagai berikut (Safaat, 2001:6-9):

1. *Application* dan *Widgets*, merupakan layer dimana kita berhubungan dengan aplikasi saja, seperti aplikasi untuk browsing. Selain itu, fungsi-fungsi seperti telepon dan sms juga terdapat pada layer ini.
2. *Application Frameworks*, merupakan layer dimana para pembuat aplikasi melakukan pengembangan/ pembuatan aplikasi yang akan dijalankan di sistem operasi *Android*. Beberapa komponen yang terdapat pada layer ini adalah, *Views*, *Content Provider*, *Resource Manager*, *Notification Manager* dan *Activity Manager*.
3. *Libraries*, merupakan layer dimana fitur-fitur *Android* berada yang dapat digunakan untuk menjalankan aplikasi. *Library* yang disertakan seperti *library* untuk pemutaran audio dan video, tampilan, grafik, *SQLite*, *SSL* dan *Webkit*, dan 3D.

4. *Android Run Time*, merupakan layer yang berisi *Core Libraries* dan *Dalvik Virtual Machine* (DVK). *Core libraries* berfungsi untuk menerjemahkan bahasa Java/C. Sedangkan DVK merupakan sebuah virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien.
5. *Linux Kernel*, merupakan layer yang berfungsi sebagai *abstraction*/ pemisah antara *hardware* dan *software*. *Linux kernel* inilah yang merupakan inti sistem operasi dari *Android* yang berfungsi untuk mengatur sistem proses, *memory*, *resouce*, dan *driver*. *Linux kernel* yang digunakan *Android* adalah *linux kernel* release 2.6.



Gambar 1. 1 Arsitektur Android (Sumber: Safaat, 2001 : 9)

Beberapa versi dari android yang sudah diluncurkan diantaranya:

1. **Andorid versi 1.1**. Pada 9 Maret 2009, Google merilis Android versi 1.1. Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam

alarm, voice search (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.

2. **Android versi 1.5 (Cupcake).** Pada pertengahan Mei 2009, Google kembali merilis telepon seluler dengan menggunakan Android dan SDK (Software Development Kit) dengan versi 1.5 (Cupcake). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke Youtube dan gambar ke Picasa langsung dari telepon, dukungan Bluetooth A2DP, kemampuan terhubung secara otomatis ke headset Bluetooth, animasi layar, dan keyboard pada layar yang dapat disesuaikan dengan sistem.
3. **Android versi 1.6 (Donut).** Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol applet VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus; kamera, camcorder dan galeri yang dintegrasikan; CDMA / EVDO, 802.1x, VPN, Gestures, dan Text-to-speech engine; kemampuan dial kontak; teknologi text to change speech (tidak tersedia pada semua ponsel; pengadaan resolusi VWGA.
4. **Android versi 2.0/2,1 (Eclair).** Pada 3 Desember 2009 kembali diluncurkan ponsel Android dengan versi 2.0/2.1 (Eclair), perubahan yang dilakukan adalah pengoptimalan hardware, peningkatan Google Maps 3.1.2, perubahan UI dengan browser baru dan dukungan HTML5, daftar kontak yang baru, dukungan flash untuk kamera 3,2 MP, digital Zoom,

dan Bluetooth 2.1. Untuk bergerak cepat dalam persaingan perangkat generasi berikut, Google melakukan investasi dengan mengadakan kompetisi aplikasi mobile terbaik (killer apps - aplikasi unggulan). Kompetisi ini berhadiah $25,000 bagi setiap pengembang aplikasi terpilih. Kompetisi diadakan selama dua tahap yang tiap tahapnya dipilih 50 aplikasi terbaik. Dengan semakin berkembangnya dan semakin bertambahnya jumlah handset Android, semakin banyak pihak ketiga yang berminat untuk menyalurkan aplikasi mereka kepada sistem operasi Android. Aplikasi terkenal yang diubah ke dalam sistem operasi Android adalah Shazam, Backgrounds, dan WeatherBug. Sistem operasi Android dalam situs Internet juga dianggap penting untuk menciptakan aplikasi Android asli, contohnya oleh MySpace dan Facebook.

5. **Android versi 2.2 (Froyo: Frozen Youghurt).** Pada 20 Mei 2010, Android versi 2.2 (Froyo) diluncurkan. Perubahan-perubahan umumnya terhadap versi-versi sebelumnya antara lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat, intergrasi V8 JavaScript engine yang dipakai Google Chrome yang mempercepat kemampuan rendering pada browser, pemasangan aplikasi dalam SD Card, kemampuan WiFi Hotspot portabel, dan kemampuan auto update dalam aplikasi Android Market.
6. **Android versi 2.3 (Gingerbread).** Pada 6 Desember 2010, Android versi 2.3 (Gingerbread) diluncurkan. Perubahan-perubahan umum yang didapat dari Android versi ini antara lain peningkatan kemampuan permainan (gaming), peningkatan fungsi copy paste, layar antar muka (User

Interface) didesain ulang, dukungan format video VP8 dan WebM, efek audio baru (reverb, equalization, headphone virtualization, dan bass boost), dukungan kemampuan Near Field Communication (NFC), dan dukungan jumlah kamera yang lebih dari satu.

7. **Android 3.0 - 3.2 Honeycomb.** Android Honeycomb dirancang khusus untuk tablet. Android versi ini mendukung ukuran layar yang lebih besar. User Interface pada Honeycomb juga berbeda karena sudah didesain untuk tablet. Honeycomb juga mendukung multi prosesor dan juga akselerasi perangkat keras (hardware) untuk grafis. Tablet pertama yang dibuat dengan menjalankan Honeycomb adalah Motorola Xoom. Perangkat tablet dengan platform Android 3.0 akan segera hadir di Indonesia. Perangkat tersebut bernama Eee Pad Transformer produksi dari Asus. Rencana masuk pasar Indonesia pada Mei 2011.
8. **Android 4.0 Ice Cream Sandwich**. Ice Cream Sandwich adalah versi terbaru Android untuk smartphone, tablet, dan lainnya. Ice Cream Sandwich dirilis pada 19 October 2011. Versi ini didasarkan untuk mengoptimalkan multitasking, banyak notifikasi, layar beranda yang dapat disesuaikan, dan interaktivitas mendalam serta cara baru yang ampuh untuk berkomunikasi dan berbagi konten. Ice Cream Sandwich adalah lapisan es krim yang biasanya berupa vanila yang terjepit antara dua cookies coklat, dan biasanya berbentuk persegi panjang.
9. **Android 4.1 - 4.2 Jelly Bean**. Android 4.1 Jelly Bean diumumkan pada 27 Juni 2012 pada konferensi Google l/O. Versi ini adalah yang tercepat dan terhalus dari semua versi Android. Jelly Bean 4.1 meningkatkan

kemudahan dan keindahan tampilan dari Ice Cream Sandwich dan memperkenalkan pengalaman pencarian Google yang baru di Android. Android 4.2 Jelly Bean diumumkan pada 29 October 2012, versi ini menawarkan peningkatkan kecepatan dan kemudahan Android 4.1 serta mencak up semua fitur baru seperti Photo Sphere dan desain baru aplikasi kamera, keyboard Gesture Typing, Google Now dan lainnya. Jelly Bean adalah sejenis permen yang juga populer disebut kacang jeli.

Beberapa keunggulan *Platform Android* adalah sebagai berikut (Safaat, 2001:3):

1. Lengkap (*Complete Platform*). Para desainer dapat melakukan pendekatan yang komprehensif ketika sedang mengembangkan *platform Android*. *Android* menyediakan banyak *tools* dalam membangun software dan merupakan sistem operasi yang aman.
2. Terbuka (*Open Source Platform*). *Platform Android* disediakan melalui lisensi *open source*.
3. Bebas (*Free Platform*). *Android* merupakan *platform* atau aplikasi yang bebas untuk dikembangkan. Tidak ada lisensi atau biaya royalti untuk dikembangkan pada *platform Android*.

1.1.5 AndEngine

Gambar 1. 2 Logo AndEngine

AndEngine adalah sebuah library untuk bahasa pemrograman android yang digunakan untuk mendevelop sepuah aplikasi permainan. Pada andengine, sudah terdapat kelas-kelas yang dapat mempermudah dalam pembuatan game yang kita inginkan. Kita hanya tinggal memanggil dan mengimplementasikan apa yang ada dalam kelas-kelas tersebut. Sebagai contoh saja kelas animasi, sound, physic dan lain sebagainya. Library ini bersifat open *source* sehingga siapapun bisa memakainya tanpa harus bingung akan biaya pembuatan game. Seperti halnya android yang memakai bahasa java sebagai sandaran untuk pembuatan aplikasinya, andengine juga berbasis bahasa pemrograman java dalam pengembangannya.

Jika dibandingkan dengan engine-engine lainnya untuk pembuatan game di android, andengine mungkin masih kalah jauh. Akan tetapi andengine sendiri memiliki beberapa kelebihan lain dibandingkan dengan engine-engine lainnya.

Berikut kelebihan dari andengine:

- Free
- Mudah digunakan
- Terdapat berbagai *extension* yang dapat digunakan hanya dengan mengimportnya saja. Sebagai contoh yaitu *Physics extension, Multitouch extension, Multiplayer extension, bahkan Augmented Reality(AR) extension*
- Komunitas pengguna andengine lumayan banyak sehingga mudah mencari referensi dan solusi masalah.

Dalam penggunaannya, andengine sendiri mengadopsi cara kerja dunia perfilman. Dimana setiap layar di ibaratkan sebuah scene dalam film. Seperti layar splash di inisialisasikan sebagai *SplashScene*, StartMenu diinisialisasikan sebagai *MenuScene*. Untuk pembuatan karakter sendiri digunakan object dengan tipe *Sprite* atau *Animated Sprite*. Animasi dari karakter tersebut menggunakan beberapa gambar yang diputar secara berulang-ulang seperti halnya dalam pembuatan film kartun.

1.2 Penelitian Terkait

Penelitian yang mempunyai hubungan dengan penelitian ini adalah :

1. Penelitian oleh Mahasiswa dari Universitas Islam Negeri Maulana Malik Ibrahim Malang. Penelitiannya ialah membuat aplikasi permainan *Aplikasi Permainan Sudoku huruf hijaiyah Menggunakan Algoritma Backtracking Dan Multiplicative CRNG Sebagai Pembangkit Dan Penyelesai Permainan*. Dari penelitian tersebut, penulis mengambil referensi

bagaimana membangkitkan nilai pada permainan Sudoku tersebut. Dan kemudian mengaplikasikannya pada permainan yang akan penulis buat. (Tofin, 2012)

2. Game Santri Story untuk Pengenalan Huruf Hijaiyah Menggunakan Metode MCRN-Generator dari Universitas Islam Negeri Maulana Malik Ibrahim Malang. Penelitian dari Miftakhul Huda tersebut menggunakan MCRN-Generator untuk pengacakan huruf hijaiyah, serta posisi musuh yang akan muncul dalam setiap level (Huda, 2012).
3. Penelitian Oleh Hapsari Tilawah dari ITB, Bandung yang menerapkan algoritma A Star (A*) untuk menyelesaikan sebuah Maze. Penelitian tersebut menjelaskan langkah-langkah algoritma A Star untuk menyelesaikan *maze* dari awal sampai akhir. Hasil kesimpulan penelitian tersebut menunjukkan bahwa algoritma A Star dapat diterapkan untuk menyelesaikan suatu *maze* yang digunakan penulis sebagai algoritma pencarian pada *asteroid* untuk menemukan *player* (Tilawah, 2010)

1.3 Metode Penelitian

Peneliti membagi pengerjaan penelitian ini menjadi beberapa tahap, antara lain:

1. Studi literatur

   Pada tahap ini dilakukan berbagai pengumpulan informasi terkait beberapa hal berikut:

   - Pengumpulan informasi tentang pembuatan game pada android

- Pengumpulan informasi tentang algoritma *Multiplicative CRNG* sebagai pengacak dalam permainan

2. Perancangan dan desain aplikasi

   Perancangan aplikasi terdiri atas perancangan proses-proses utama dan desain aplikasi yang terdiri atas desain menu *game* dan desain utama dari *game* itu sendiri.

3. Pembuatan aplikasi

   Perancangan dan desain aplikasi diimplementasikan menggunakan bahasa pemrograman java.

4. Uji coba dan evaluasi

   Uji coba dan evaluasi dilakukan terhadap tahapan aplikasi dan hasil dari pengacakan letak *asteroid* yang telah dibangkitkan menggunakan *Multiplicative CRNG*

5. Penyusunan laporan

   Penyusunan laporan akhir merupakan dokumentasi dari keseluruhan pelaksanaan penelitian dan diharapkan bermanfaat bagi penelitian lebih lanjut.

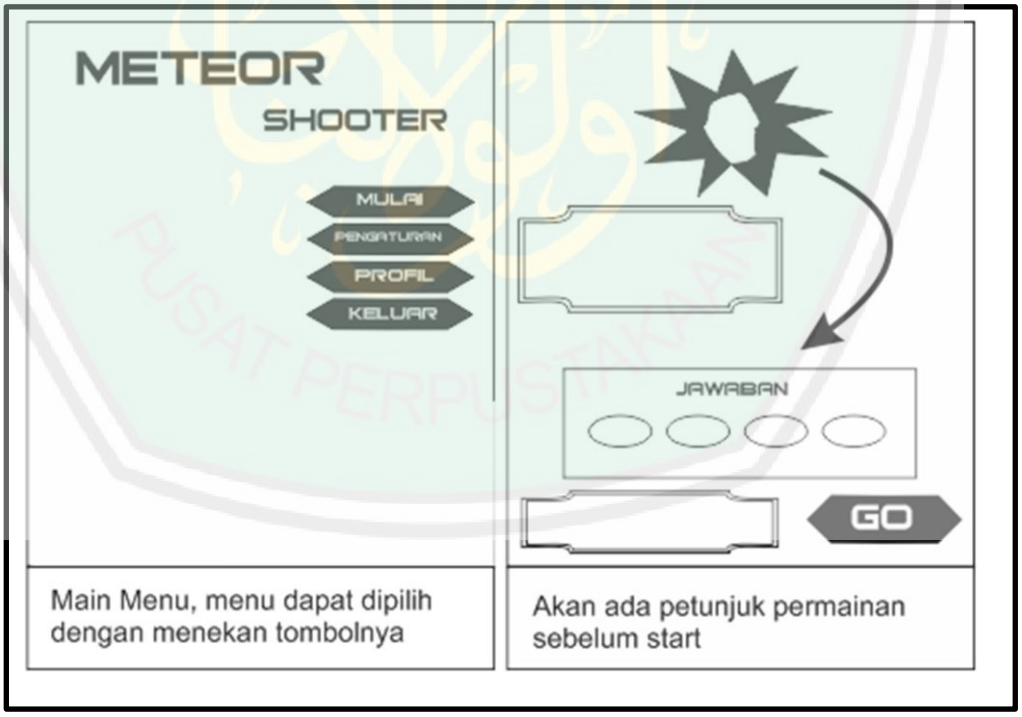
BAB III

DESAIN DAN RANCANGAN SISTEM

1.1 Analisis dan Perancangan Sistem

Konsep permainan adalah dimana pemain bertahan selama mungkin dari segala serangan *asteroid* yang muncul dengan menjawab pertanyaan yang ada di asteroid. Poin bertambah setiap *user* berhasil meledakkan asteroid. Dan game akan berakhir disaat asteroid berhasil melewati batas sebanyak nyawa yang disediakan.

1.1.1 Story Board & Story Line



Gambar 3. 1 Story Board Asteroid Shooter

Pada game yang akan dibuat, akan disajikan beberapa *scene* seperti yang ada pada Gambar 3.1, dimana *scene* pertama yang pemain dapatkan adalah *scene* main menu. Di dalam main menu terseput terdapat beberapa pilihan menu yaitu mulai bermain (*Play*), menu pengaturan (*Option*), menu profil, dan menu mengakhiri permainan(*Quit*). Jika pemain memulai permainan, maka pemain akan berpindah *scene* menuju ke *scene* permainan. Akan tetapi sebelum memulai permainan, pemain akan menuju *loading scene* terlebih dahulu. Di dalam *loading scene* ini pemain akan mendapatkan petunjuk permainan dan barulah pemain akan menuju ke *scene* permainan. Dalam *scene* permainan, pemain akan mendapati HUD (Heads-Up Display) yang menampilkan score dan panel untuk menjawab pertanyaan yang ada untuk setiap target dan menampilkan nyawa sang pemain.

Dalam permainan ini, permainan akan berakhir jika batas nyawa yang diberikan kepada pemain habis. Besarnya nyawa pemain tergantung pada pemilihan tingkat kesulitan. Saat permainan berakhir, *scene* permainan akan dialihkan menuju ke *gameover scene.* Dalam gameover scene ini, pemain di sajikan menu konfirmasi, apakah pemain akan bermain lagi atau kembali ke main menu.

Permainan ini tidak memiliki *ending* atau tamat. Permainan ini bertujuan untuk menunjukan siapa yang dapat memiliki *score* tertinggi dalam permainan sehingga satu-satunya *ending* dalam permainan ini adalah *gameover*

1.1.2 Tingkat Kesulitan

Dalam permainan ini, tingkat kesulitan dibagi menjadi beberapa tingkat yaitu mudah, sedang, dan sulit. Tingkatan ini, didasarkan dari setting yang ditentukan oleh user sendiri. Setiap tingkatan memiliki karakteristik sendiri-sendiri. Berikut rincian dari tingkatan-tingkatan tersebut:

1. Tingkat Mudah
   - Kecepatan musuh lambat
   - *Respawn time* musuh lambat
   - Pertanyaan hanya dengan operator pertambahan
   - Batas tertabrak musuh 5 kali
2. Tingkat Sedang
   - Kecepatan musuh meningkat menjadi sedang
   - *Respawn time* musuh sedang
   - Pertanyaan bertambah dengan operator pertambahan dan pengurangan
   - Batas tertabrak musuh 4 kali
3. Tingkat Sulit

- Kecepatan musuh meningkat menjadi cepat
- *Respawn time* musuh cepat
- Pertanyaan bertambah dengan operator pertambahan, pengurangan dan perkalian
- Batas tertabrak musuh 3 kali

1.1.3 Character

- *Player*

Player dalam permainan ini di inisialisasikan sebagai pesawat prajurit yang diharuskan melindungi pesawat induk yang ada di belakangnya. Pesawat ini tidak dapat bergerak hanya diam di tempat dan dapat menembak dan menjawab *asteroid* yang ada.

Gambar 3. 2 Pesawat *Player*

- Asteroid

Karakter target dalam permainan ini adalah asteroid. Asteroid ini dalam permainan merupakan sebuah objek yang harus dihancurkan sebelum asteroid tersebut menghancurkan pesawat induk. Disini untuk menghancurkan asteroid tersebut, maka kita harus menjawab pertanyaan yang terdapat di asteroid.

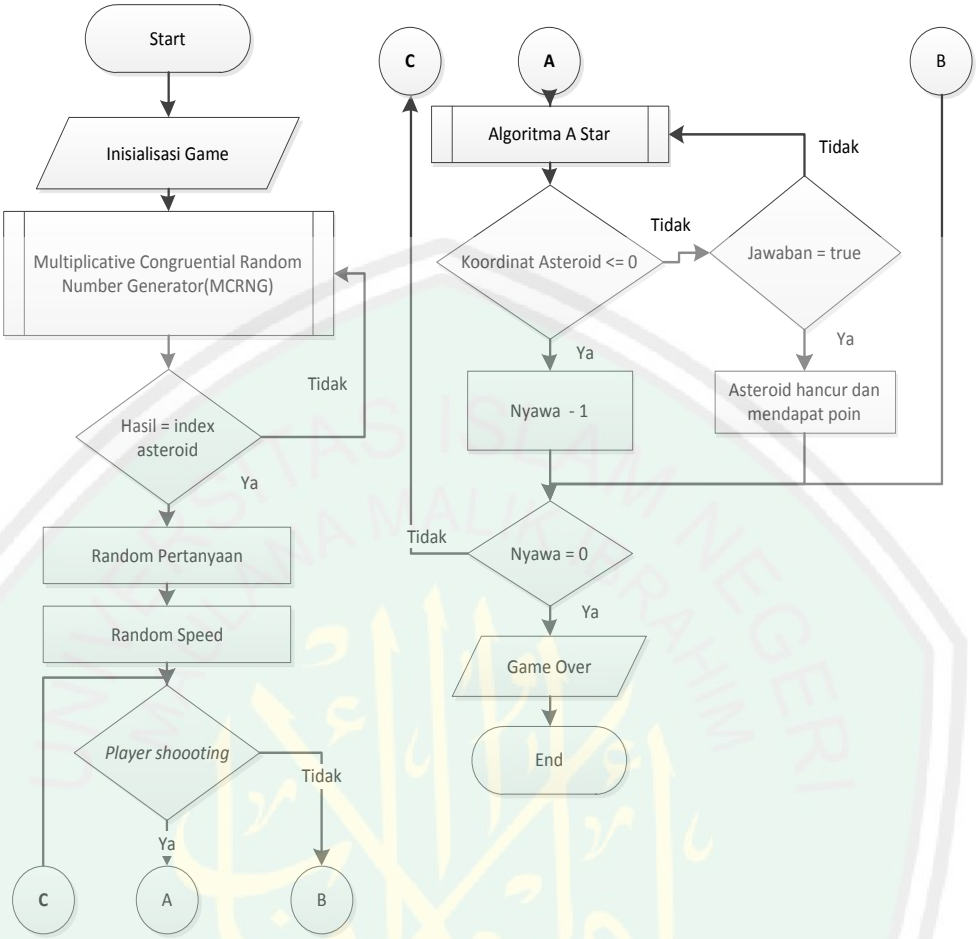
Gambar 3. 3 Asteroid

- Pesawat Induk

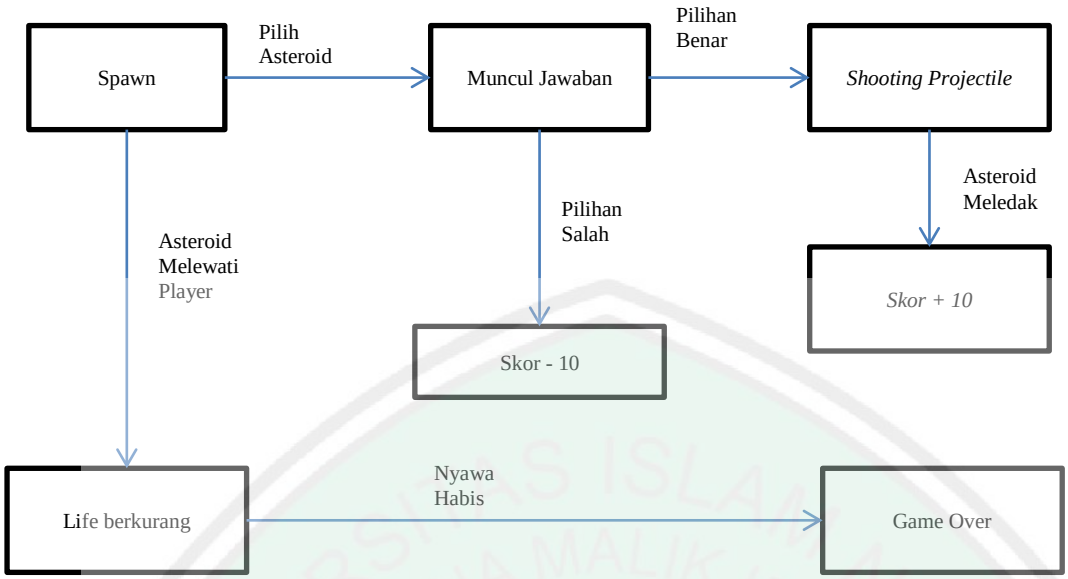
Pesawat induk ini adalah NPC pemanis yang hanya keluar pada saat awal permainan. Tugas dari kapal induk ini adalah awal dari keluarnya *player*

Gambar 3. 4 Pesawat Induk

1.1.4 Flowchart dan FSM Permainan



Gambar 3. 5 Flowchart Permainan



Gambar 3. 6 FSM Permainan

Berdasarkan Flowchart permainan di atas, metode MCRNG digunakan untuk pengacakan *spawning position* setelah game di inisialisasikan. Hasil random dari generator dicocokan dengan index asteroid yang ada dalam *array* asteroid. Kemudian asteroid yang memiliki indeks yang sama dengan hasil random, maka digerakan dengan kecepatan yang random juga. Kemudian saat *player* menembak, maka Algoritma AStar akan bekerja pada *projectile* yang ditembakan, dimana *projectile* tersebut akan mengejar asteroid yang menjadi target.

1.2 Flowchart MCRNG

Gambar 3. 7 Flowchart MCRNG

Misalkan kita akan mengacak 10 nilai. Kemudian kita memiliki sebuah seed (Z_0) yang diambil secara acak. Z_0 = 12357, a = 19, dan modulo (m) kita ambil 128, kita masukkan ke rumus $Zi = (aZi\text{-}1) \bmod m$. Setelah menghitungnya, bagi Zi tersebut dengan nilai m. Maka kita akan mendapatkan nilai probabilitas(kemungkinan) 0.2421875, 0.6015625, 0.4296875, 0.1640625, 0.1171875, 0.2265625, 0.3046875, 0.7890625, 0.9921875, 0.8515625. kemudian setelah kita menemukan probabilitasnya, kita kali kan dengan rentang nilai yang akan diacak dan kemudian kita bulatkan . Misalkan saja rentang nilainya adalah 10 maka nilai yang akan muncul adalah 2,6,4,1,1,2,3,7,9,8. Untuk percobaan lainnya kita coba aplikasi kan pada method seperti di bawah ini:

```
public void random_MCRNG() {
      int seed = (int) (System.nanoTime());
      int m = 128;
      for (int i = 0; i < 10; i++) {
            int xn = (19 * seed) % m;
            double r = (double) xn / m;
            int nilai = ba + (int)(r*(bb-ba+1));
            System.out.println(r + ", " + nilai);
            seed = xn;
      }
}
```

Gambar 3. 8 *Source* Uji MCRNG

Dengan menggunakan Z_0 yang bernilai acak yang diambil dari *System.nanoTime* dan nilai *a* dan *m* merupakan variabel bebas yang dapat diubah-ubah, kita dapatkan nilai randomnya seperti tabel dibawah. Kemudian nilai random tersebut kita gunakan sebagai probabilitas untuk mendapatkan nilai X(*integer*) dengan menggunakan rumus distribusi diskrit uniform

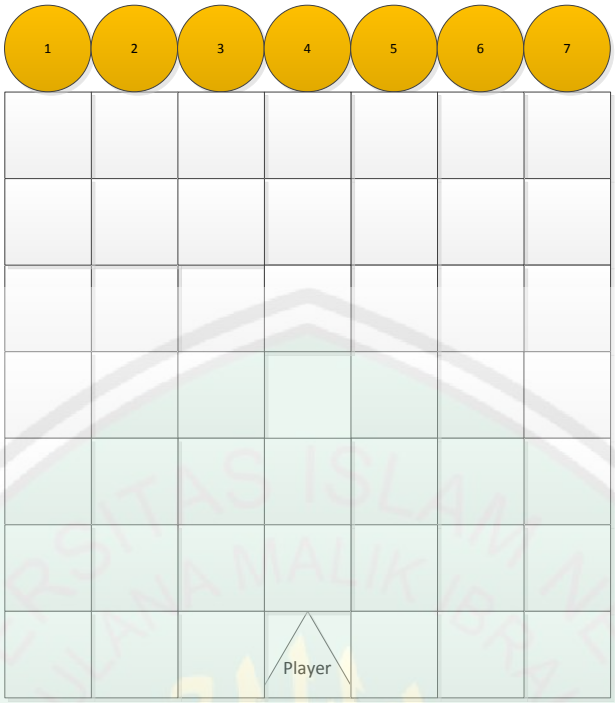
$$X = a + [\mathrm{R}(b - a + 1)]$$

Tabel 3. 1 Percobaan pengacakan 10 data menggunakan MCRNG.

No	*a*	*m*	Nilai Random	Hasil nilai yang didapatkan
1	3	11	0.5455, 0.6364, 0.9091, 0.7273, 0.1818, 0.5455, 0.6364, 0.9091, 0.7273, 0.1818	5,6,9,7,1,5,6,9,7,1
2	5	17	0.2941, 0.4706, 0.3529, 0.7647, 0.8235, 0.1176, 0.5882, 0.9412,	2,4,3,7,8,1,5,9,7,5

			0.7059, 0.5294	
3	7	17	0.9412, 0.5882, 0.1176, 0.8235, 0.7647, 0.3529, 0.4706, 0.2941, 0.0588, 0.4118	9,5,1,8,7,3,4,2,0,4
4	5	11	0.5455, 0.0909, 0.8182, 0.4545, 0.6364, 0.5455, 0.0909, 0.8182, 0.4545, 0.6364	5,0,8,4,6,5,0,8,4,6
5	77	127	0.252, 0.4016, 0.9213, 0.937, 0.1496, 0.5197, 0.0157, 0.2126, 0.3701, 0.4961	2,4,9,9,1,5,0,2,3,4

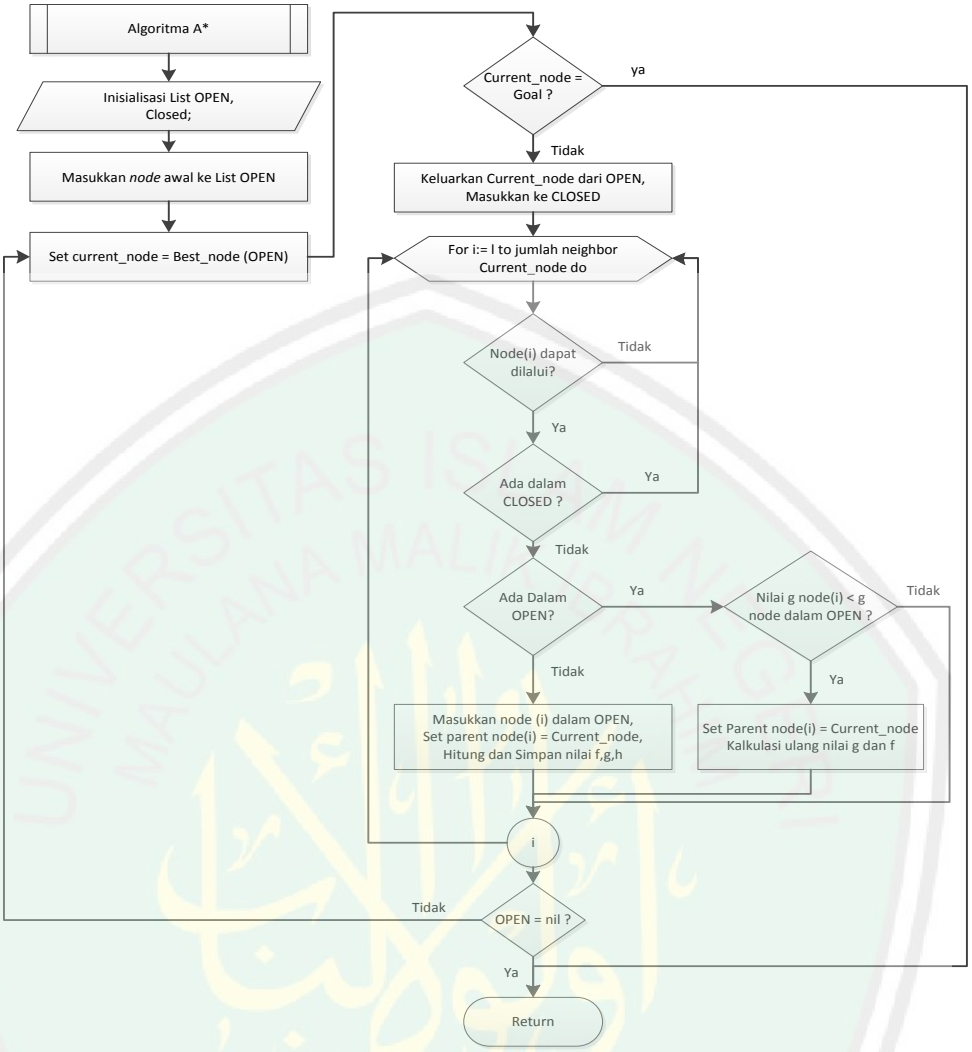
Dalam permainan ini algoritma MCRNG ini akan digunakan untuk mengacak asteroid manakah yang akan muncul. Maksudnya adalah, dalam permainan ini, asteroid-asteroid tersebut sudah di inisialisasi terlebih dahulu dan diberi index. Seperti ilustrasi dibawah ini:



Gambar 3. 9 Ilustrasi penempatan Asteroid.

Pada Gambar 3.7, MCRNG akan men-*generate* index tersebut secara acak dan memunculkan asteroid dengan nomor index yang sesuai dengan hasil dari MCRNG tadi. Asteroid yang muncul tersebut akan bergerak lurus sampai melewati *player*. Ketika asteroid telah melewati *player*, maka nyawa *player* akan berkurang.

1.3 Flowchart Algoritma A*



Gambar 3. 10 Flowchart AStar

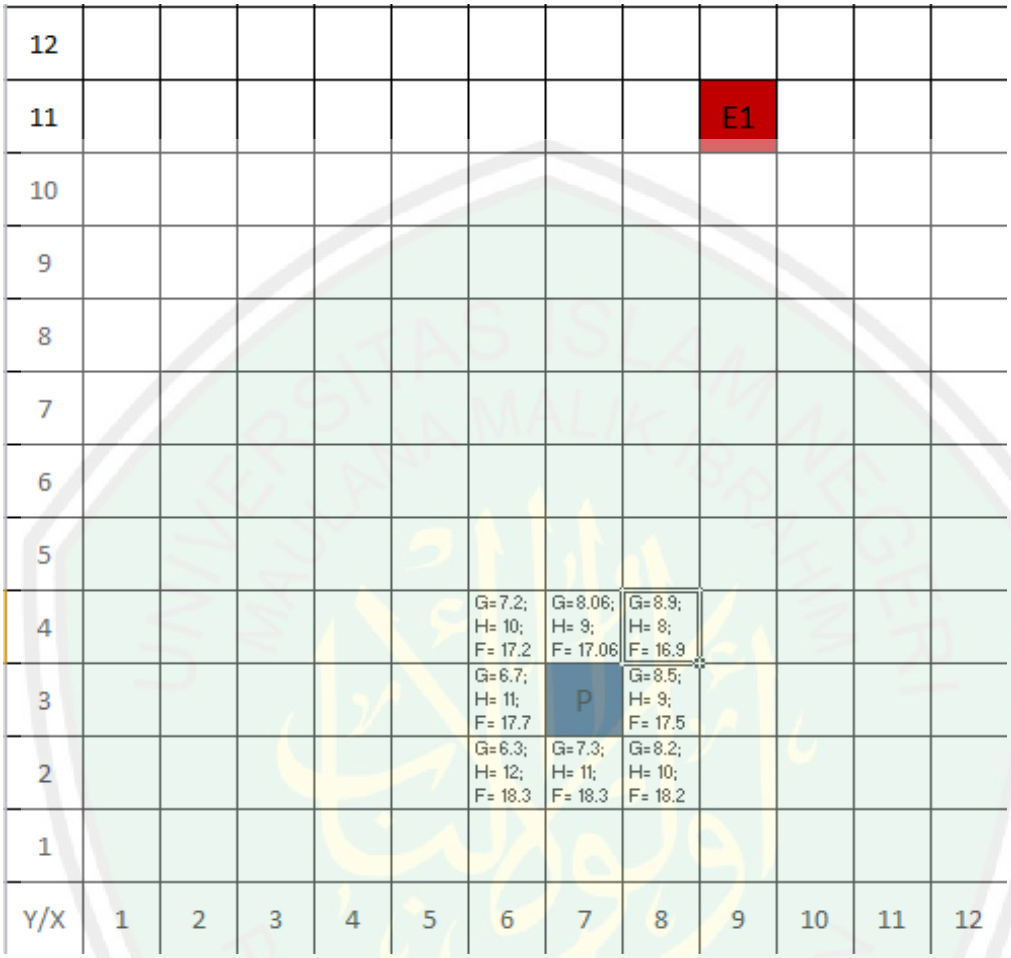
Contoh perhitungan:

Misalkan terdapat kondisi seperti dibawah ini, *player* berada pada koordinat (7,3) dan target berada pada koordinat (9,11) . Maka langkah pertama yaitu, deteksi semua node di sekitar *player* yang memungkinkan untuk *player* lalui. Hitung untuk setiap node yang dapat dilalui tersebut nilai G yaitu nilai yang dibutuhkan untuk berjalan satu node, H yaitunilai heuristic atau perkiraan dari node tersebut menuju target, dan F jumlah dari G dan H. seperti rumus dibawah ini:

$$G(n) = \sqrt{Xn^2 + Yn^2}$$

$$H(n) = |X(target) - X(n)| + |Y(target) - Y(n)|$$

$$F(n) = G(n) + H(n)$$



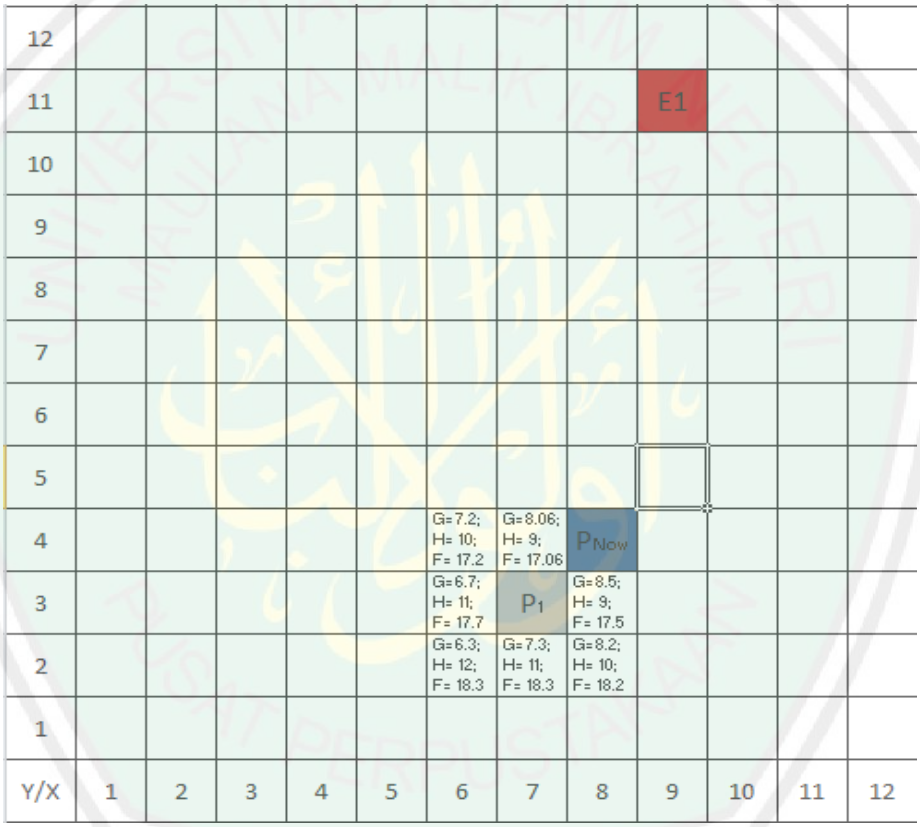
Gambar 3. 11 Pencarian F cost untuk tiap node di sekitar *player*

Seperti yang terlihat pada ilustrasi gambar 3.8, kita mendeteksi setiap node disekitar *player*.

- Node ke-1 memiliki nilai F sebesar 17.2
- Node ke-2 memiliki nilai F sebesar 17.06
- Node ke-3 memiliki nilai F sebesar 16.9
- Node ke-4 memiliki nilai F sebesar 17.5
- Node ke-5 memiliki nilai F sebesar 18.2

- Node ke-6 memiliki nilai F sebesar 18.3
- Node ke-7 memiliki nilai F sebesar 18.3
- Node ke-8 memiliki nilai F sebesar 17.7

Dari hasil tersebut diketahui bahwa node ke-3 memiliki nilai F yang terendah jika dibandingkan dengan tujuh node lainnya, sehingga kita pindahkan posisi *player* saat ini menuju ke node ke-3 seperti pada gambar 3.10 dibawah ini.



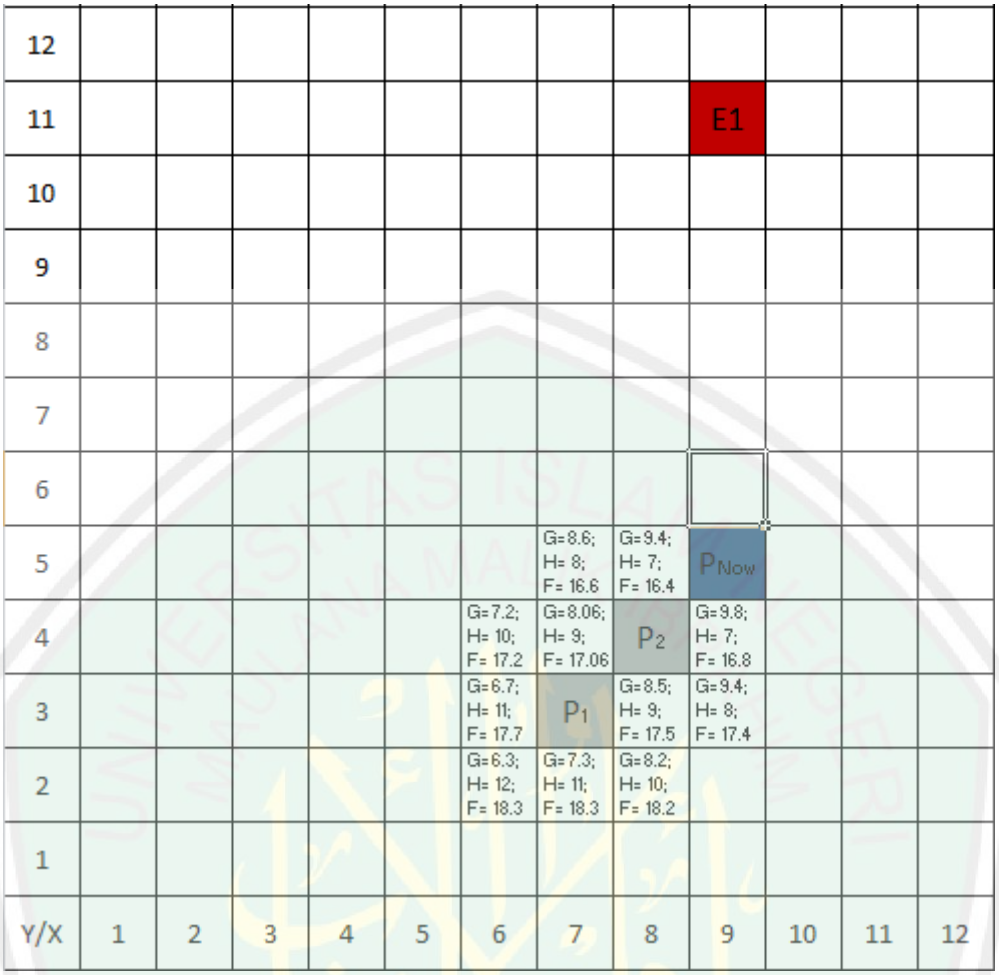
Gambar 3. 12 Posisi *player* dipindah menuju ke node 3

Kemudian setelah berpindah tempat, maka kita hitung kembali nilai F untuk setiap node tetangga yang baru

12												
11									E1			
10												
9												
8												
7												
6												
5							G=8.6; H= 8; F= 16.6	G=9.4; H= 7; F= 16.4	G=10.2; H= 6; F= 16.2			
4						G=7.2; H= 10; F= 17.2	G=8.06; H= 9; F= 17.06	P_{Now}	G=9.8; H= 7; F= 16.8			
3						G=6.7; H= 11; F= 17.7	P_1	G=8.5; H= 9; F= 17.5	G=9.4; H= 8; F= 17.4			
2						G=6.3; H= 12; F= 18.3	G=7.3; H= 11; F= 18.3	G=8.2; H= 10; F= 18.2				
1												
Y/X	1	2	3	4	5	6	7	8	9	10	11	12

Gambar 3. 13 Menghitung nilai F cost untuk setiap nilai tetangga yang baru

Seperti yang kita lihat pada gambar 3.11, nilai F terkecil pada iterasi ke 2 yaitu bernilai 16,2. Maka, sama seperti proses sebelumnya yaitu kita pindah kan *player* menuju ke node dengan nilai F terkecil tersebut.



Gambar 3. 14 Pergerakan *player* pada iterasi ke-2

Setelah berpindah, proses yang dilakukan sama seperti yang dilakukan sebelum-sebelumnya. Iterasi tersebut diulang terus menerus hingga akhirnya *player* berada di posisi target yang diinginkan. Hasil akhir ditunjukkan seperti gambar di bawah ini.

12												
11								G=13.6; H= 1; F= 14.6	E1	G=14.8; H=1; F= 15.86		
10								G=12.8; H= 2; F= 14.8	P_{Now}	G=14.14; H=2; F= 16.14		
9								G=12.0; H= 3; F= 15.04	P_7	G=13.4; H= 3; F= 16.4		
8								G=11.3; H= 4; F= 15.3	P_6	G=12.8; H= 4; F= 16.8		
7								G=10.6; H= 5; F= 15.6	P_5	G=12.2; H= 5; F= 17.2		
6								G=10; H= 6; F= 16	P_4	G=11.6; H= 6; F= 17.6		
5							G=8.6; H= 8; F= 16.6	G=9.4; H= 7; F= 16.4	P_3	G=11.1; H= 7; F= 18.1		
4						G=7.2; H= 10; F= 17.2	G=8.06; H= 9; F= 17.06	P_2	G=9.8; H= 7; F= 16.8	G=10.7; H= 8; F= 18.7		
3						G=6.7; H= 11; F= 17.7	P_1	G=8.5; H= 9; F= 17.5	G=9.4; H= 8; F= 17.4			
2						G=6.3; H= 12; F= 18.3	G=7.3; H= 11; F= 18.3	G=8.2; H= 10; F= 18.2				
1												
Y/X	1	2	3	4	5	6	7	8	9	10	11	12

Gambar 3. 15 Hasil akhir saat *player* mencapai target

BAB IV

HASIL DAN PEMBAHASAN

4.1. Implementasi

Bab ini menjelaskan tentang pengujian metode yang digunakan dan analisanya dalam membangun aplikasi yang telah dibuat serta implementasi metode tersebut di game yang akan dibuat. Tujuan dilakukan pengujian ini adalah untuk mengetahui apakah metode yang dipilih oleh penulis ini cocok dan dapat digunakan dalam aplikasi yang digunakan. Selain itu agar dapat diketahui kekurangannya agar dapat dikembangkan dan diperbaiki lebih lanjut.

4.1.1. Peralatan yang digunakan

Sebelum diimplementasikan, terlebih dahulu dipaparkan spesifikasi *hardware* dan *software* yang akan digunakan

Hardware:

Adapun hardware yang digunakan untuk pembuatan aplikasi dan pengujian metode tersebut adalah sebagai berikut :

- *Notebook* Asus K42JK : Processor Intel Core i3 2.3 GHZ, RAM 2 GB, HD 200 GB
- *Smartphone* Cross A7S : Processor Dual-Core ARMv7, RAM 512 MB, Storage 2GB

Software:

Sedangkan software yang digunakan dalam pembuatan aplikasi dan pengujian metode ialah:

- Netbeans IDE 7.3
- Eclipse Juno
- ADT 22.0
- Android ICS 4.0

4.1.2. Pengujian MCRNG

Metode pertama yang diuji adalah *Multiplicative Congruential Random Number Generator(MCRNG).* Untuk menghasilkan nilai acak, dibutuhkan *Seed* atau Z_0 yang bagus untuk menghasilkan nilai acak yang sempurna dan susah untuk di tebak. Untuk mendapatkan Z_0 , dalam aplikasi ini menggunakan *System.nanoTime()* dimana function tersebut mengembalikan nilai *long* dari waktu yang ditempuh aplikasi dari saat dimana ia mulai berjalan hingga *function* tersebut dipanggil.

Dengan rumus MCRNG :

$$Z_i = (aZ_{i-1}) \bmod m$$

pengujian dilakukan dalam 2 tahap. Tahap pertama untuk mencari seberapa efektif MCRNG bekerja dibandingkan dengan *random generator* bawaan java. Kemudian pada pengujian tahap kedua, karena nilai acuan (*a*) dan modulo (*m*) merupakan variable bebas yang dapat di ubah-ubah, maka kita uji untuk nilai acuan (*a*) dan modulo (*m*) dengan nilai yang berbeda-beda, dan kita hitung seberapa besar tingkat kesalahan pada MCRNG ini.

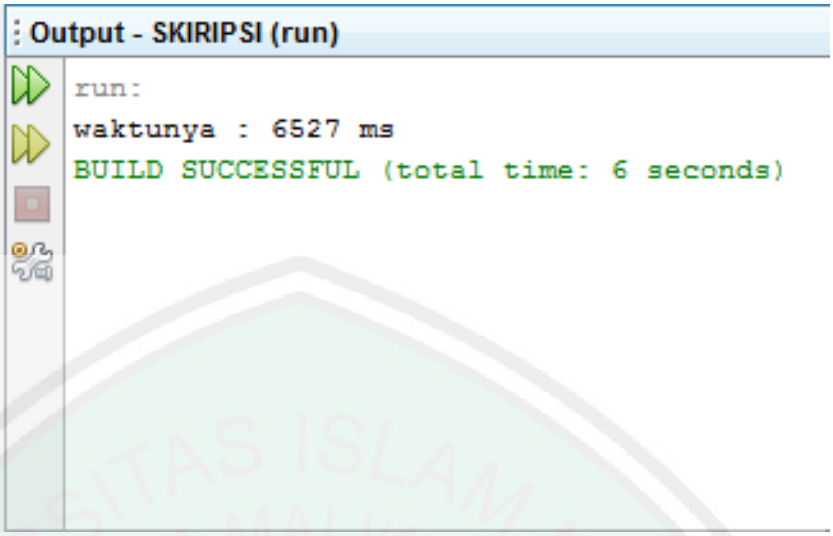
- **Hasil pengujian Tahap I :**

Pada pengujian tahap pertama ini kita membandingkan kecepatan kerja dari MCRNG yang di bandingkan dengan generator bawaan dari kelas *Math* di java. Kita buat terlebih dahulu sebuah kelas untuk pengujian *Math.random*. Dengan menggunakan *source code* sebagai berikut ini:

```
public class Test {
  public static void main(String[] args) {
     long time1 = System.nanoTime()/1000000;
     for(int i=0; i<100000000; i++){
       int a =(int) Math.random();
     }
     long time2 = System.nanoTime()/1000000;
     long time = time2 - time1;
     System.out.println("waktunya : " + time +
     "ms");
  }}
```

Gambar 4. 1 *Source code* looping test menggunakan Math.Random

Penggunaan *System.nanoTime* pada variable time1, digunakan untuk mengambil waktu awal program dijalankan, dan pada time2 digunakan untuk mengambil waktu pada saat program berakhir. *Random generator* bawaan java tersebut di tes kecepatan kerjanya dengan me-*looping* random sebanyak 100 juta kali. Dan hasilnya adalah :



Gambar 4. 2 Hasil perhitungan waktu menggunakan Math.random

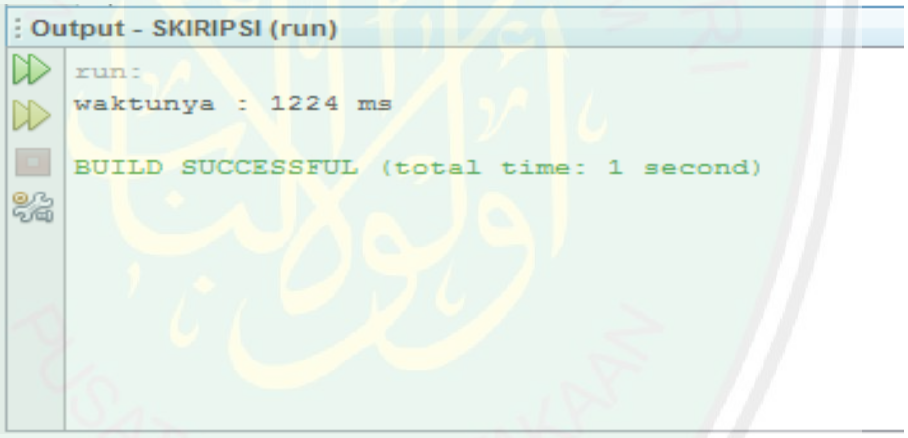
Dari uji coba tersebut dihasilkan waktu tempuh untuk melakukan perulangan yaitu selama 6.527 milidetik. Kemudian hasil tersebut dibandingkan dengan looping menggunakan MCRNG sebanyak 100 juta data juga. Kita buat kelas baru lagi untuk pengujian speed menggunakan MCRNG dengan *source code* sebagai berikut:

```
public class random_engine {
  public static void main(String[] args) {
     long time1 = System.nanoTime()/1000000;
     double seed = (double)(System.nanoTime()/1000000);

     for (int i = 0; i < 100000000; i++) {
           double xn = (77 * seed) % 127 ;
           seed = xn
     }
     long time2 = System.nanoTime()/1000000;
     long time = time2 - time1;
     System.out.println("waktunya : " + time + " ms");
```

Gambar 4. 3 *Source code* looping test menggunakan MCRNG

Didalam *code* tersebut, kita juga menggunakan *System.nanoTime* untuk perhitungan kecepatan. Akan tetapi *System.nanoTime* tersebut digunakan juga sebagai seed untuk MCRNG. Mengapa kita menggunakan *System.nanoTime* hal itu karena kita tidak bisa menebak secara pasti kapan waktu saat *System.nanoTime* tersebut dipanggil. Karena perhitungan System.nanoTime menggunakan satuan nano second.

Setelah *seed* didapat, maka kita looping perandoman menggunakan rumus MCRNG tersebut sebanyak 100 juta data. Dan hasil yang didapatkan yaitu :



Gambar 4. 4 Hasil perhitungan waktu menggunakan MCRNG

Hasilnya menunjukkan bahwa perulangannya berjalan hanya 1.224 milidetik. Dari hasil kedua uji coba tersebut, terbukti bahwa pengacakan angka menggunakan MCRNG hampir 5 kali lebih cepat dibandingkan menggunakan *Math.random*..

- **Hasil Pengujian Tahap II:**

Pengujian tahap kedua ini lebih diarahkan untuk pencarian nilai acuan(*a*) dan modulo(*m*) agar menghasilkan nilai yang tepat, dan juga kita lihat berapakah tingkat kesalahan pada penggunaan metode MCRNG ini. Untuk pengujian ini kita pilih beberapa nilai untuk nilai *a* dan *m* kemudian untuk nilai Z_0 atau *seed*-nya kita tetap memakai *System.nanoTime*. Pengacakan dilakukan sebanyak 10 kali dengan data yang di acak sebanyak 10 data.

Tabel 4. 1 Pengacakan dengan nilai *a* dan *m* bernilai genap

No	*a*	*m*	Z_0	Angka Random	Nilai	
1	2	6	84395	0.6667, 0.3333, 0.6667, 0.3333, 0.6667, 0.3333, 0.6667, 0.3333, 0.6667, 0.3333	7, 3, 7, 3, 7, 3, 7, 3	7, 3,
2	2	128	5002	0.625, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0	1, 3, 6, 2, 0, 0, 0, 0	5, 0,
3	4	64	77066	0.625, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0	6, 5, 0, 0, 0, 0, 0, 0	0, 0,
4	8	32	54705	0.25, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0	2, 0, 0, 0, 0, 0, 0, 0	0, 0,
5	14	32	57893	0.1875, 0.625, 0.75, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0	2, 6, 8, 5, 0, 0, 0, 0	0, 0,
6	28	44	86889	0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0	0, 0, 0, 0, 0, 0, 0, 0	0, 0,
7	32	128	22042	0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0	5, 0, 0, 0, 0, 0, 0, 0	0, 0,
8	72	1366	86182	0.5359, 0.5827, 0.9561, 0.8375, 0.2987, 0.5051, 0.369, 0.5652, 0.6911, 0.757	5, 6, 10, 9, 3, 5, 4, 6, 7, 8	
9	112	1128	27707	0.0496, 0.5603, 0.7518, 0.1986, 0.2411, 0.0071, 0.7943, 0.9645, 0.0284, 0.1773	0, 6, 8, 2, 2, 0, 8, 10, 0, 1	
10	256	1280	48049	0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8	8, 8, 8, 8, 8, 8, 8, 8, 8, 8	

Pada percobaan pertama yang kita lakukan, menghasilkan data seperti pada tabel diatas. Kita menggunakan *sample* data acuan(*a*) dan modulo(*m*) berupa nilai genap, dan sesuai dengan aturan MCRNG dimana:

m	*modulus*	$m > 0$
a	faktor pengali	$0 \leq a < m$
Z_0	angka permulaan (*seed*)	$0 \leq Z_0$

Data yang *error* atau gagal kita blok dengan warna merah. Merujuk pada tabel di atas, dapat kita lihat terdapat banyak data yang error atau gagal. Terhitung 8 dari sepuluh kali percobaan menghasilkan data yang kurang memuaskan atau bisa dikatakan *success rate*-nya hanya 20%. Jadi kita tarik kesimpulan bahwa bilangan genap kurang cocok jika dijadikan nilai *a* dan *m*.

Tabel 4. 2 Pengacakan dengan nilai a dan m bernilai ganjil

No	*a*	*m*	Z_0	Angka Random	Nilai
1	3	15	92632	0.4, 0.2, 0.6, 0.8, 0.4, 0.2, 0.6, 0.8, 0.4, 0.2	4, 2, 6, 8, 4, 2, 6, 8, 4, 2
2	3	1571	33438	0.8536, 0.5608, 0.6824, 0.0471, 0.1413, 0.4239, 0.2718, 0.8154, 0.4462, 0.3386	9, 6, 7, 0, 1, 4, 2, 8, 4, 3
3	7	9	10711	0.7778, 0.4444, 0.1111, 0.7778, 0.4444, 0.1111, 0.7778, 0.4444, 0.1111, 0.7778	8, 4, 1, 8, 4, 1, 8, 4, 1, 8
4	9	21	61	0.1429, 0.2857, 0.5714, 0.1429, 0.2857, 0.5714, 0.1429, 0.2857, 0.5714, 0.1429	1, 3, 6, 1, 3, 6, 1, 3, 6, 1
5	5	27	80932	0.4074, 0.037, 0.1852, 0.9259, 0.6296, 0.1481, 0.7407, 0.7037, 0.5185, 0.5926	4, 0, 2, 10, 6, 1, 8, 7, 5, 6
6	25	135	20374	0.963, 0.0741, 0.8519, 0.2963, 0.4074, 0.1852, 0.6296, 0.7407, 0.5185, 0.963	10, 0, 9, 3, 4, 2, 6, 8, 5, 10

7	13	39	54151	0.3333, 0.3333, 0.3333, 0.3333, 0.3333, 0.3333, 0.3333, 0.3333, 0.3333, 0.3333	3, 3, 3, 3, 3, 3, 3, 3, 3, 3
8	15	60	50854	0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5	5, 5, 5, 5, 5, 5, 5, 5, 5, 5
9	153	1235	59254	0.7789, 0.1789, 0.3789, 0.9789, 0.7789, 0.1789, 0.3789, 0.9789, 0.7789, 0.1789	8, 1, 4, 10, 8, 1, 4, 10, 8, 1
10	625	731	74148	0.0328, 0.5198, 0.8974, 0.8755, 0.1956, 0.264, 0.0137, 0.5499, 0.7073, 0.0315	0, 5, 9, 9, 2, 2, 0, 6, 7, 0

Karena pada percobaan pertama saat menggunakan angka genap banyak menemukan kesalahan, maka pada percobaan kedua ini kita mengambil nilai ganjil untuk dijadikan nilai *a* dan *m*. Hasil dari percobaan dapat kita lihat pada tabel di atas. Terdapat 5 kesalahan dari 10 kali percobaan. Itu berarti tingkat keberhasilan meningkat menjadi 40%. Sekarang kita lihat pada hasil yang gagal. Sekilas seperti tidak ada yang salah, tetapi jika kita lebih teliti lebih lanjut, maka akan terlihat pola terjadinya kesalahan. Kesalahan tersebut akan terjadi jika nilai *a* merupakan kelipatan dari nilai *m*. Oleh sebab itu kita harus menambahkan syarat lagi yaitu:

$$0 \leq a < m;\ m\%a \neq 0$$

Karena penggunaan bilangan ganjil masih menuai kesalahan, maka kita coba gabungan antara nilai ganjil dan genap.

Tabel 4. 3 Pengacakan dengan nilai a dan m bernilai ganjil atau genap

No	*a*	*m*	Z_0	Angka Random	Nilai
1	2	57	6165	0.3158, 0.6316, 0.2632, 0.5263, 0.0526, 0.1053, 0.2105, 0.4211, 0.8421, 0.6842	3, 6, 2, 5, 0, 1, 2, 4, 9, 7

2	4	7	89607	0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0	0, 0, 0, 0, 0, 0, 0, 0, 0, 0
3	6	13	68585	0.6154, 0.6923, 0.1538, 0.9231, 0.5385, 0.2308, 0.3846, 0.3077, 0.8462, 0.0769	6, 7, 1, 10, 5, 2, 4, 3, 9, 0
4	7	24	80938	0.9167, 0.4167, 0.9167, 0.4167, 0.9167, 0.4167, 0.9167, 0.4167, 0.9167, 0.4167	10, 4, 10, 4, 10, 4, 10, 4, 10, 4
5	9	14	47034	0.1429, 0.2857, 0.5714, 0.1429, 0.2857, 0.5714, 0.1429, 0.2857, 0.5714, 0.1429	1, 3, 6, 1, 3, 6, 1, 3, 6, 1
6	91	148	61537	0.9392, 0.4662, 0.4257, 0.7365, 0.0203, 0.8446, 0.8581, 0.0878, 0.9932, 0.3851	10, 5, 4, 8, 0, 9, 9, 0, 10, 4
7	12	1573	10868	0.0146, 0.1755, 0.1055, 0.2664, 0.1964, 0.3573, 0.2873, 0.4482, 0.3783, 0.5391	0, 1, 1, 2, 2, 3, 3, 4, 4, 5
8	11	1213	39937	0.2722, 0.9942, 0.9367, 0.3035, 0.338, 0.7179, 0.8972, 0.8692, 0.5617, 0.1785	2, 10, 10, 3, 3, 7, 9, 9, 6, 1
9	1462	12167	79765	0.6495, 0.5129, 0.9254, 0.8937, 0.6317, 0.5581, 0.014, 0.4274, 0.8377, 0.6818	7, 5, 10, 9, 6, 6, 0, 4, 9, 7
10	123	456	98756	0.1316, 0.1842, 0.6579, 0.9211, 0.2895, 0.6053, 0.4474, 0.0263, 0.2368, 0.1316	1, 2, 7, 10, 3, 6, 4, 0, 2, 1

Pada percobaan ketiga ini, tingkat kesalahan pada MCRNG ini dapat direduksi menjadi hanya 30%. Itu berarti dalam percobaan tersebut hanya terdapat 3 kesalahan saja dalam iterasinya. Jika kita tinjau kesalahan tersebut, kesalahan tersebut sering terjadi pada perhitungan dengan nilai *a* dan *m* yang rendah. Kemudian untuk selanjutnya, kita akan mencoba perhitungan dengan menggunakan nilai (*a*) adalah bilangan prima dan dengan modulo merupakan bilangan yang besar dalam hal ini kita akan menggunakan Short.MaxValue(2^{15}-1).

Tabel 4. 4 Pengacakan dengan nilai *a* bilangan prima dan m = 2^{15}-1

No	*a*	*m*	Z_0	Angka Random	Nilai
1	3	2^{15}-1	99370	0.0979, 0.2936, 0.8809, 0.6426, 0.9277, 0.7831, 0.3493, 0.0479, 0.1438, 0.4315	1, 3, 9, 7, 10, 8, 3, 0, 1, 4
2	5	2^{15}-1	67780	0.3427, 0.7136, 0.5681, 0.8404, 0.2018, 0.0089, 0.0447, 0.2235, 0.1177, 0.5887	3, 7, 6, 9, 2, 0, 0, 2, 1, 6
3	7	2^{15}-1	117	0.025, 0.175, 0.2247, 0.5732, 0.0122, 0.0852, 0.5967, 0.1767, 0.2367, 0.6569	0, 1, 2, 6, 0, 0, 6, 1, 2, 7
4	11	2^{15}-1	43817	0.7095, 0.8048, 0.8525, 0.3775, 0.153, 0.6829, 0.5117, 0.6287, 0.9162, 0.0778	7, 8, 9, 4, 1, 7, 5, 6, 10, 0
5	13	2^{15}-1	42282	0.775, 0.0748, 0.9728, 0.6465, 0.4046, 0.2592, 0.3695, 0.8037, 0.4486, 0.8313	8, 0, 10, 7, 4, 2, 4, 8, 4, 9
6	17	2^{15}-1	37170	0.2843, 0.8338, 0.1745, 0.9671, 0.4407, 0.4922, 0.3674, 0.2465, 0.191, 0.2467	3, 9, 1, 10, 4, 5, 4, 2, 2, 2
7	19	2^{15}-1	67044	0.8756, 0.6359, 0.0829, 0.576, 0.9447, 0.9493, 0.0369, 0.7005, 0.3088, 0.8664	9, 6, 0, 6, 10, 10, 0, 7, 3, 9
8	23	2^{15}-1	3660	0.569, 0.0881, 0.0265, 0.6086, 0.9971, 0.9333, 0.4663, 0.7247, 0.6686, 0.3785	6, 0, 0, 6, 10, 10, 5, 7, 7, 4
9	29	2^{15}-1	58583	0.8481, 0.5952, 0.26, 0.5396, 0.6492, 0.8274, 0.9942, 0.8327, 0.1491, 0.3243	9, 6, 2, 5, 7, 9, 10, 9, 1, 3
10	31	2^{15}-1	73243	0.2933, 0.0918, 0.8448, 0.1902, 0.895, 0.7446, 0.0814, 0.5222, 0.1892, 0.8657	3, 1, 9, 2, 9, 8, 0, 5, 2, 9

Pada percobaan keempat ini, kita dapat lihat bahwa sudah tidak ada lagi data yang *error* atau kurang sempurna. Hal ini menandakan bahwa dengan acuan menggunakan bilangan prima, dan modulo yang digunakan adalah bilangan yang besar, maka hasil yang didapatkan dari pengacakan bisa dikatakan sempurna. Oleh karena itu, pada aplikasi yang akan kita buat kita akan menggunakan pengacakan MCRNG dengan aturan sebagai berikut:

Pseudo-random yang digunakan:

$$Z_i = (aZ_{i-1}) \bmod m;$$

$Z_0 = Seed = System.nanoTime;$

$0<a<m;$ *a* adalah bilangan *prima;*

$m = SHORT.MaxValue;$

4.1.3. Implementasi MCRNG

Setelah uji coba yang dilakukan diatas, kita mendapatkan nilai acuan(*a*). kemudian kita implementasikan *source code* MCRNG tersebut dalam *source* game dengan membuat class baru MCRNG(). Kita akan menggunakan data dari hasil uji coba, dimana nilai *a* yang kita gunakan yaitu bilangan prima, *seed* menggunakan *System.nanoTime*, dan modulo menggunakan *Short.MaxValue.*

```
public class MCRNG {
    ArrayList<Double> array = new ArrayList<Double>();
    public void random() {
        int z = (int) (System.nanoTime() / 1000000);
        String c1 = Integer.toString(z);
        String c2 = c1.substring(c1.length() - 5, c1.length());
        String c3 = c1.substring(c1.length() - 2, c1.length());
        int seed = Integer.parseInt(c2);
        int length = Integer.parseInt(c3);
        int m = Short.MAX_VALUE;
        for (int i = 0; i < length; i++) {
                int xn = (31 * seed) % m;
                double r = (double) (xn) / m;
                array.add(r);
                seed = xn;
        }
    }

    public int nextInt(int bb, int ba) {
        if (array.isEmpty()) {
                random();
        }
        Iterator<Double> i = array.iterator();
        if (i.hasNext()) {
                int nilai = bb + (int) (i.next() * (ba - bb));
                i.remove();
                return nilai;
        }
        return 0;
    }
}
```

Gambar 4. 5 *Class* MCRNG

Pada Class MCRNG tersebut, fungsi List *array* adalah untuk menampung hasil looping MCRNG (dalam *source* tersebut, panjang *array* diambil dari 2 angka terakhir dari *System.nanoTime*). Setelah *array* tersebut terisi dengan data random, data-data tersebut digunakan untuk membangkitkan suatu bilangan saat fungsi *nextInt* dipanggil dengan batasan nilai dari batas bawah (*bb*) sampai batas atas (*ba*). Ibarat kita menembakkan 50 peluru,dengan sebuah revolver dan AK-47 maka pasti akan lebih cepat menembakkan AK-47 karena AK-47 mempunyai

magazine lebih banyak dari sebuah revolver dan dapat langsung menembakkan 50 peluru tersebut dibandingkan revolver yang harus me reload setiap menembakkan 5 peluru. Seperti halnya pada persoalan ini, dimana List *array* berfungsi sebagai magazine dari generator dan disaat *array* tersebut kehabisan data, maka akan mereload datanya kembali.

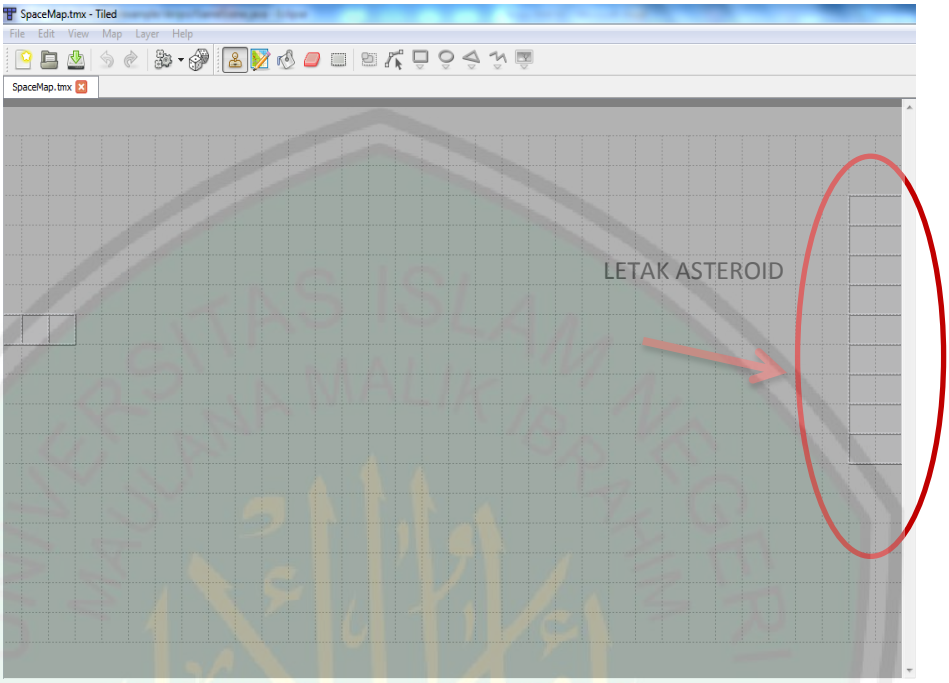
Dari *class* MCRNG tersebut fungsi *nextInt* dipanggil pada saat pembuatan asteroid di dalam fungsi *createAsteroid*.

```
MCRNG random = new MCRNG()
public void createAsteroid(final TMXTiledMap map) {
   for (final TMXObjectGroup group : map.getTMXObjectGroups()) {
    if (group.getName().equals("Asteroid")) {
      asteroids = new ArrayList<Sprite>();
      for (TMXObject object : group.getTMXObjects()) {
       asteroids.add(makingAsteroid(object.getX(), object.getY()));
      }
      asteroid = asteroids.get(random.nextInt(0,asteroids.size()));

      int minDuration = Setting.getMinSpeed();
      int maxDuration = Setting.getMaxSpeed();
      int actualDuration = random.nextInt(minDuration, maxDuration);

      MoveXModifier asteroid_mod = new MoveXModifier (actualDuration,
                                   asteroid.getX() +
                                   asteroid.getWidth(),
                                   -asteroid.getWidth());

      asteroid.registerEntityModifier(asteroid_mod);
      resourcesManager.listAsteroidAdded.add(asteroid);
      attachChild(asteroid);
      }
   }
}
```

Gambar 4. 6 Method Create Asteroid dengan MCRNG

Fungsi *nextInt* tersebut digunakan untuk membangkitkan bilangan acak dari 0 sampai besarnya list *asteroids. Asteroids* adalah sekumpulan

object lokasi yang telah dibuat pada TMXTiledMap. Dari lokasi object inilah nantinya akan muncul sebuah asteroid.



Gambar 4. 7 Inisialisasi letak asteroid di TMX map

Pada gambar 4.5 terlihat letak kemunculan asteroid yang sudah ditentukan berada di sebelah kanan layar. Posisi-posisi tersebut disimpan dalam *array* posisi yang nantinya di munculkan secara random berdasarkan indeksnya. Selain itu, fungsi *nextInt* juga digunakan untuk mengacak kecepatan asteroid itu sendiri. Kecepatan asteroid itu sendiri memiliki rentang yang bervariasi tergantung tingkat kesulitan yang di pilih. Batas bawah dan batas atas kecepatan tersebut diambilkan dari *class Setting* menurut *source code* pada gambar 4.4

Selain untuk pengacakan lokasi kemunculan dan pengacakan kecepatan asteroid, fungsi *nextInt* pada *class* MCRNG tersebut dapat digunakan untuk pengacakan pertanyaan dan operator bilangan pada

permainan ini. Pengacakan tersebut dipanggil pada method making asteroid, seperti yang ada pada *source code* di bawah ini.

```
public Sprite makingAsteroid(final int pX, final int pY) {

// ------ Make Question------//
   ChangeableText Question = new ChangeableText(25, 23,
                             resourcesManager.game_font1, "XXX");
   final int nilai1 = random.nextInt(0,10);
   final int nilai2 = random.nextInt(0,10);
   final int r = random.nextInt(0,Setting.getOperator());
   final int hasil = Answering(nilai1, nilai2, r);
   switch (r) {
   case 0:
           Question.setText(Integer.toString(nilai1) + "+" +
                             Integer.toString(nilai2));
       break;
   case 1:
           Question.setText(Integer.toString(nilai1) + "-" +
                             Integer.toString(nilai2));
       break;
   case 2:
           Question.setText(Integer.toString(nilai1) + "*" +
                             Integer.toString(nilai2));

       break;
   }
```

Gambar 4. 8 Pengacakan pertanyaan dalam *source code* making asteroid

Hasil dari kemunculan asteroid tersebut dalam beberapa kali dapat dilihat pada beberapa *screenshoot* di bawah ini.

Gambar 4. 9 Kemunculan acak asteroid screenshoot pertama

Gambar 4. 10 Kemunculan acak asteroid screenshoot kedua

Gambar 4. 11 Kemunculan acak asteroid screenshoot ketiga

4.1.4. Pengujian AStar

Metode selanjutnya yang akan diuji ialah AStar. Dimana metode ini digunakan untuk *tracking* jalur *projectile* menuju asteroid. Pada dasarnya Astar adalah algoritma *path finding* dengan cara melakukan pencarian nilai heuristik terkecil dari setiap node yang ada di sekitar. Object node tersebut kita implementasikan dalam suatu class bernama *Node.class*

```
public class Node {
    int x;
    int y;
    Node parent;

    public Node(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public void SetParent(Node n) {
        this.parent = n;
    }

    public Node getParent() {
        return parent;
    }
    public boolean hasParent(){
        if (parent != null) {
            return true;
        }
        return false;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public void setX(int pX){
        this.x = pX;
    }

    public void setY(int pY){
        this.y = pY;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj instanceof Node) {
            Node n = (Node) obj;
            if (n.getX() == this.getX() && n.getY() ==
                this.getY()) {
                return true;
            } else {
                return false;
            }
        } else {
            return false;
        }}
}
```

Gambar 4. 12 Node Class untuk metode AStar

Kelas *Node* tersebut digunakan untuk merepresentasikan posisi target dan posisi object yang digerakkan pada saat ini. Di dalam kelas *Node* tersebut juga memiliki object bernama parent yang menunjukkan bahwa node tersebut datang dari node mana.

Sedangkan pengujian Astar ini dilakukan menggunakan IDE Netbeans 7.1 dengan *source code* sebagai berikut:

```
public class AStarTest {
    double nilaiG(Node node) {
      double c2 = Math.pow(node.getX(), 2) +
                  Math.pow(node.getY(), 2);
      double x = Math.sqrt(c2);
         return x;
    }

    double nilaiH(Node awal, Node target) {
        return Math.abs(target.getX() - awal.getX())
                + Math.abs(target.getY() -
               awal.getY());
    }

    double nilaiF(Node awal, Node target) {
        return nilaiG(awal) + nilaiH(awal, target);
    }

    public ArrayList<Node> ListNeighborNodes(Node
    currentNode) {
        ArrayList<Node> sekitar = new ArrayList<>();
        sekitar.clear();
        for (int dX = -1; dX <= 1; dX++) {
            for (int dY = -1; dY <= 1; dY++) {
                if ((dX == 0) && (dY == 0)) {
                    continue;
                }
                Node neighborNode = new Node(dX +
                currentNode.getX(), dY +
                currentNode.getY());
                sekitar.add(neighborNode);
            }
        }
```

```
    public Node nextNode(Node current, Node target) {
        Node best = current;
        ArrayList<Node> neighbor = ListNeighborNodes(
                                    current);

        double min = nilaiF(neighbor.get(0), target);
        for (Node node : neighbor) {
            if(best.hasParent()){
                if(node.equals(best.parent)){
                    continue;
                }
            }
            double g = nilaiG(node)*10;
            double h = nilaiH(node, target)*10;
            double f = nilaiF(node, target)*10;
            if(nilaiF(node, target)<=min){
                min = nilaiF(node, target);
                best = node;
            }
        }
        System.out.println("nilai G minimum: " +
                           nilaiG(best));
        System.out.println("nilai H minimum: " +
                           nilaiH(best, target));
        System.out.println("nilai F minimum: " +
                           nilaiF(best, target));
        return best;
    }
}
```

Gambar 4. 13 Implementasi AStar dalam *source code*

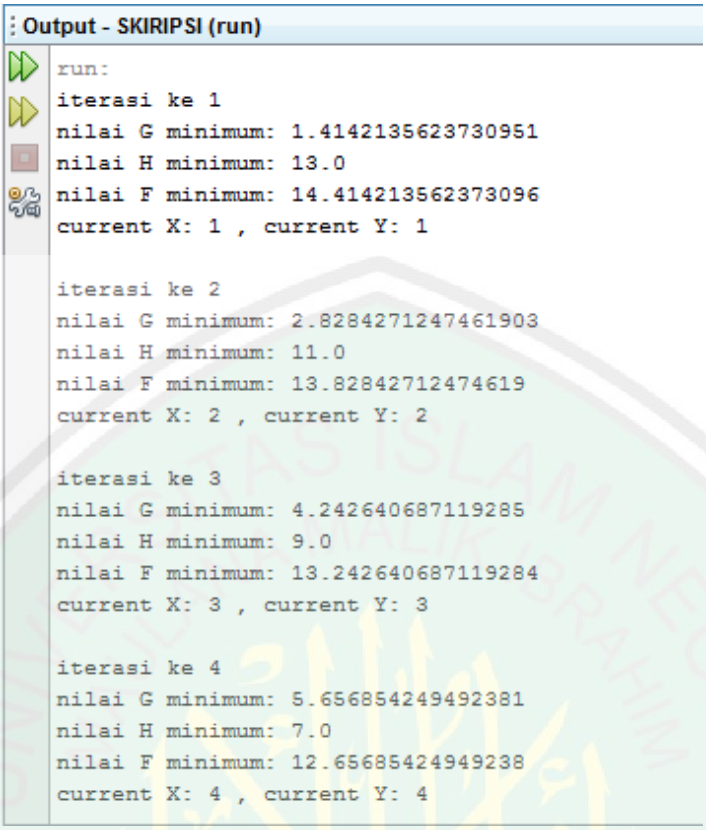
Pada *Astar.class* seperti gambar 4.13, langkah pertama yaitu mencari node-node di sekitar current node kemudian kita tampung semua tersebut kedalam suatu *array*. Setelah ditampung maka kita hitung nilai F untuk setiap node yang ada dalam *array* sekitar tersebut dan kita pilih node yang memiliki nilai F terkecil sebagai *best node.* Kemudian kita pindahkan *current node* menuju ke *best node.* Lakukan hal tersebut berulang-ulang hingga nilai *current node* sama dengan nilai *target node.*

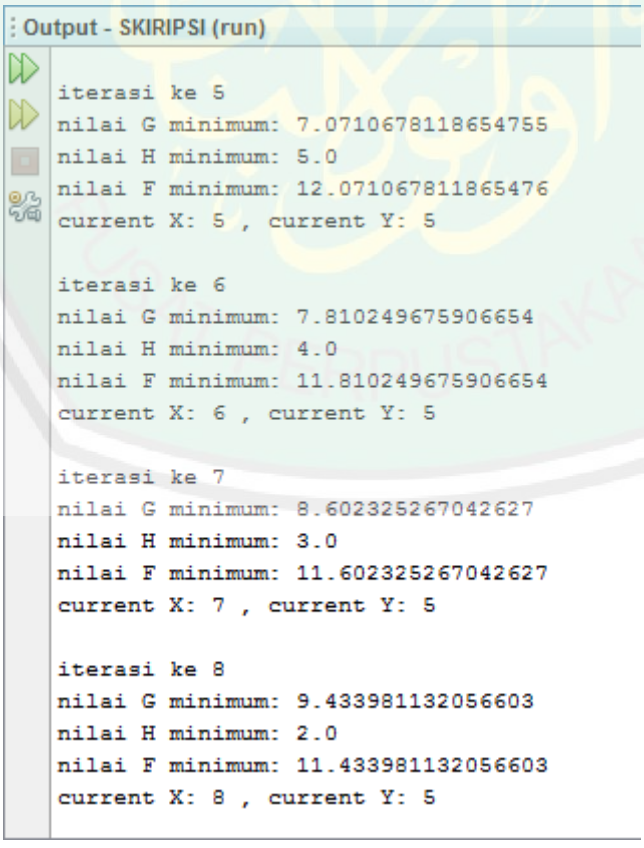
Untuk percobaannya, kita siap kan terlebih dahulu nilai *current node* dan *target node* yang akan digunakan misalkan koordinat current node yaitu (0,0) dan koordinat target node yaitu (10,5) seperti pada *source code* di bawah ini:

```
public static void main(String[] args) {
   AStarTest astar = new AStarTest();
   Node a = new Node(0,0);
   Node target = new Node(10, 5);
   Node temp;
   while (!a.equals(target)) {
      temp = astar.nextNode(a, target);
      temp.SetParent(a);
      a = temp;
      System.out.print("current X: "+a.getX()+" ,
      current X: "+a.getY());
      System.out.println();
      System.out.println();
```

Gambar 4. 14 Void Main Uji coba AStar dengan Netbeans

Kemudian kita looping hingga nilai *current node* sama dengan *target node*. Saat memindahkan *current node* menuju *next node*, kita set parent dari next node tersebut adalah current node agar pada saat penghitungan nilai F, nilai parent tidak usah dihitung lagi agar dapat lebih optimal.Maka output yang kita dapatkan adalah seperti yang ada di bawah ini.





```
iterasi ke 9
nilai G minimum: 10.295630140987
nilai H minimum: 1.0
nilai F minimum: 11.295630140987
current X: 9 , current Y: 5

iterasi ke 10
nilai G minimum: 11.180339887498949
nilai H minimum: 0.0
nilai F minimum: 11.180339887498949
current X: 10 , current Y: 5

BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 4. 15 Hasil Uji coba menggunakan AStar

Dari data diatas sudah dapat terlihat bahwa *current node* berhasil menuju *target node* dalam 10 iterasi dengan menggunakan nilai F terkecil. Pada iterasi pertama, *current node* menghitung nilai G, H, dan F node-node disekitarnya. Hasilnya, dipilih sebuah node yang berada pada koordinat (1,1) dengan nilai F paling minimum bernilai 14,414. Kemudian pindahkan current node menuju node tersebut, lalu hitung kembali nilai G, H, dan F pada lokasi *current node* yang baru. Didapatkanlah koordinat (2,2) dengan nilai F 13,828 untuk node selanjutnya. Pindahkan lagi *current node* menuju node tersebut, dan ulangi terus prosesnya hingga koordinat *current node* sama dengan koordinat *target node*.

Kita telah mengetahui bahwa sebuah node dapat berpindah menuju posisi target yang dia inginkan dengan cara seperti diatas, kemudian sekarang saatnya mengaplikasikan metode tersebut ke dalam permainan yang akan kita buat.

4.1.5. Implementasi AStar

Setelah berhasil dalam uji coba yang kita lakukan sebelumnya, Astar akan diterapkan pada object *Projectile* atau peluru dimana peluru ini akan membidik asteroid yang dipilih dan menghancurkannya setelah bertabrakan. Seperti pada *source code* dibawah ini.

```
private AStar aStar = new AStar();

currentNode = new Node(projectile.getX(), projectile.getY());
targetNode = new Node(touch_x, touch_y);

projectile.registerUpdateHandler(new TimerHandler(0.005f,
                                 true, new ITimerCallback() {

    @Override
    public void onTimePassed(TimerHandler pTimerHandler) {

       if(currentNode.equals(targetNode)) {
          projectile.unregisterUpdateHandler(pTimerHandler);
          resourcesManager.activity.runOnUpdateThread(new
                                    Runnable() {

             @Override
             public void run() {
                   detachChild(projectile);
             }
          });
       }else{
          float xawal = currentNode.getX();
          float yawal = currentNode.getY();

          temp = aStar.nextNode(currentNode, targetNode);
          temp.SetParent(currentNode);
          currentNode = temp;

          float xnext = currentNode.getX();
          float ynext = currentNode.getY();

          projectile.setPosition(xnext,ynext);
          double tan= (ynext-yawal)/(xnext-xawal);
          float atan = (float)(Math.atan(tan)*180/Math.PI);
          projectile.setRotation(atan);
       }}
}));
```

Gambar 4. 16 Penggunaan class Astar dalam *source code* game

Berdasarkan kode program di atas, pertama kita menginisialisasikan kelas AStar terlebih dahulu. Kemudian kita ambil posisi projectile saat ini sebagai posisi *current node*. Kemudian untuk *target node*, kita ambil dari koordinat *touch screen* pada *asteroid*. Setelah itu, untuk setiap 0.005 detik, kita deteksi apakah *current node* tersebut telah mencapai *target node*? Jika sudah mencapai target, kita *remove projectile* tersebut. Tetapi jika belum, kita inisialisasikan x awal dan y awal adalah x dan y *current node* kemudian kita pindahkan *current node* ke *next node*. Dan kemudian kita simpan x dan y nya sebagai x *next* dan y *next*. Setelah itu baru kita ubah posisi projectile dengan menggunakan fungsi *setPosition*. Untuk *heading rotation* kita menggunakan nilai `arces tan` untuk mendapatkan nilai sudut.

Hasilnya setelah dijalan kan seperti dibawah ini:

Gambar 4. 17 Posisi projectile menuju asteroid sebelum bertabrakan

Gambar 4. 18 Projectile mencapai target dan bertabrakan

4.2. Hasil Akhir

Setelah uji coba dan penelitian yang dilakukan oleh peneliti, akhirnya aplikasi asteroid shooter ini dapat diselesaikan dengan baik dan sempurna. Hasil tersebut dapat dilihat dari screenshoot dibawah ini.

- **Splash Screen :**

Gambar 4. 19 Tampilan SplashScreen

Diatas adalah tampilan splash dari aplikasi. Splash tersebut akan muncul selama ± 5-6 detik.

- **Main Menu :**

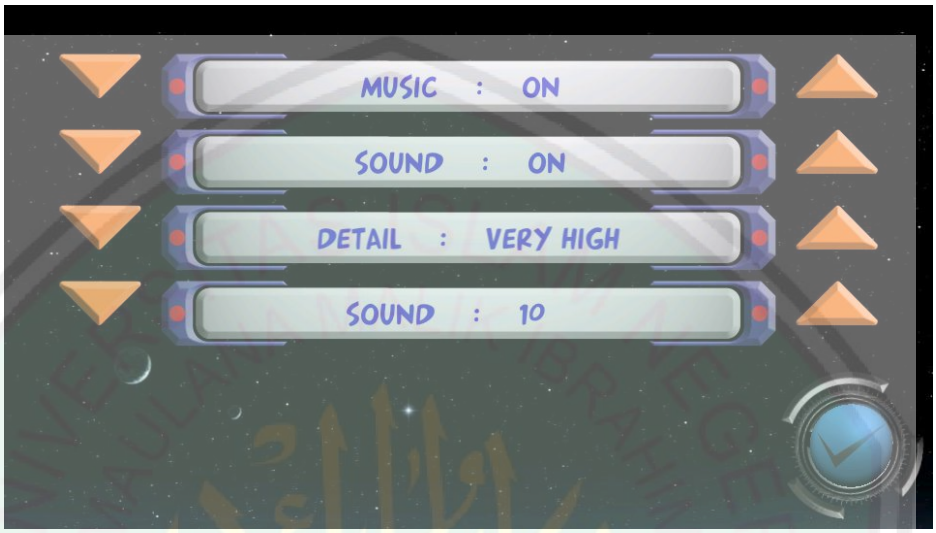
Gambar 4. 20 Tampilan MainMenu

Diatas merupakan tampilan main menu dimana terdapat pilihan, yaitu bermain, pengaturan atau keluar dari *system*

Gambar 4. 21 Tampilan Difficulty Menu

Sedangkan jika kita memilih untuk bermain, akan muncul pilihan tingkat kesulitan seperti gambar diatas.

- **Options :**



Gambar 4. 22 Tampilan Options

Menu pada layar options terdiri atas pengaktifan music background, efek suara, ketajaman gambar dan tingkat volume.

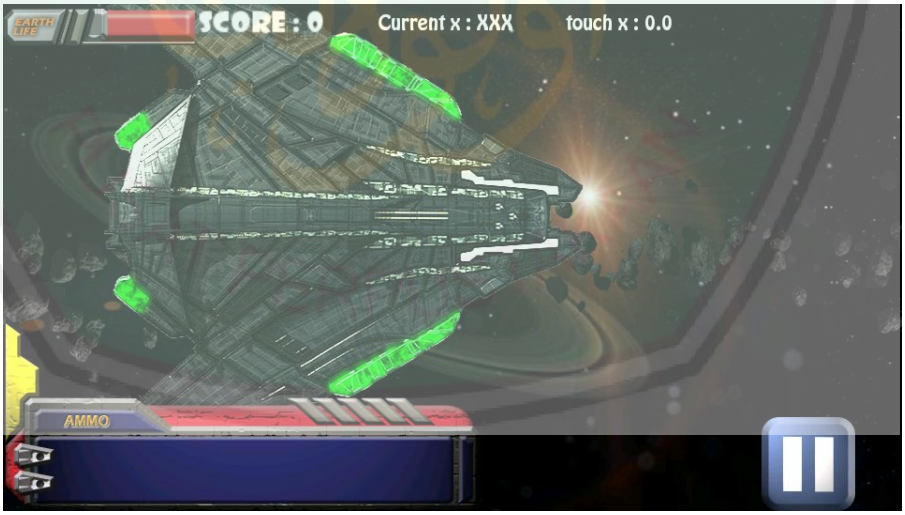
- **Loading Screen :**

Gambar 4. 23 Tampilan Loading Screen 1

Gambar 4. 24 Tampilan Loading Screen 2

Loading screen terdiri dari dua fase yang juga merangkap sebagai tutorial untuk bermain.

- **Game :**



Gambar 4. 25 Tampilan Intro Game

Intro permainan akan berlangsung selama ± 10 detik.

Gambar 4. 26 Tampilan Ketika asteroid muncul

Disini kemunculan asteroid menggunakan MCRNG sehingga posisi kemunculannya acak dan tidak bisa ditebak.

Gambar 4. 27 Tampilan ketika menjawab dengan benar pertanyaan yang ada

Ketika berhasil menjawab pertanyaan, maka akan muncul rudal yang mengejar target. Disini metode Astar berlangsung.

Gambar 4. 28 Tampilan ketika rudal mengenai asteroid dan meledakkannya

Ketika rudal mengenai target, maka akan meledakkan target tersebut dan poin kita akan bertambah.

Gambar 4. 29 Tampilan On Game Pause Menu

Jika kita pencet tombol paused di pojok kanan bawah, atau kita pencet tombol back, maka akan keluar tampilan *pause menu*.

Gambar 4. 30 Tampilan ketika asteroid berhasil melewati *player*

Ketika asteroid dapat melewati *player* akan mengurangi nyawa *player*.

Gambar 4. 31 Tampilan system saat game over

Ketika nyawa *player* telah habis, maka akan game over dan muncul pilihan untuk *retry* atau kembali ke main menu.

4.3. Integrasi Aplikasi dengan Islam

Dalam perspektif agama islam, belajar merupakan kewajiban bagi setiap muslim dalam rangka memperoleh ilmu pengetahuan sehingga derajat kehidupannya meningkat. Hal ini dinyatakan dalam surah Mujadalah: 11

يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُوٓاْ إِذَا قِيلَ لَكُمۡ تَفَسَّحُواْ فِي ٱلۡمَجَٰلِسِ فَٱفۡسَحُواْ يَفۡسَحِ
ٱللَّهُ لَكُمۡۖ وَإِذَا قِيلَ ٱنشُزُواْ فَٱنشُزُواْ يَرۡفَعِ ٱللَّهُ ٱلَّذِينَ ءَامَنُواْ مِنكُمۡ وَٱلَّذِينَ أُوتُواْ
ٱلۡعِلۡمَ دَرَجَٰتٖۚ وَٱللَّهُ بِمَا تَعۡمَلُونَ خَبِيرٞ ١١

Hai orang-orang beriman apabila kamu dikatakan kepadamu: "Berlapang-lapanglah dalam majlis", Maka lapangkanlah niscaya Allah akan memberi kelapangan untukmu. dan apabila dikatakan: "Berdirilah kamu", Maka berdirilah, niscaya Allah akan meninggikan orang-orang yang beriman di antaramu dan orang-orang yang diberi ilmu pengetahuan beberapa derajat. dan Allah Maha mengetahui apa yang kamu kerjakan.(QS: Al- Mujaadilah, 11). (Muhibbin, 2010)

Ilmu dalam hal ini tentu saja harus berupa pengetahuan yang relevan dengan tuntutan zaman dan bermanfaat bagi kehidupan orang banyak.

Agaknya tidak ada satu pun agama, termasuk Islam, yang menjelaskan secara rinci dan operasional mengenai proses belajar, proses kerja sistem memori (akal), dan proses dikuasainya pengetahuan dan ketrampilan oleh manusia. Namun Islam, dalam hal penekanannya terhadap signifikansi

fungsi kognitif (akal) dan fungsi sensori (indera-indera) sebagai alat-alat penting untuk belajar, sangat jelas. Kata-kata kunci, seperti *ya'qulun, yatafakkarun, yubshirun, yasma'un,* dan sebagainya yang terdapat dalam Al-Qura'an, merupakan bukti betapa pentingnya penggunaan fungsi ranah cipta dan karsa manusia dalam belajar dan meraih ilmu pengetahuan.

Tuhan memberikan potensi kepada manusia yang bersifat jasmaniah dan rohaniah untuk belajar dan mengembangkan ilmu pengetahuan dan teknologi untuk kemaslahatan umat manusia itu sendiri. Potensi-potensi tersebut terdapat dalam organ-organ fisio-psikis manusia yang berfunsi sebagai alat-alat penting untuk melakukan kegiatan belajar, diantaranya adalah sebagai berikut :

- Indera penglihat (mata), yakni alat fisik yang berguna untuk menerima informasi visual.
- Indera pendengar (telinga), yakni alat fisik yang berguna untuk menerima informasi verbal.
- Akal, yakni potensi kejiwaan manusia berupa sistem psikis yang kompleks untuk menyerap, mengolah, menyimpan, dan memproduksi kembali item-item informasi dan pengetahuan (ranah kognitif).

Dari pemaparan akan pentingnya belajar diatas, maka dapat diambil kesimpulan bahwa aplikasi yang telah dibuat ini, dapat digunakan seseorang untuk membantunya dalam belajar ilmu pengetahuan dalam hal ini yaitu ilmu pengetahuan tentang matematika. Sehingga kita dapat mengamalkan apa yang ada dalam surat Al-Mujaadilah ayat 11 tersebut.

BAB V

PENUTUP

2.1. Kesimpulan

Dari hasil implementasi dan uji coba yang dilakukan oleh peneliti, kita dapat menarik kesimpulan bahwa algoritma MCRNG dan AStar merupakan sebuah algoritma yang dapat diaplikasikan pada sebuah game dengan baik dan optimal. Algoritma MCRNG tersebut akan dapat berjalan dengan baik jika kita menggunakan nilai acuan (a) dengan bilangan prima dan modulo(m) adalah bilangan yang lebih besar ±1000 keatas dan bukan merupakan kelipatan dari acuan(a). MCRNG sendiri jika dibandingkan dengan random generator bawaan java, memiliki keunggulan dalam kecepatan berpikir.

2.2. Saran

Tentunya masih banyak kekurangan dalam penelitian dan pembuatan aplikasi permainan *Asteroid Shooter* ini. Oleh karena itu, sang penulis menyarankan untuk pengembang selanjutnya, diantaranya:

1. Kembangkanlah aplikasi Asteroid Shooter ini dengan lebih menarik, seperti dengan penambahan level atau musuh agar permainan ini lebih disukai tidak hanya oleh anak-anak, akan tetapi orang dewasa juga.
2. Kembangkanlah aplikasi ini dengan device-device mobile yang lain seperti pada iOS, Windows Phone, Blackberry, ataupun yang lainnya.
3. Tidak menutup kemungkinan untuk mengembangkan permainan ini dalam versi 3D nya sehingga lebih menantang dan menarik.

DAFTAR PUSTAKA

Buxton, Laurie. 1984. *Mathematics for Everyman.* Littlehampton Book Services Ltd. Inggris.

Dienes, Paul Dienes. *I Will Tell You Algebra Stories You've Never Heard Before*. Upfront Publishing Ltd. Inggris.

Hidayat, Tadya Rahanady. 2010. *Random Number Generator,* Makalah IF2091 Struktur Diskrit – Sem. I Tahun 2010/2011. Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Jl. Ganesha 10 Bandung, Indonesia.

Huda, Miftakhul. 2012. *Game Santri Story untuk Pengenalan Huruf Hijaiyah Menggunakan Metode MCRN-Generator*. *Skripsi*. Jurusan Teknik Informatika Fakultas Sains dan Teknologi UIN Maulana Malik Ibrahim. Malang.

L, Riani. 2010. *Pembangkit Bilangan Acak*. Mata Kuliah Pemodelan & Simulasi. Jurusan Teknik Informatika Universitas Komputer Indonesia. Bandung.

Muhibbin Syah, 2010. Psikologi Pendidikan dengan Pendekatan Baru. Bandung: PT. Remaja Rosdakarya.

Neumann, John Von. 2007. *Theory of Games and Economic Behavior*. Princeton University Press, USA

Ramadhani, Aristam. 2008, Menggerakkan Karakter Game Menggunakan Algoritma *Breadth-First Search* (BFS) dan Algortma Algoritma A*(A Star). Makalah IF2251 Strategi Algoritmik Tahun 2008. Program Studi

Teknik Informatika Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Jalan Ganesha 10, Bandung, Indonesia.

Russell, Stuart and Norvig, Peter. 2003. *Artificial Intelligence A Modern Approach.* Pearson Education, Inc. Upper Saddle River, New Jersey 07458

Safaat H, Nazruddin. 2011. *Android, Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android*. Informatika. Bandung.

Skinner, B. F. (1938). *The Behavior of Organisms: An Experimental Analysis*. New York: Appleton-Century.

Tofin, Misbakhul. 2012 . *Aplikasi Permainan Sudoku huruf hijaiyah Menggunakan Algoritma Backtracking dan Multiplicative CRNG Sebagai Pembangkit Dan Penyelesai Permainan*. *Skripsi*. Jurusan Teknik Informatika Fakultas Sains dan Teknologi UIN Maulana Malik Ibrahim. Malang.

Zeeb, Charles N. 1984. *Random Number Generator Recommendation*. Department of Mechanical Engineering Colorado State University. Colorado.