# RANCANG BANGUN *GAME* SEBAGAI TERAPI *COGNITIVE BEHAVIOUR* PADA PENDERITA *ATTENTION DEFICIT HYPERACTIVITY DISORDER* (ADHD) MENGGUNAKAN *KINECT*

## SKRIPSI

Untuk memenuhi sebagian persyaratan untuk mencapai gelar Sarjana Komputer


Disusun oleh :

**IKA KUSUMANING PUTRI**

**NIM. 0910680070**

**KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN**

**UNIVERSITAS BRAWIJAYA**

**PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER**

**MALANG**

**2013**

**LEMBAR PERSETUJUAN**

**RANCANG BANGUN *GAME* SEBAGAI TERAPI *COGNITIVE BEHAVIOUR* PADA PENDERITA *ATTENTION DEFICIT HYPERACTIVITY DISORDER* (ADHD) MENGGUNAKAN *KINECT***

# SKRIPSI

Untuk memenuhi sebagian persyaratan untuk mencapai gelar Sarjana Komputer


Disusun oleh :

**IKA KUSUMANING PUTRI**

**NIM. 0910680070**

Telah diperiksa dan disetujui oleh:

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| **Eriq M. Adams J., ST.M.Kom** | **Aditya Rachmadi, S.ST,.MTI** |
| **NIK. 850410 06 1 10027** | **NIK. 860421 16 1 10426** |

**LEMBAR PENGESAHAN**

**RANCANG BANGUN *GAME* SEBAGAI TERAPI *COGNITIVE BEHAVIOUR* PADA PENDERITA *ATTENTION DEFICIT HYPERACTIVITY DISORDER* (ADHD) MENGGUNAKAN *KINECT***

**SKRIPSI**

Untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun oleh :

**IKA KUSUMANING PUTRI**

**NIM. 0910680070**

Skripsi ini telah diuji dan dinyatakan lulus pada

Tanggal 2 Januari 2014

| Penguji I | Penguji II |
|---|---|
| **Wibisono Sukmo Wardhono, S.T., M.T.** | **Denny Sagita Rusdianto, S.Kom., M.Kom** |
| **NIK. 820404 06 1 1 0091** | **NIK. 851124 06 1 1 0250** |

Penguji III

**Aryo Pinandito, S.T., M.MT.**

**NIK. 830519 16 1 1 0374**

Mengetahui

Ketua Program Studi Teknik Informatika

**Drs. Marji, M.T.**

**NIP. 19670801 199203 1 001**

**PERNYATAAN**
**ORISINALITAS SKRIPSI**

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, Januari 2014

Mahasiswa,

**Ika Kusumaning Putri**

**NIM 0910680070**

# PRAKATA

Puji syukur saya ucapkan kepada Allah SWT yang senantiasa melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Proposal Skripsi yang berjudul, "Rancang Bangun *Game* sebagai Terapi *Cognitive Behaviour* pada Penderita *Attention Deficit Hyperactivity Disorder* (ADHD) menggunakan *Kinect*".

Skripsi ini diajukan sebagai salah satu syarat untuk mendapatkan gelar sarjana S-1 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya. Penyusun menyelesaikan Skripsi ini berdasarkan teori-teori yang telah di peroleh dalam perkuliahan, literatur dari beberapa buku dan paper dan bimbingan dari dosen pembimbing serta pihak-pihak lain yang telah banyak memberikan semangat dan bantuan. Penyusun ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Kedua orang tua penulis (Sujadi dan Endah Dwi Ariani) yang senantiasa memberikan dukungan, kasih sayang dan do'a demi terselesaikannya proposal skripsi ini.
2. Bapak Drs. Marji, M.Si. dan Issa Arwani, ST., MT. selaku Ketua dan Sekretaris Program Studi Informatika serta segenap Bapak/Ibu Dosen, Staff Administrasi dan Perpustakaan Program Studi Informatika Universitas Brawijaya.
3. Bapak Eriq M. Adams J. ST,M.Kom dan Bapak Aditya Rachmadi, S.ST,.MTI. selaku dosen pembimbing Penulis. Terima kasih atas semua bimbingan dan dorongan semangatnya.
4. Seluruh Dosen Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya yang telah membekali penulis dengan ilmu – ilmu yang bermanfaat.
5. Teman – teman tim Raion's Head Gemastik 6 (Hanas Subakti, S.Kom., Dwi Hardyanto, S.Kom. dan Izhar Priandana Saputra, S.Sn.) yang telah membantu penulis dalam pembuatan *asset* gambar dan suara pada pembuatan skripsi ini.

6. Teman – teman Herky's (Eka Sagita Putri, S.Psi dan Aprilia Indra Cahya, A.Md) yang telah memberikan inspirasi dalam pembuatan ide skripsi ini.
7. Ibu Elisabet Selaku kepala sekolah "Zero to Five" tempat penulis melakukan penelitian pada anak berkebutuhan khusus
8. Teman – teman siswa "Zero to Five" (Kid, Nathan, Ben dan Lius) yang telah membantu penulis dalam penelitian dengan mengikuti terapi.
9. Kakak – kakak angkatan 2007 dan 2008 yang telah berbagi referensi kepada penulis.
10. Teman – teman angkatan 2009 yang telah berbagi ilmu, keluh kesah dan semangat dengan penulis.
11. Teman – teman Raion Studio yang telah berbagi ilmu dan pengalaman dengan penulis.
12. Serta semua pihak yang namanya tidak bisa penulis sebutkan satu persatu. Terima kasih atas do'a dan dukungannya.

Penyusun sadar bahwa masih banyak kesalahan dan kekurangan dalam penyusunan Skripsi ini, untuk itu penyusun mohon maaf dan mengharapkan kritik dan saran guna penyempurnaan selanjutnya.

Malang, Desember 2013

Penyusun

# ABSTRAK

**Ika Kusumaning Putri. 2013. : Rancang Bangun *Game* Sebagai Terapi *Cognitive Behaviour* pada Penderita *Attention Deficit Hyperactivity Disorder* (ADHD) Menggunakan Kinect. Skripsi Program Studi Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing: Eriq M. Adams J., ST., Mkom. dan Aditya Rachmadi, S.ST, M.TI**

*Attention Deficit Hyperactivity Disorder* (ADHD) merupakan gangguan perkembangan peningkatan aktifitas motorik sehingga menyebabkan aktifitas yang tidak lazim dan cenderung berlebihan. Penderitanya seringkali mendapat kesulitan dalam memahami instruksi dan mengingat sesuatu. ADHD merupakan penyakit yang tidak dapat disembuhkan. Penyakit ini hanya dapat dikurangi gejalanya dengan terapi, salah satunya adalah terapi *cognitive behaviour*. Berdasarkan pada permasalahan tersebut, penulis membuat *game* AGTA (Aplikasi *Game* Terapi ADHD) yang membantu meningkatkan konsentrasi penderita *Attention Deficit Hyperactivity Disorder* (ADHD). *Game* AGTA menerapkan teknik interaksi alami dengan memproses setiap gerakan tangan pemain memanfaatkan sensor Kinect. Tujuan dari aplikasi ini untuk meningkatkan daya fokus penderita ADHD melalui implementasi terapi *cognitive behaviour* sehingga dapat melakukan kegiatan layaknya manusia pada umumnya.

*Game* dirancang dengan melalui dua tahapan yaitu *game concept design* dan *technical design*. Implementasi permainan ini menggunakan bahasa pemrograman C# dan menggunakan *framework* XNA. Permainan ini diaplikasikan pada *platform desktop* dengan menggunakan sensor Kinect untuk mendeteksi anggota tubuh. Pengujian aplikasi ini menggunakan metode *white-box testing* dengan strategi pengujian unit dan pengujian integrasi serta metode *black-box testing* dengan strategi pengujian validasi. Pengujian performa permainan dilakukan melalui pengujian FPS (*Frame Per Second*) dan dihasilkan nilai rata-rata FPS *game* yaitu 54 FPS. Dari hasil pengujian didapatkan kesimpulan bahwa permainan ini telah memenuhi kebutuhan fungsional dan non fungsional.

Kata Kunci: *Game*, ADHD, terapi, *cognitive behaviour*, Kinect

## *ABSTRACT*

**Ika Kusumaning Putri. 2013. : *Game Development for Cognitive Behavioral Therapy for People with Attention Deficit Hyperactivity Disorder* (ADHD) *Using Kinect. Undergraduate Thesis of Informatic Engineering Study Program, Information Technology and Computer Science Program,* Brawijaya *University,* Malang. *Advisor*: Eriq M. Adams J., ST., Mkom. *and* Aditya Rachmadi, S.ST, M.TI**

*Attention Deficit Hyperactivity Disorder ( ADHD ) is a mental disorder caused by unusual growth in motor activity. ADHD causing unusual and excessive activity. People with ADHD often have difficulty in understanding the instructions and remember something. ADHD is a disease that can not be cured. Only the symptoms can be reduced with therapy, such as cognitive behavior therapy. Based on these problems, the authors makes AGTA (Applications Games Therapy ADHD) game which help to increase the concentration of children with Attention Deficit Hyperactivity Disorder (ADHD). AGTA applying natural interaction techniques with process each player's hand movements utilizing the Kinect sensor. The purpose of this application is to improve the focus of children with ADHD through the implementation of cognitive behavior therapy that may act like a human being.*

*Game designed in two stages: game concept design and technical design. This game is implemented using C# programming language and XNA framework. This game was applied to the desktop platform using the Kinect sensor to detect the parts of the body. This application was tested using white - box testing that used unit testing and integration testing strategies and black - box testing methods with validation testing strategies. Testing is done by testing the performance of the game FPS (Frames Per Second) and the resulting average value is 54 - FPS. From the test results it was concluded that this game has completed the functional and non-functional requirements.*


Keywords: *Game*, ADHD, *therapy*, *cognitive behaviour*, Kinect

## DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# DAFTAR LAMPIRAN

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

*Attention Deficit Hyperactivity Disorder* (ADHD) merupakan gangguan perkembangan peningkatan aktifitas motorik sehingga menyebabkan aktifitas yang tidak lazim dan cenderung berlebihan. ADHD membawa pengaruh pada setiap aspek kehidupan anak. Penderita ADHD seringkali mendapat kesulitan dalam memahami instruksi dan mengingat sesuatu. Mereka juga sulit untuk ikut serta dalam aktivitas. Terlebih lagi ADHD merupakan penyakit yang tidak dapat disembuhkan. Penyakit ini hanya dapat dikurangi gejalanya dengan terapi. Terapi yang mudah dan tidak menggunakan obat-obatan adalah terapi *cognitive behaviour*. Terapi *cognitive behaviour* dilakukan untuk membantu penderita ADHD untuk beradaptasi dan memperbaiki kemampuan untuk memecahkan masalah. Namun, sampai saat ini fasilitas terapi *cognitive behavior* kebanyakan masih belum banyak tersedia [ELJ-08].

Penderita ADHD membutuhkan suasana yang menarik dan tidak membosankan untuk berkonsentrasi pada kegiatan yang sedang dilakukannya [SHB-05]. Untuk itu dibutuhkan *game* yang dapat menarik perhatian. Objek-objek yang menarik akan ditambahkan pada *game* ini untuk menarik perhatian pengguna. Penderita perlu melakukan kegiatan yang terarah sekaligus meningkatkan fokusnya melalui *game*. Oleh karena itu dibutuhkan teknologi *augmented virtuality* dengan inputan berupa gerakan pengguna yang dapat mendukung terapi tersebut. Teknologi *augmented virtuality* merupakan penggabungan realitas di dunia nyata ke dalam dunia maya [MIL-94]. Teknologi ini mulai banyak dikembangkan pada teknologi kinect. Kemampuan *augmented virtuality* ini dapat memberikan terapi *cognitive behaviour* yang tepat bagi penderita yang kehilangan konsentrasinya.

*Game* ini berupa *game* yang membantu meningkatkan konsentrasi penderita dengan mengadaptasi metode terapi *cognitive behaviour* konvensional. Dalam *game* ini digunakan teknologi kinect sehingga pengguna tidak perlu menggunakan *controller* mengingat pengguna memiliki gangguan ADHD. Peralatan *mobile* seperti *controller* dapat membahayakan baik bagi penderita ADHD maupun peralatan itu sendiri. Harapan dengan adanya aplikasi ini dapat meningkatkan daya fokus penderita ADHD sehingga dapat melakukan kegiatan layaknya manusia pada umumnya.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, penulis merumuskan beberapa masalah sebagai berikut :

1. Bagaimana merancang *gameplay game* sebagai terapi *cognitive behaviour* pada penderita ADHD?
2. Bagaimana mengimplementasikan metode terapi *cognitive behaviour* ke dalam *game* memanfaatkan teknologi *kinect* ?
3. Bagaimana menguji *game* sebagai terapi *cognitive behaviour* pada penderita ADHD?

## 1.3 Batasan Masalah

Agar permasalahan yang dirumuskan lebih terfokus, maka penelitian ini dibatasi oleh hal – hal sebagai berikut :

1. Teknik interaksi alami dalam *game* memanfaatkan sensor Kinect
2. Pengguna *game* ini adalah anak penderita ADHD usia 3-7 tahun
3. Perancangan dan implementasi *game* hanya dikhususkan pada sistem operasi Windows 7 atau versi diatasnya.
4. Proses produksi *game* hanya dilakukan sampai dengan fase *testing*.

## 1.4 Tujuan

Tujuan penulisan skripsi ini adalah untuk dapat merancang *gameplay*, mengimplementasikan metode terapi *cognitive behaviour* ke dalam *game* memanfaatkan teknologi kinect dan menguji *game* sebagai terapi *cognitive behaviour* pada penderita ADHD.

## 1.5 Manfaat

Manfaat yang dapat diberikan *Game* Terapi *Cognitive behaviour* adalah:

1. Mengurangi gejala ADHD untuk usia dini dengan memanfaatkan metode terapi *game* elektronik
2. Meningkatkan daya fokus penderita ADHD sehingga dapat melakukan kegiatan layaknya manusia pada umumnya.
3. Menyediakan sarana aplikasi *game* terapi yang berbeda dan inovatif menggunakan teknologi kinect.

## 1.6 Sistematika Penulisan

Sistematika isi dan penulisan dalam skripsi ini antara lain :

**Bab I** **Pendahuluan**

Berisi tentang latar belakang masalah, perumusan masalah dan pokok-pokok bahasan, tujuan dan manfaat dari penelitian serta sistematika pembahasan.

**Bab II** **Dasar Teori**

Berisi tentang teori dasar *game, ADHD,* sensor Kinect, bahasa pemrograman C#, *Framework* XNA dan penjelasan mengenai proses - proses pembuatan *game*.

**Bab III** **Metodologi dan Perancangan**

Bab ini berisi tentang gambaran umum langkah – langkah dalam pembuatan *game* terapi *cognitive behaviour* dan membahas tentang *game concept design* dan *technical design*.

**Bab IV** **Implementasi**

Membahas tentang implementasi *game*.

**Bab V** **Pengujian dan Analisis**

Memuat hasil pengujian dan analisis terhadap *game* yang telah direalisasikan

**Bab VI** **Penutup**

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian *game*, serta saran – saran untuk pengembangan lebih lanjut.

# BAB II
# DASAR TEORI

Pada bab ini akan diuraikan mengenai teori-teori dasar pembuatan *game* terapi *cognitive behaviour* diantaranya tentang *game*, ADHD, sensor Kinect, XNA dan proses produksi *game*.

## 2.1 *Game*

*Game* adalah media untuk melakukan aktifitas bermain. *Game* dalam penulisan skripsi ini adalah permainan dalam bentuk aplikasi atau perangkat lunak. Aktifitas bermain merupakan suatu aktifitas yang meliputi pemecahan masalah yang menjadi tantangan dari *game* tersebut, dengan mengikuti suatu aturan tertentu [MAH-10]. *Game* dikategorikan menjadi beberapa tipe atau yang disebut dengan *genre game* yaitu *action, adventure, casual, educational, role-playing game* (RPG), *simulation, sports dan strategy. Game* terapi *cognitive behaviour* ini adalah *game* yang memiliki *genre action.*

*Action game* adalah tipe *game* yang menuntut pemain untuk selalu bergerak, menyerang dan bereaksi. Aksi adalah penekanan utama dari *action game* bukan pada *story telling*. *Action game* akan menggunakan animasi atau *splash screen* pada awal permulaan *game* untuk menjelaskan jalan cerita [PED-08].

## 2.2 ADHD (*Attention Deficit Hyperactivity Disorder*)

ADHD merupakan gangguan perkembangan dalam peningkatan aktifitas motorik anak-anak hingga menyebabkan aktifitas anak-anak yang tidak lazim dan cenderung berlebihan. Hal ini ditandai dengan keluhan perasaan gelisah, tidak bisa diam, tidak dapat duduk dengan tenang dan selalu meninggalkan keadaan yang tetap seperti sedang duduk.

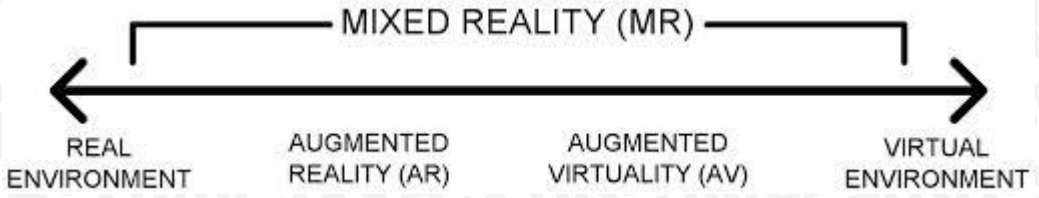
Gejala kurang konsentrasi yang terjadi pada anak ADHD, sangat mempengaruhi perkembangan kognitif, perilaku, sosialisasi, dan komunikasi sehingga dalam kegiatan akademisnya terganggu dan sering

mendapatkan nilai yang jelek meskipun mempunyai intelegensi yang baik [IND-07]. Oleh karena itu dibutuhkan terapi yang mampu melatih konsentrasi untuk membantu anak ADHD. Banyak sekali terapi – terapi yang disarankan untuk meningkatkan konsentrasi pada anak ADHD, namun terapi bermain akan lebih efektif karena anak- anak akan menyukai proses terapi yang tidak membosankan. Dengan permainan anak- anak dapat bermain tanpa merasa sedang menjalani proses terapi dan menjadi lebih bebas mengekspresikan tingkah lakunya.

Berdasarkan penelitian yang dilakukan oleh Lina Indah Wahyuni, bahwa dari hasil penelitiannya diketahui subyek penderita ADHD sebelum diberikan terapi bermain mengalami kesulitan memberikan perhatian terhadap detail dan selalu membuat kesalahan pada banyak kegiatan, tidak mampu mempertahankan perhatian dalam menerima tugas dan bermain, tampak tidak mendengarkan pada saat diajak berbicara langsung, dan sering tidak mengikuti instruksi dan tidak mampu menyelesaikan tugasnya. Setelah terapi bermain diberikan subyek mengalami peningkatan dalam mempertahankan perhatiannya, sedangkan dalam memberikan perhatian terhadap detail, selalu membuat kesalahan, tidak mendengarkan saat diajak bicara dan tidak menyelesaikan tugasnya, hanya mengalami sedikit perubahan.

### *2.3 Augmented Virtuality(AV)*

*Augmented virtuality* (AV) merupakan penggabungan objek dunia nyata ke dunia maya [MIL-94]. Sesuai dengan *Continuum* Milgram yang ditunjukkan pada gambar 2.1, AV lebih dominan pada lingkungan *virtual*, dimana unsur-unsur fisik secara dinamis diintegrasikan dan dapat berinteraksi dengan dunia maya secara *real time*. Integrasi ini dicapai dengan menggunakan berbagai teknik misalnya menggunakan kamera atau menggunakan digitalisasi 3 dimensi dari benda-benda fisik. Penggunaan informasi sensor dunia nyata untuk mengendalikan lingkungan *virtual* merupakan bentuk tambahan dari AV, dimana input eksternal menyediakan konteks untuk tampilan *virtual*.




**Gambar 2. 1 *Virtuality Continuum* Paul Milgram**
**Sumber**: [MIL-94]

## 2.4 Kinect

Kinect adalah perangkat *input* untuk mendeteksi gerakan yang diproduksi oleh Microsoft untuk *Video Game* XBOX 360 dan PC dengan sistem operasi Windows. Dengan menggunakan kamera yang mirip dengan *webcam*, memungkinkan Kinect untuk menangkap gerakan pengguna yang akhirnya pengguna tidak perlu menyentuh secara langsung *controller game*. Cukup dengan melakukan gerakan-gerakan yang alami.

Kinect dibangun dengan menggunakan teknologi *software* yang dikembangkan secara internal oleh Rare, sebuah perusahaan *game* dibawah Microsoft *Game* Studios milik Microsoft. Kamera pada Kinect dikembangkan oleh pengembang asal Israel yakni PrimeSense, yang mengembangkan sebuah sistem yang mampu mengartikan gerakan secara tepat, yang akhirnya memungkinkan pengaturan tanpa tangan pada perangkat elektronik dengan menggunakan proyektor *infrared* dan kamera dan sebuah *microchip* untuk mendeteksi gerakan obyek dalam 3 dimensi.

Sensor Kinect terdiri dari sebuah *horizontal bar* yang terhubung pada kaki kecil dengan sebuah poros yang dilengkapi motor dan didesain memanjang diatas atau dibawah *video display*. Alat ini memiliki sebuah kamera RGB, sensor kedalaman dan *multi-array michrophone* yang dilengkapi *software*, yang mampu menyajikan pengenalan secara 3D pada seluruh tubuh dan juga kemampuan pengenalan suara. Penjelasan bagian sensor Kinect dapat dilihat pada gambar 2.2.

Sensor kedalaman terdiri dari proyektor laser *infrared* yang dikombinasikan dengan sensor CMOS yang menangkap data *video* dalam bentuk 3D pada kondisi cahaya ambien. Jarak penginderaan dari sensor

kedalaman dapat diatur, dan *software* Kinect secara otomatis mampu melakukan kalibrasi pada sensor berdasarkan *gameplay* dan lingkungan fisik pemain, mampu mengakomodasi adanya furnitur atau halangan lainnya.

Terdapat 3 inovasi *hardware* yang bekerja bersama-sama di dalam sensor Kinect:

1. *Color VGA video camera*

*Camera video* ini membantu dalam pengenalan wajah dan fitur deteksi lainnya dengan mendeteksi 3 komponen warna yaitu *Red*, *Green*, dan *Blue*. Microsoft menamakannya "*RGB Camera*" dengan mengacu pada 3 komponen warna tersebut.


**Gambar 2. 2 Sensor Kinect**
**Sumber :** [BOU-11]

2. *Depth Sensor*

*Depth sensor* atau sensor kedalaman merupakan sebuah proyektor *infrared* dan sebuah sensor *monochrome* CMOS yang bekerja secara bersama-sama untuk "melihat" ruangan atau area dalam bentuk 3D dengan tanpa memperdulikan kondisi cahaya

3. *Multi-array michrophones*

Merupakan sebuah susunan yang terdiri dari 4 kamera yang dapat mengisolasikan suara dari pemain dengan suara-suara lain(*noise*) yang ada di ruangan. Hal ini memungkinkan pemain *game* untuk berada agak jauh dari *microphone* dan masih dapat menggunakan kontrol suara atau *voice control*

## 2.5 Bahasa Pemrograman C#

C# adalah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh *Microsoft Corporation* sebagai bagian dari kerangka *.Net Framework*. Bahasa ini yang sangat fleksibel dan kuat. C# memiliki kemiripan kuat dengan C++ dan bahasa pemrograman Java, setelah memperbaiki fitur yang disediakan oleh bahasa tersebut.

Seperti halnya bahasa Java, bahasa C# telah membuang beberapa fitur berbahaya dari bahasa C. C# adalah bahasa pemrograman yang telah di-*compile*. Komputer tidak mengerti bahasa tersebut secara langsung, oleh karena itu sebuah program yang bernama *compiler* mengkonversi teks C# menjadi instruksi level bawah yang lebih sederhana [ROM-11].

## 2.6 *Framework* XNA

XNA adalah sebuah *framework* dari Microsoft untuk memudahkan *developer* membangun *game* untuk PC dan XBOX, dan Zune. XNA berjalan diatas Visual Studio dalam bahasa C#. XNA merupakan singkatan dari '*XNA's Not Acronymed*'. Keunggulan XNA antara lain:

- Gratis.
- Portabilitas. *Game* yang dibuat dengan XNA dapat dijalankan di semua *Platform* yang mendukung XNA Framework.

Selain ditujukan untuk pengembangan *game-game* yang berbasis Windows *desktop*, XNA *Game* Studio juga dapat digunakan untuk mengembangkan *game* yang berbasis *console* Xbox 360 dan Zune. Sebuah *project* yang dibuat untuk *platform* Windows *desktop* pun dapat dikonversikan ke dalam *project* Xbox atau Zune. Sehingga hal ini sangat memudahkan pengembang yang ingin membuat *game multiplatform*.

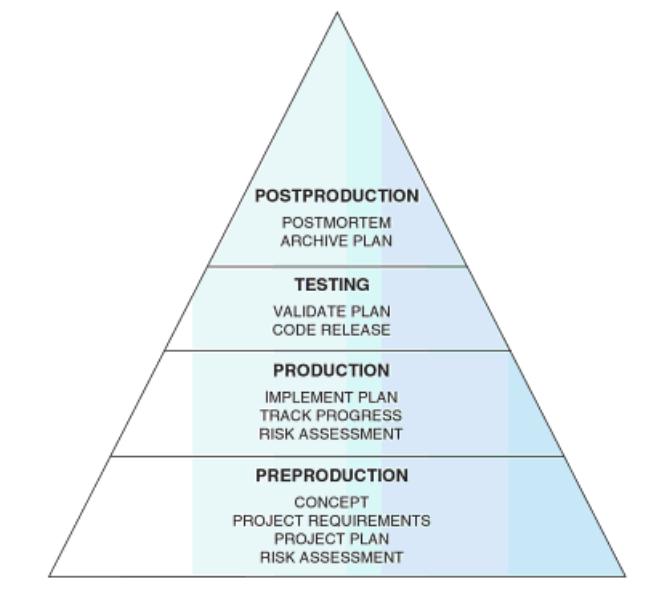
## 2.7 *Game Production*

Proses produksi dimulai dengan mendefinisikan konsep *game* dan berakhir dengan membuat sebuah *gold master of the final game code*. Proses dapat dipecah menjadi 4 tahap yaitu : (1) *preproduction*,(2)

*production*, (3) *testing*, (4) *postproduction* [CHA-11]. Gambar 2.3 menunjukan gambaran ikhtisar dari *basic production cycle*.

### *2.7.1 Preproduction Phase*

Tujuan dari tahap ini adalah membuat *game plan* yang menjadi *road map* untuk menyelesaikan *game* dan menyelesaikan kode *game* [CHA-11]. Pada pembuatan *game* AGTA, *preproduction phase* terdiri dari *game concept design* dan *technical design.*




**Gambar 2.3.** *Basic Production Cycle*

**Sumber :** [CHA-11]

#### *2.7.1.1 Game Concept Design*

Tahap *game concept design* dilakukan dengan pembuatan *game design document*. Pada tahap pembuatan *game design document*, langkah yang pertama kali dilakukan adalah mencari ide. Pencarian ide dapat dilakukan dengan menuliskan ide yang ada atau berdiskusi dengan kelompok orang yang memiliki disiplin ilmu yang berbeda. Kemudian dari ide-ide yang didapatkan akan ditulis pada sebuah catatan yang disebut dengan *brainstorming note* [ROG-10]. *Brainstroming note* disusun dalam sebuah *one sheet document* agar lebih terstruktur sehingga pembaca awam mengerti ringkasan tentang *game* yang dibuat [ROG-10]. Langkah selanjutnya adalah membuat *ten page design document*, isi dari dokumen

ini adalah detail dari *one sheet document*. Dokumen ini maksimal terdiri dari 10 halaman dimana isi dari dokumen ini hanya bagian-bagian penting dari *game* yang dibuat seperti *title, gameplay, game controller, character, game interface,* dan lain-lain sehingga pembaca mengerti secara garis besar *game* yang dibuat tanpa harus mengetahui produk final *game*[ROG-10]. Langkah terakhir dalam *game concept design* ini adalah membuat *game design document* yang merupakan detail dari *ten page design document*. Tujuan dibuatnya dokumen ini adalah untuk mengkomunikasikan kepada tim untuk meminimalisir kesalahan pada saat implementasi karena kurang mengerti tentang *game* yang dibuat [ROG-10].

#### *2.7.1.2 Technical Design*

Pada tahap *technical design,*pemodelan sistem dilakukan dengan menggunakan UML (*Unified Markup Language*). UML adalah bahasa grafis untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. UML berorientasi objek, menerapkan banyak level abstraksi, tidak bergantung proses pengembangan, tidak bergantung bahasa dan teknologi [HAR-04]. Pada tahapan ini akan menghasilkan spesifikasi teknis yang biasanya ditulis oleh *lead programmer* dan digunakan sebagai acuan oleh tim pemrograman. Spesifikasi teknis ini disebut juga dengan *technical design document.* Pada *game design document* berfokus pada bagaimana *game* akan berfungsi, sedangkan dalam *technical design document* membahas bagaimana fungsi tersebut akan diimplementasikan [ROU-10]. Tahapan dari *technical design game* AGTA yaitu : (1) pembuatan diagram *use case,*(2) *pembuatan* rancangan diagram kelas, (3) pembuatan rancangan *activity diagram.*

1. *Use Case Diagram*

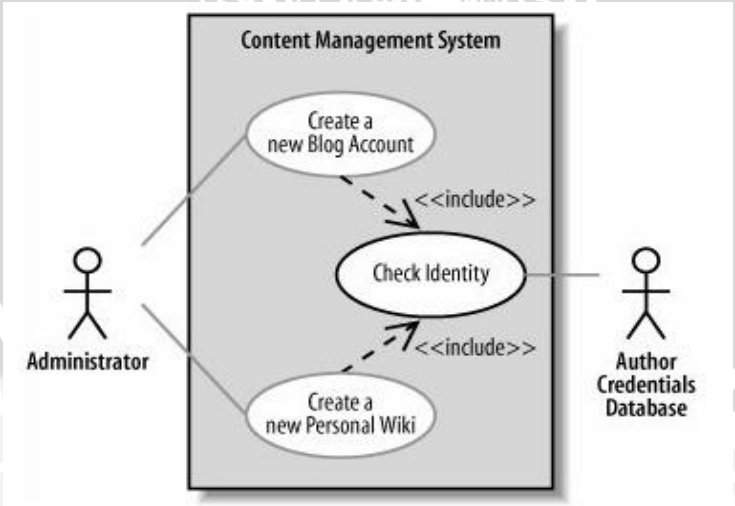
   *Use case diagram* merupakan salah satu diagram untuk memodelkan aspek perilaku sistem. Diagram *use case* adalah penting untuk memvisualisasikan, menspesifikasikan dan mendokumentasikan

kebutuhan perilaku sistem. Tujuan utama pemodelan *use case* adalah [HAR-04] :

1. Memutuskan dan mendeskripsikan kebutuhan – kebutuhan fungsional sistem.
2. Memberikan deskripsi jelas dan konsisten dari apa yang seharusnya dilakukan, sehingga model *use case* digunakan di seluruh proses pengembangan untuk komunikasi dan menyediakan basis untuk pemodelan berikutnya yang mengacu sistem harus memberikan fungsionalitas yang dimodelkan pada *use case*.
3. Menyediakan basis untuk melakukan pengujian sistem yang memverifikasi sistem. Untuk menguji fungsionalitas sistem sesuai yang diminta.
4. Menyediakan kemampuan melacak kebutuhan fungsionalitas menjadi kelas – kelas dan operasi – operasi aktual di sistem.

Elemen diagram *use case* adalah [HAR-04] :

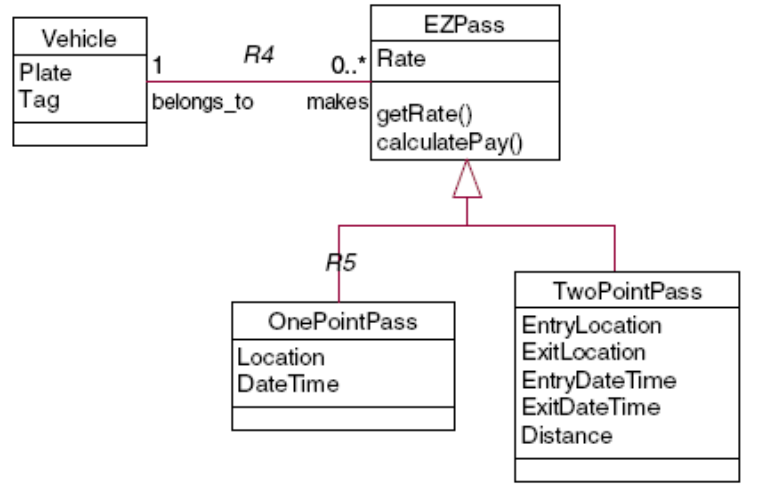
1. Aktor merupakan pemakai sistem.
2. *Use case* adalah cara spesifik penggunaan sistem oleh aktor.
3. Hubungan ketergantungan, generalisasi dan asosiasi.
4. Contoh dari diagram *use case* ditunjukan pada gambar 2.4.




**Gambar 2.4.** Diagram *Use Case*.
**Sumber :** [HAM-06]

2. *Class Diagram*

Diagram kelas merupakan diagram paling umum dipakai di semua pemodelan berorientasi objek. Pemodelan kelas menunjukan kelas- kelas yang ada di sistem dan hubungan antar kelas – kelas itu, atribut – atribut dan operasi – operasi di kelas – kelas. Diagram kelas menunjukan aspek statik sistem terutama untuk mendukung kebutuhan fungsionalitas sistem. Kelas di diagram kelas dapat secara langsung diimplementasikan ke dalam bahasa pemrograman berorientasi objek yang secara langsung mendukung bentukan kelas [HAR-04]. Contoh diagram kelas ditunjukan pada gambar 2.5.




**Gambar 2.5. Diagram Kelas**
**Sumber :** [LIU-06]

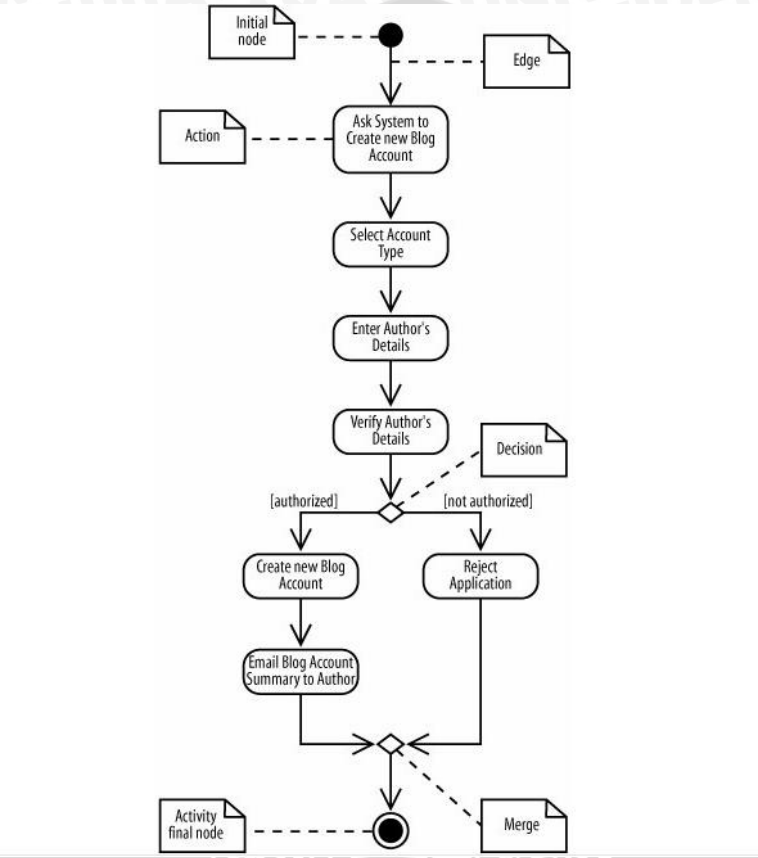
Elemen – elemen esensi di diagram kelas adalah sebagai berikut :

1. Kelas
2. Antarmuka (*Interfaces*)
3. Kolaborasi
4. Hubungan (*Relationship*) seperti kebergantungan, generalisasi dan asosiasi.

3. *Activity Diagram*

Tahap perancangan selanjutnya adalah pembuatan diagram aktivitas. Dengan diagram aktivitas memungkinkan untuk menentukan bagaimana sistem akan mencapai tujuannya [HAM-06]. Sebagai contoh, diagram aktivitas digunakan untuk memodelkan langkah – langkah dalam

pembuatan akun sebuah *blog*. Dalam perancangan *game* AGTA, diagram aktivitas digunakan untuk menggambarkan aliran aktivitas atau proses dari setiap *use case* dalam *game* AGTA. Gambar 2.6 menunjukan contoh dari diagram aktivitas proses pembuatan akun *blog*.




**Gambar 2.6.** Diagram Aktivitas Proses Pembuatan Akun *Blog*.
**Sumber :** [HAM-06]

### *2.7.2 Production Phase*

Tahap ini adalah tahap yang dilakukan untuk merealisasikan rancangan dari *game*. Tahap ini terdiri dari *early production*, *mid-production* dan *late production*.

#### *2.7.2.1 Early Production*

*Early production* merupakan tahap realisasi konsep *game*. Pada *game* AGTA, tahapan ini terdiri atas pembuatan *gameplay* utama, pengintegrasian dengan *kinect*, pembuatan gambar utama/elemen *game* dan kontrol atau inputan *game*.

#### *2.7.2.2 Mid-Production*

*Mid-production* merupakan tahap pengembangan *game Alpha production* untuk mendeteksi kelemahan utama dalam permainan saat masih relatif mudah untuk diperbaiki. Pada *game* AGTA, tahapan ini terdiri dari melengkapi semua fitur *game* yang telah dirancang, pembuatan beberapa gambar tambahan dan *layout* serta pembuatan musik latar.

#### *2.7.2.3 Late Production*

*Late-production* merupakan tahap pengembangan *game Beta production* untuk memastikan bahwa *game* ini siap untuk dirilis. Pada tahapan ini *game* AGTA harus benar-benar lengkap dari fitur hingga gambar dan musik.

### *2.7.3 Testing*

Pengujian perangkat lunak memerlukan perancangan kasus uji (*test case*) agar dapat menemukan kesalahan dalam waktu singkat dan usaha minimum. Berbagai macam metode perancangan kasus uji telah berevolusi. Metode ini menyediakan pendekatan sistematis untuk pengujian oleh *developer*. Terlebih lagi metode ini menyediakan mekanisme yang dapat membantu memastikan kelengkapan dari pengujian dan menyediakan kemungkinan tertinggi untuk menemukan kesalahan-kesalahan dalam perangkat lunak. Teknik atau metode perancangan kasus uji yang digunakan adalah *black-box testing* dan *white-box testing* [PRE-10].

#### *2.7.3.1 White Box Testing*

*White box testing* adalah teknik pengujian yang menguji berdasarkan jalur internal, struktur dan implementasi dari perangkat lunak yang sedang diuji. *White box testing* adalah teknik pengujian yang menggunakan struktur kontrol dari prosedur yang terdapat dalam perancangan untuk membuat kasus uji. Ada dua jenis pengujian yang termasuk *white box testing* yaitu *basis path testing* dan *control structure testing. Basis path testing* adalah teknik dalam metode *white box testing* untuk menyediakan desain *test case* yang dapat digunakan untuk menghitung kompleksitas desain prosedural untuk menghitung banyaknya jalur eksekusi. *Test case* tersebut digunakan untuk menjamin apakah tiap statement telah dijalankan minimal satu kali tiap pengujian. Dalam konteks *basis path*, nilai

yang dihitung di dalam *cyclomatic complexity* menjelaskan banyaknya jumlah jalur *independent* yang harus disediakan dalam pengujian *basis path* [PRE-10].

#### *2.7.3.2 Black Box Testing*

*Black box testing* adalah teknik pengujian yang menguji hanya berdasarkan kebutuhan dan spesifikasi. *Black box testing* juga disebut sebagai *behavioral testing* dan berfokus pada kebutuhan fungsi dari perangkat lunak [PRE-10]. Proses umum yang terjadi pada *black box testing* yaitu:

a. Kebutuhan atau spesifikasi dianalisa terlebih dahulu.
b. Penentuan *input valid* terpilih berdasarkan spesifikasi untuk menentukan perangkat lunak berjalan dengan benar. *Input* yang tidak *valid* juga harus dipilih untuk memverifikasi bahwa perangkat lunak dapat mendeteksinya dan menanganinya dengan baik.
c. Penentuan *output* yang diharapkan sesuai dengan *input* yang telah dipilih.
d. Pengujian dibuat dengan *input* yang telah dipilih.
e. Pengujian dijalankan.
f. *Output* yang sebenarnya dibandingkan dengan *output* yang diharapkan

Penentuan dibuat menyangkut perangkat lunak berfungsi sesuai dengan spesifikasi yang telah ditentukan.

#### 2.7.3.3 Pengujian Performa

Pengujian ini dilakukan dengan melihat *frame rate* (*Frame Per Second*), merupakan jumlah *frame* animasi bergerak yang ditampilkan dalam setiap detik. Misal dalam kartun Bugs Bunny, normalnya kartun berkedip dengan 16 FPSper detik. Ini cukup cepat untuk mata manusia percaya bahwa gambar bergerak. Dalam permainan untuk mencapai 16 FPS, perangkat perlu menggambar setiap frame selama 62,5 milidetik. Namun permainan tidak sedetail kartun.

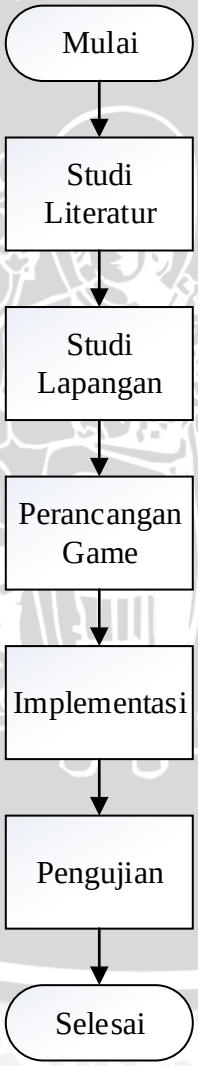
### *2.7.4 Postproduction Phase*

Pada proses *postproduction* yang dilakukan adalah pembuatan *archieve* dan publikasi *game*. *Archieve game* AGTA berisi dokumen konsep, aset dan kode dari *game* AGTA. Publikasi *game* diawali dengan pembuatan *installer game* dan *manual book* atau *user guide*. *Game* AGTA dibuat untuk anak ADHD saja, oleh

karena itu *game* ini hanya akan dipublikasikan pada sekolah anak berkebutuhan khusus. *Postproduction phase* tidak dilaksanakan pada skripsi ini, sehingga tidak dilaporkan dalam penulisan laporan skripsi ini.

# BAB III
# METODOLOGI DAN PERANCANGAN

Pada bab ini dijelaskan mengenai metodologi yang dilakukan dalam penelitian serta perancangan *Game* sebagai Terapi *Cognitive Behaviour* pada Penderita *Attention Deficit Hyperactivity Disorder* (ADHD) menggunakan *Kinect*. Diagram alir metodologi penelitian ditunjukkan pada gambar 3.1. Perancangan yang terdiri atas dua tahap yaitu *Game Design* dan *Technical Design* dijelaskan pada sub bab selanjutnya.




**Gambar 3. 1 Diagram alir metode penelitian**

## 3.1 Studi Literatur

Sebelum perancangan *game* akan dilakukan studi literatur terlebih dahulu untuk memahami konsep-konsep yang harus dipelajari.

Literatur tersebut antara lain:

1. *Game*
2. ADHD
3. *Augmented Virtuality*
4. *Kinect*
5. Bahasa Pemrograman C#
6. *Framework* XNA
7. Proses produksi *game*

## 3.2 Studi Lapangan

Penelitian sebelum melakukan pembuatan *game* untuk terapi *cognitive behaviour* perlu dilakukan untuk mendapatkan hasil terapi yang efektif. Penelitian yang dilakukan antara lain:

- Mencari informasi tentang ADHD dan implementasi terapi yang tepat
- Menentukan perilaku penderita ADHD yang perlu ditingkatkan melalui *game*
- Menentukan *gameplay* berdasarkan informasi dari terapis dan berdasarkan penelitian
- *Gameplay* yang diberikan harus yang tidak membosankan bagi pengguna dan gambar di dalamnya sangat menarik
- Menentukan metode penilaian untuk mengetahui keberhasilan program

## 3.3 Perancangan *Game*

Setelah dilakukan studi lapangan, dapat ditentukan perancangan *game* yang tepat untuk terapi berdasarkan informasi terapis dan literatur. Tahap perancangan *game* terdiri dari dua bagian yaitu *game concept design* dan *technical design.*

### *3.3.1 Game Concept Design*

Tahap pembuatan *game concept design,* langkah yang pertama kali

dilakukan adalah mencari ide. Pencarian ide dapat dilakukan dengan menuliskan ide yang ada atau berdiskusi dengan kelompok orang yang memiliki disiplin ilmu yang berbeda. Kemudian dari ide-ide yang didapatkan akan ditulis pada sebuah catatan yang disebut dengan *brainstorming note* [ROG-10]. Pada bab ini dijelaskan pembuatan *game concept design* yang merupakan detail-detail dalam permainan. Tujuan dibuatnya dokumen ini adalah untuk mengkomunikasikan kepada tim untuk meminimalisir kesalahan pada saat implementasi karena kurang mengerti tentang *game* yang dibuat [ROG-10]. *Game concept design* mencantumkan segala rincian dalam desain pembuatan *game*. Rincian tersebut meliputi judul *game*, konsep utama *game*, kebutuhan teknologi, tujuan *game*, deskripsi *gameplay*, kontrol permainan, dunia dalam permainan, *gameflow*, *game elements*, antarmuka, musik dan suara.

#### *3.3.1.1 Title Page*

Berikut merupakan hal-hal yang terdapat pada *title page*

***Game Title* :**AGTA(Aplikasi *Game* Terapi ADHD)


**Gambar 3. 2 Logo *game* AGTA**

***Intended Game System* :** Microsoft Windows Vista, 7 atau versi diatasnya

***Target User* :** 3-7 tahun

**Logo :** logo *game* AGTA ditunjukan pada gambar 3.2.

#### *3.3.1.2 Main Gameplay Concepts and Platform Specific Features*

- *Genre* : *Action*
- *Platform*: PC Windows
- *Features*

- ***Therapy Functionality*** **:** sebagai media terapi *cognitive behaviour* pada anak ADHD
- ***Natural Interaction techniques*****:** kursor *game* digerakkan dengan gerakan tangan menggunakan sensor Kinect.
- ***Interesting Art*** **:** Menggunakan gambar 2D yang menarik untuk anak ADHD.
- ***Fun and Active Gameplay*****:** menangkap ubur-ubur dan hewan laut kursor yang digerakan menggunakan gerakan tangan.
- ***Two Gameplay in One Game*****:** Terdapat dua *game* dalam satu aplikasi *game* terapi agar anak tidak bosan.

#### *3.3.1.3 Gameplay*

AGTA merupakan permainan *single player*. Terdapat dua *game* pada AGTA, antara lain:

*a. Jellyfish Catcher*

*Jellyfish Catcher* merupakan permainan menangkap ubur-ubur. Pemain diharuskan menggerakkan jaring pada layar untuk menangkap ubur-ubur yang sedang berenang.

*b. Falling Party*

*Falling Party* merupakan permainan menangkap hewan-hewan laut yang jatuh. Pemain diharuskan menggerakkan mangkuk pada layar untuk menangkap hewan-hewan laut tersebut.

*Game* yang diberikan hanya berdurasi 3 menit untuk setiap *game*. Di dalam *game* tersebut telah disisipi perhitungan waktu saat pemain mulai bermain hingga pemain meninggalkannya

#### *3.3.1.4 Game Controller*

Kendali pada permainan AGTA yang utama yaitu menggunakan gerakan tangan memanfaatkan sensor kamera Kinect. Kontrol permainan utama adalah menggerakkan tangan kanan dan kiri menyesuaikan posisi musuh yang akan ditangkap. Contoh kontrol permainan ditunjukkan pada gambar 3.3.


**Gambar 3.3 Kontrol permainan**
**Sumber** : [ANO-11]

### *3.3.1.5 Game World*

Dunia dalam permaian di rancang sedemikan rupa dengan konsep *art* 2D menyerupai lingkungan bawah laut dengan latar belakang dunia nyata pemain sehingga pemain serasa berada dalam laut.

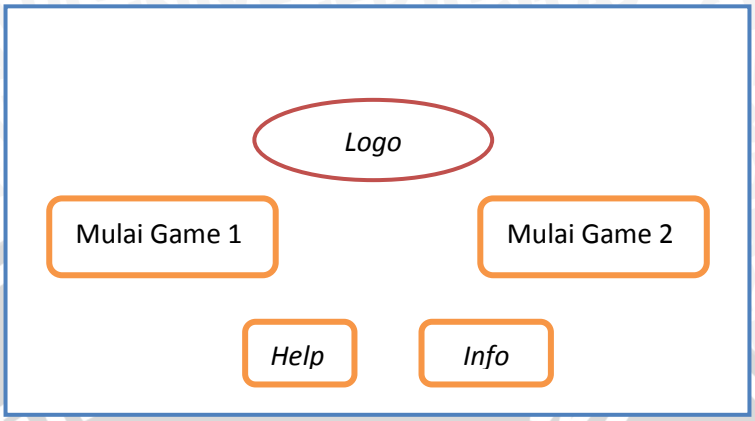
Rancangan dunia dalam permainan ditunjukkan pada gambar 3.4.

**Gambar 3.4 Gambar dunia dalam permainan**

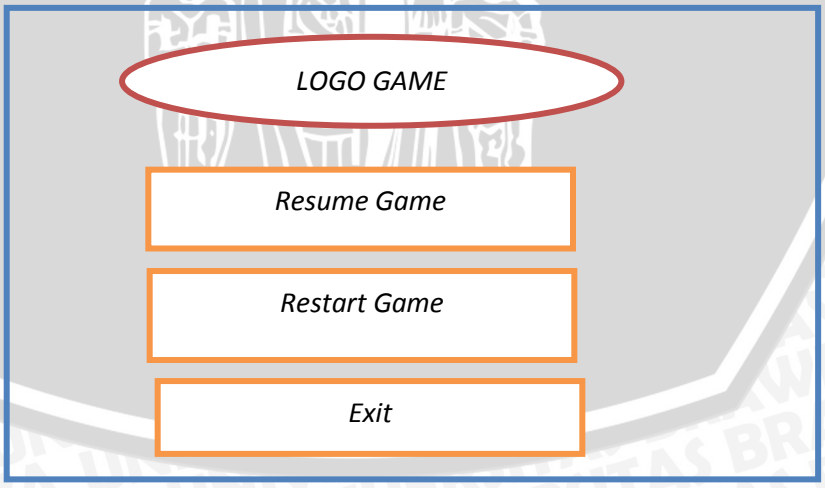
### *3.3.1.6 Interface*

Pada *game* AGTA dirancang 5 *screen* yang terdiri dari menu utama pada tabel 3.1, *information screen, help screen, in game screen* pada tabel 3.3 *dan* menu *pause* pada tabel 3.2. Gambar latar belakang pada permainan AGTA merupakan gambar dua dimensi dengan kamera yang tetap. Sistem HUD pada permainan AGTA memberikan informasi waktu dan skor pemain.

**Tabel 3. 1 Rancangan Antarmuka Menu Utama**

| **Nama Layar** | ***Main Menu*** |
|---|---|
| **Sketsa** |  |
| **Keterangan** | *Main menu* berisi tombol *Catch the Jellyfish*, *Falling Party*, *help* dan *information.*<br>1. *Catch the Jellyfish* : Mulai permaian *Catch the Jellyfish*<br>2. *Falling Party* : Mulai permainan *Falling Party*<br>3. *Help* : Cara menggunakan *game*<br>4. *Info* : Melihat informasi dan manfaat *game* |

**Tabel 3. 2 Rancangan Antarmuka *Pause***

| **Nama Layar** | ***Pause*** |
|---|---|
| **Sketsa** |  |
| **Keterangan** | *Pause Screen* berisi tombol *resume*, *restart* dan *exit*.<br>1. *Resume*: Kembali ke *game screen* dan melanjutkan permainan. |

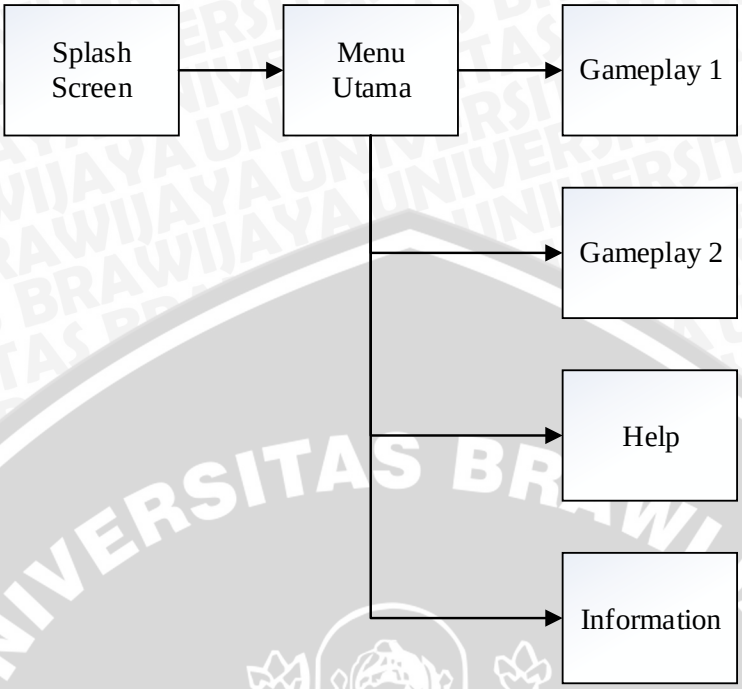
| | |
|---|---|
| | 2. *Restart* : Mengulangi *game* dari awal permainan<br>3. *Exit* : Keluar dari *game* dan kembali ke main menu |

**Tabel 3. 3 Rancangan Antarmuka *Gameplay***

| **Nama Layar** | ***Gameplay*** |
|---|---|
| **Sketsa** |  |
| **Keterangan** | *Gameplay screen* berisi tombol *pause*<br>1. *Pause* : Menghentikan permainan sementara dan menampilkan *pause screen*. |

### *3.3.1.7 Game Flow*

*Game flow* merupakan gambaran aliran jalan permainan yang dapat dilihat oleh pemain. Terdapat beberapa layar dalam permainan ini meliputi *splash screen*, menu utama, *gameplay 1, gameplay 2, information* dan *help*. Gambaran *game flow* ditunjukkan pada Gambar 3.5.




**Gambar 3. 5 *Game Flow***

***3.3.1.8 Game elements***

Prototipe elemen *game* AGTA terdiri dari :

1) Gambar tangkapan

   Dirancang 6 sketsa untuk gambar tangkapan yaitu gambar macam-macam ubur-ubur, gambar bintang laut dan gambar kuda laut. Sketsa gambar tangkapan ubur-ubur ditunjukan pada tabel 3.4, 3.5, 3.6 dan 3.7, sketsa gambar tangkapan bintang laut ditunjukan pada tabel 3.8 dan sketsa gambar tangkapan kuda laut ditunjukan pada tabel 3.9.

2) *Hand Cursor*

   Sketsa *Hand Cursor* pemain pada *main menu* ditunjukkan oleh Tabel 3.10 sedangkan pada *game Catch the Jelly* ditunjukan oleh tabel 3.11 dan tabel 3.12 untuk *Hand Cursor* pada *game Falling Party*. *Hand Cursor* ini yang akan digerakan oleh gerakan tangan pemain untuk menangkap hewan tangkapan yang melintas pada layar.

**Tabel 3. 4 Sketsa Ubur-Ubur *King* pada *gameplay* 1**

| Sketsa | Profil | |
|---|---|---|
|  | **Nama** | *Jelly King* |
| | **Tampilan** | Berwarna ungu, tentakel panjang, menggunakan mahkota raja |
| | **Ukuran** | Sedang |
| | **Kecepatan** | Cepat |
| | **Skor** | 10 |

**Tabel 3. 5 Sketsa Ubur-Ubur *Queen* pada *gameplay* 1**

| Sketsa | Profil | |
|---|---|---|
|  | **Nama** | *Jelly Queen* |
| | **Tampilan** | Berwarna merah muda, tentakel panjang, menggunakan mahkota ratu |
| | **Ukuran** | Sedang |
| | **Kecepatan** | Cepat |
| | **Skor** | 10 |

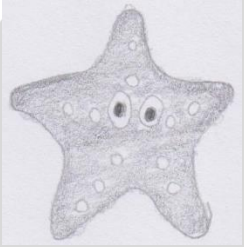
**Tabel 3. 6 Sketsa Ubur-Ubur Gemuk pada *gameplay* 1**

| Sketsa | Profil | |
|---|---|---|
|  | **Nama** | *Jelly Fat* |
| | **Tampilan** | Berwarna ungu dan orange, tentakel pendek, badan gendut |
| | **Ukuran** | Sedang |
| | **Kecepatan** | Lambat |
| | **Skor** | 3 |

**Tabel 3. 7 Sketsa Ubur-Ubur panjang pada *gameplay* 1**

| Sketsa | Profil | |
|---|---|---|
|  | **Nama** | *Jelly Long* |
| | **Tampilan** | Berwarna biru dan merah, tentakel pendek, badan panjang |
| | **Ukuran** | Sedang |
| | **Kecepatan** | Sedang |
| | **Skor** | 5 |

**Tabel 3.8 Sketsa Bintang Laut pada *gameplay* 2**

| Sketsa | Profil | |
|---|---|---|
|  | **Nama** | Bintang Laut |
| | **Tampilan** | Berwarna biru, orange, pink dan hijau |
| | **Ukuran** | Sedang |
| | **Kecepatan** | Sedang |
| | **Skor** | 10 |

**Tabel 3.9 Sketsa Kuda Laut pada *gameplay* 2**

| Sketsa | Profil | |
|---|---|---|
|  | **Nama** | Kuda Laut |
| | **Tampilan** | Berwarna coklat, hijau dan ungu |
| | **Ukuran** | Sedang |
| | **Kecepatan** | Lambat |
| | **Skor** | 10 |

**Tabel 3. 10 Sketsa *Hand Cursor* pada *Main Menu***

| Sketsa | Profil | |
|---|---|---|
|  | **Nama** | *Hand Cursor* |
| | **Keterangan** | *Hand Cursor* ini yang akan digerakan oleh gerakan tangan pemain untuk memilih pada menu utama. |

**Tabel 3. 11 Sketsa Jaring pada *gameplay* 1**

| Sketsa | Profil | |
|---|---|---|
|  | **Nama** | Jaring |
| | **Keterangan** | Jaring ini yang akan digerakan oleh gerakan tangan kanan pemain untuk menangkap ubur-ubur pada *game Catch the Jelly*. |

**Tabel 3. 12 Sketsa Ember pada *gameplay* 2**

| Sketsa | Profil | |
|---|---|---|
|  | **Nama** | Ember |
| | **Keterangan** | Ember ini yang akan digerakan oleh gerakan tangan kanan dan kiri pemain untuk menangkap hewan laut pada *game Falling Party.* |

#### *3.3.1.9 Musik dan Sound Effect*

Musik yang digunakan dalam permainan memiliki karakteristik cepat dan semangat agar membangkitkan semangat pemain.

Terdapat 1 musik latar pada permainan AGTA yaitu musik utama yang dimainkan pada menu utama, *information* dan *help.*

*Sound effect* permainan AGTA terdiri dari :

1) Hewan laut ketika tertangkap
2) *Sound effect* ketika waktu telah habis

#### *3.3.1.10 Kebutuhan Teknologi*

Daftar kebutuhan teknologi untuk membangun *game* ditunjukkan pada Tabel 3.13.

**Tabel 3.13 Kebutuhan Teknologi**

| No | Kebutuhan | Kegunaan |
|---|---|---|
| 1. | Aplikasi pengolah grafis | Sebagai media gambar *digital*. |
| 2. | Aplikasi Visual Studio 2010 | Untuk membangun aplikasi menggunakan bahasa pemrograman C# |
| 3. | Kinect | Sebagai alat sensor gerakan tangan untuk inputan pemain |

#### *3.3.1.11 Tujuan Game*

Tujuan utama dari *game* ini adalah mengimplementasikan terapi *cognitive behaviour* pada anak ADHD untuk mengurangi gejala penyakitnya.

### *3.3.2 Technical Design*

*Technical design* menjelaskan aspek teknis dari pengembangan *game y*ang meliputi pembuatan diagram *use case*, diagram *class* dan diagram *activity*.

#### 3.3.2.1 *Use Case*

*Use case* merekam kebutuhan dari sebuah sistem. *Use case* dimodelkan dengan sebuah diagram *use case*. Pembuatan diagram *use case* dimulai dengan identifikasi aktor dan identifikasi kebutuhan.

1. Indentifikasi aktor

   Tahap ini adalah tahap untuk melakukan identifikasi terhadap aktor yang akan berinteraksi dengan *game* AGTA. Tabel 3.14 memperlihatkan aktor yang terlibat beserta penjelasannya yang merupakan hasil dari proses identifikasi aktor.

**Tabel 3. 14 Identifikasi aktor**

| Aktor | Deskripsi |
|---|---|
| Pemain | Pemain adalah pengguna yang dapat memainkan permainan AGTA, mengontrol *game*, mendapatkan akses untuk melihat info dan bantuan. |

2. Daftar Kebutuhan

   Daftar kebutuhan terdiri dari kebutuhan fungsional dan kebutuhan non-fungsional. Spesifikasi kebutuhan fungsional yang ada

di dalam permainan AGTA, ditunjukkan pada Tabel 3.15. Sedangkan spesifikasi kebutuhan non-fungsional ditunjukkan oleh tabel 3.16.

**Tabel 3. 15 Spesifikasi Kebutuhan Fungsional Permainan**

| ID | Kebutuhan | Aktor | Nama *Use Case* |
|---|---|---|---|
| F01 | *Game* harus dapat menampilkan *splash screen* berupa pengecekan koneksi kinect, informasi *game* dan logo UB, PTIIK dan GameLab sebelum masuk ke `dalam permainan. | Pemain | Melihat *Splash Screen* |
| F02 | Perangkat lunak harus menyediakan antarmuka untuk menampilkan berbagai menu utama permainan | Pemain | Melihat menu utama. |
| F03 | Perangkat lunak harus menyediakan antarmuka untuk proses memainkan permainan. | Pemain | Memainkan permainan. |
| F04 | *Game* harus menyediakan fasilitas untuk melihat bantuan, yaitu halaman yang menunjukkan cara kontrol permainan. | Pemain | Melihat Bantuan |
| F05 | *Game* harus menyediakan fasilitas untuk melihat informasi permainan, yaitu halaman yang menunjukkan informasi dan kegunaan permainan. | Pemain | Melihat Informasi |
| F06 | Perangkat lunak harus menyediakan proses untuk menghentikan sementara permainan yang berlangsung dan menampilkan menu pause yang memberikan pilihan untuk kembali ke permainan atau keluar dari *game*. | Pemain | Melihat *pause menu*. |
| F07 | Perangkat lunak harus menyediakan proses untuk memproduksi dan menampilkan | Pemain | Melihat item tangkapan |

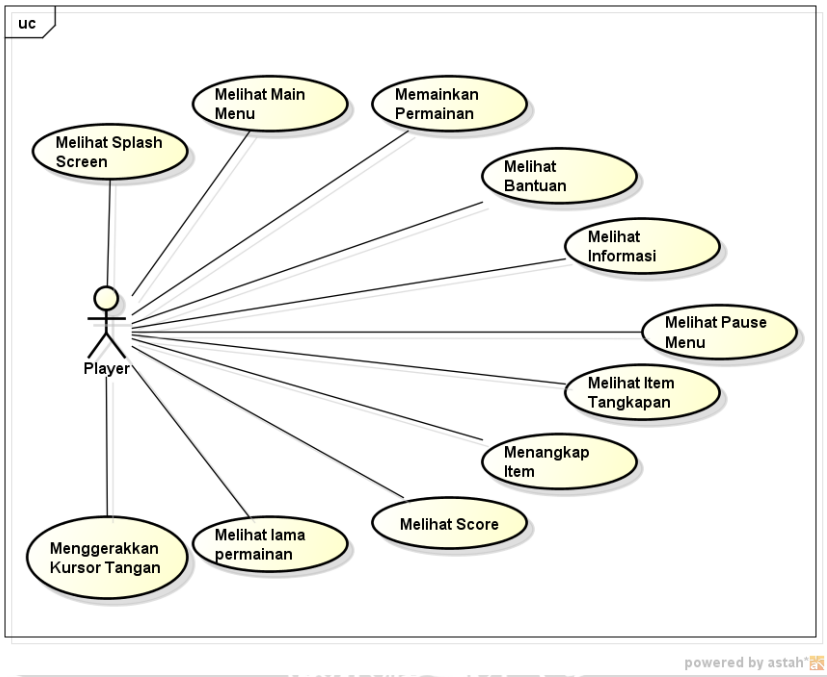
| | | | |
|---|---|---|---|
| | gambar hewan tangkapan secara *random*. | | |
| F08 | Perangkat lunak harus menyediakan proses untuk menangkap *target item* dan menghilangkan *target item* jika mengenai kursor tangan *player*. | Pemain | Menangkap Item |
| F09 | Perangkat lunak harus menyediakan komponen untuk menampilkan informasi skor pemain. | Pemain | Melihat skor pemain. |
| F10 | Perangkat lunak harus menyediakan komponen untuk menampilkan lama permainan. | Pemain | Melihat lama permainan |
| F11 | Perangkat lunak harus menyediakan proses untuk mendeteksi gerakan tangan pemain untuk menggerakkan kursor tangan. | Pemain | Menggerakkan kursor tangan. |

**Tabel 3.16 Spesifikasi Kebutuhan Non-Fungsional Permainan**

| ID | Deskripsi Kebutuhan |
|---|---|
| N01 | Perangkat lunak harus dapat menghasilkan rata – rata nilai *frame per second* lebih dari sama dengan 30 FPS |

3. Diagram *Use Case*

Diagram *use case* digunakan untuk menggambarkan fungsionalitas dari *game* AGTA. Aktor dalam *use case* ini adalah pemain. Diagram *use* case dibuat berdasarkan identifikasi kebutuhan yang telah dilakukan sebelumnya. Model diagram *use case game* AGTA ditunjukkan pada Gambar 3.6.




**Gambar 3. 6 Diagram *use case game* AGTA**

1. Skenario *Use Case* Melihat *Splash Screen* ditunjukkan pada tabel 3.17

**Tabel 3.17 Skenario *Use Case* Melihat *Splash Screen***

| Skenario Kasus pada Sistem | |
|---|---|
| ID | UC01 |
| Nama | Melihat *Splash Screen* |
| Tujuan | Untuk melihat *Splash Screen* dari *game* AGTA |
| Deskripsi | *Use Case* ini menjelaskan bagaimana *player* melihat *Splash Screen* dari *game* AGTA |
| Aktor | *Player* |
| Skenario Utama | |
| Kondisi Awal | *Game* belum berjalan |
| Aksi Aktor | Reaksi Sistem |
| Pemain menekan ikon permainan pada *desktop* untuk memulai eksekusi *game*. | Aplikasi dapat menampilkan *splash screen* berupa pengecekan koneksi kinect, informasi *game* dan logo UB, PTIIK dan GameLab secara berurutan. |
| Kondisi Akhir | Logo UB, PTIIK dan Game Lab ditampilkan pada layar |

2. Skenario *Use Case* Melihat Menu Utama ditunjukkan pada tabel 3.18

**Tabel 3. 18 Skenario *Use Case* Melihat Menu Utama**

| Skenario Kasus pada Sistem | |
|---|---|
| ID | UC02 |
| Nama | Melihat Menu Utama |
| Tujuan | Untuk melihat Menu Utama dari game AGTA |
| Deskripsi | *Use Case* ini menjelaskan bagaimana *player* melihat Menu Utama dari *game* AGTA |
| Aktor | *Player* |
| Skenario Utama | |
| Kondisi Awal | Game menampilkan *Splash Screen* |
| Aksi Aktor | Reaksi Sistem |
| Pemain melihat *splash screen* berupa pengecekan koneksi kinect dan logo UB, PTIIK dan GameLab. | Aplikasi dapat menampilkan *main menu* yang berupa gambar ikon *gameplay* 1, *gameplay* 2, *help* dan *information*. *Main menu* ditampilkan secara otomatis setelah *splash screen* selesai ditampilkan. |
| Kondisi Akhir | *Main Menu* ditampilkan pada layar |

3. Skenario *Use Case* Memainkan Permainan ditunjukkan pada tabel 3.19

**Tabel 3. 19 Skenario *Use Case* Memainkan Permainan**

| Skenario Kasus pada Sistem | |
|---|---|
| ID | UC03 |
| Nama | Memainkan Permainan |
| Tujuan | Untuk memainkan *game* AGTA |
| Deskripsi | *Use Case* ini menjelaskan bagaimana *player* memainkan *game* AGTA |
| Aktor | *Player* |
| Skenario Utama | |
| Kondisi Awal | *Game* sudah berjalan. Aplikasi menampilkan *in game screen.* Aplikasi menampilkan informasi |

| | |
|---|---|
| | skor pemain dan waktu permainan pada HUD *screen.* Aplikasi juga menampilkan gambar hewan tangkapan secara acak pada layar permainan |
| Aksi Aktor | Reaksi Sistem |
| 1. Pemain menggerakan tangan pada arah sumbu X dan Y<br>3. Pemain mengarahkan kursor pemain untuk menyentuh *target item* | 2. Kursor tangan yang berupa jaring pada *game Catch the Jelly* dan berupa ember pada *game Falling Party* mengikuti arah telapak tangan pemain pada arah sumbu X dan Y.<br>4. Aplikasi menghilangkan *target item* yang bertabrakan dengan kursor tangan *player* |
| Kondisi Akhir | Hewan tangkapan akan muncul terus menerus hingga waktu berakhir. |

4. Skenario *Use Case* Melihat Bantuan ditunjukkan pada tabel 3.20

**Tabel 3. 20 Skenario *Use Case* Melihat Bantuan**

| Skenario Kasus pada Sistem | |
|---|---|
| ID | UC04 |
| Nama | Melihat Bantuan |
| Tujuan | Untuk mengetahui cara bermain *game* AGTA |
| Deskripsi | *Use Case* ini menjelaskan bagaimana melihat bantuan untuk mengetahui cara bermain *game* AGTA |
| Aktor | *Player* |
| Skenario Utama | |
| Kondisi Awal | Aplikasi menampilkan tampilan awal Bantuan. Di dalamnya terdapat cara menggunakan kinect. |
| Aksi Aktor | Reaksi Sistem |
| 1. Pemain menggerakan tangan ke panah yang mengarah ke kanan | 2. Kursor tangan mengikuti arah telapak tangan pemain pada arah sumbu X dan Y.<br>3. Aplikasi akan menampilkan halaman selanjutnya |

| | |
|---|---|
| 4. Pemain menggerakan tangan ke panah yang mengarah ke kiri<br><br>6. Pemain menggerakan tangan ke tombol *exit* | 5. Aplikasi akan menampilkan halaman sebelumnya<br><br>7. Aplikasi akan keluar dari halaman bantuan dan akan menampilkan halaman menu utama |
| Kondisi Akhir | Aplikasi keluar dari halaman bantuan dan menampilkan halaman menu utama |

5. Skenario *Use Case* Melihat Informasi ditunjukkan pada tabel 3.21

**Tabel 3. 21 Skenario *Use Case* Melihat Informasi**

| Skenario Kasus pada Sistem | |
|---|---|
| ID | UC05 |
| Nama | Melihat Informasi |
| Tujuan | Untuk mengetahui informasi dan manfaat *game* AGTA |
| Deskripsi | *Use Case* ini menjelaskan bagaimana melihat mengetahui informasi dan manfaat *game* AGTA |
| Aktor | *Player* |
| Skenario Utama | |
| Kondisi Awal | Aplikasi menampilkan tampilan awal Informasi. |
| Aksi Aktor | Reaksi Sistem |
| 1. Pemain menggerakan tangan ke tombol *exit* | 2. Kursor tangan mengikuti arah telapak tangan pemain pada arah sumbu X dan Y.<br>3. Aplikasi akan keluar dari halaman bantuan dan akan menampilkan halaman menu utama |
| Kondisi Akhir | Aplikasi keluar dari halaman informasi dan menampilkan halaman menu utama |

6. Skenario *Use Case* Melihat *Pause Menu* ditunjukkan pada tabel 3.22

**Tabel 3. 22 Skenario *Use Case* Melihat *Pause Menu***

| Skenario Kasus pada Sistem | |
|---|---|
| ID | UC06 |
| Nama | Melihat *Pause Menu* |
| Tujuan | Untuk menghentikan permainan sementara |
| Deskripsi | *Use Case* ini menjelaskan bagaimana melihat *pause menu* untuk menghentikan sementara permainan |
| Aktor | *Player* |
| Skenario Utama | |
| Kondisi Awal | Aplikasi menampilkan *in-game screen* baik pada *game Catch The Jelly* maupun *game Falling Party*. |
| Aksi Aktor | Reaksi Sistem |
| 1. Pemain menggerakan tangan ke tombol *pause* pada *in-game screen* | 2. Kursor tangan mengikuti arah telapak tangan pemain pada arah sumbu X dan Y.<br>3. Aplikasi berhenti sementara dan menampilkan halaman *pause menu*. Halaman *pause menu* berisi tombol *resume*, *restart* dan *exit*. |
| 4. Pemain menggerakan tangan ke tombol *resume* | 5. Aplikasi akan keluar dari halaman *pause menu* dan akan melanjutkan permainan sebelumnya |
| 6. Pemain menggerakan tangan ke tombol *restart* | 7. Aplikasi akan keluar dari halaman *pause menu* dan akan memulai kembali permainan |
| 8. Pemain menggerakan tangan ke tombol *exit* | 9. Aplikasi akan keluar dari halaman *pause menu* dan akan menampilkan halaman menu utama |
| Kondisi Akhir | Aplikasi keluar dari halaman *pause menu* dan menampilkan halaman menu utama |

7. Skenario *Use Case* Melihat Item Tangkapan ditunjukkan pada tabel 3.23

**Tabel 3. 23 Skenario *Use Case* Melihat Item Tangkapan**

| Skenario Kasus pada Sistem | |
|---|---|
| ID | UC07 |
| Nama | Melihat Item Tangkapan |
| Tujuan | Untuk melihat item-item yang akan ditangkap pada *game* AGTA |
| Deskripsi | *Use Case* ini menjelaskan bagaimana *player* melihat lama permainan dari *game* AGTA |
| Aktor | *Player* |
| Skenario Utama | |
| Kondisi Awal | *Game* sudah berjalan. Aplikasi menampilkan *in game screen.* |
| Aksi Aktor | Reaksi Sistem |
| Pemain melihat *in game screen* dan memainkan permainan dengan menggerakkan tangan. | Aplikasi memproduksi dan menampilkan gambar item tangkapan secara *random.* |
| Kondisi Akhir | Item tangkapan ditampilkan pada *in game screen* |

8. Skenario *Use Case* Menangkap Item ditunjukkan pada tabel 3.24

**Tabel 3. 24 Skenario *Use Case* Menangkap Item**

| Skenario Kasus pada Sistem | |
|---|---|
| ID | UC08 |
| Nama | Menangkap Item |
| Tujuan | Untuk menangkap item-item yang ada pada *game* AGTA |
| Deskripsi | *Use Case* ini menjelaskan bagaimana *player* item pada *game* AGTA |
| Aktor | *Player* |
| Skenario Utama | |

| | |
|---|---|
| Kondisi Awal | *Game* sudah berjalan. Aplikasi menampilkan *in game screen.* |
| Aksi Aktor | Reaksi Sistem |
| Pemain melihat *in game screen* dan memainkan permainan dengan menggerakkan tangan. | Aplikasi menyediakan proses untuk menangkap *target item* dan menghilangkan *target item* jika mengenai kursor tangan *player*. |
| Kondisi Akhir | Item tangkapan ditampilkan pada *in game screen* |

9. Skenario *Use Case* Melihat *Score* Permainan ditunjukkan pada tabel 3.25

**Tabel 3. 25 Skenario *Use Case* Melihat *Score* Permainan**

| Skenario Kasus pada Sistem | |
|---|---|
| ID | UC09 |
| Nama | Melihat *Score* |
| Tujuan | Untuk melihat *score* permainan dari *game* AGTA |
| Deskripsi | *Use Case* ini menjelaskan bagaimana *player* melihat *score* dari *game* AGTA |
| Aktor | *Player* |
| Skenario Utama | |
| Kondisi Awal | *Game* sudah berjalan. Aplikasi menampilkan *in game screen.* |
| Aksi Aktor | Reaksi Sistem |
| Pemain melihat *in game screen* dan memainkan permainan dengan menggerakkan tangan. | Aplikasi menampilkan *score* permainan yang telah diraih pemain |
| Kondisi Akhir | *Score* permainan ditampilkan pada *in game screen* |

10. Skenario *Use Case* Melihat Lama Permainan ditunjukkan pada tabel 3.26

**Tabel 3. 26 Skenario *Use Case* Melihat Lama Permainan**

| Skenario Kasus pada Sistem | |
|---|---|
| ID | UC10 |
| Nama | Melihat Lama Permainan |
| Tujuan | Untuk melihat lama permainan dari *game* AGTA |
| Deskripsi | *Use Case* ini menjelaskan bagaimana *player* melihat lama permainan dari *game* AGTA |
| Aktor | *Player* |
| Skenario Utama | |
| Kondisi Awal | *Game* sudah berjalan. Aplikasi menampilkan *in game screen.* |
| Aksi Aktor | Reaksi Sistem |
| Pemain melihat *in game screen* dan memainkan permainan dengan menggerakkan tangan. | Aplikasi menampilkan lama permainan yang telah dilakukan pemain berupa menit dan detik. |
| Kondisi Akhir | Lama permainan ditampilkan pada *in game screen* |

11. Skenario *Use Case* Menggerakkan Kursor Tangan ditunjukkan pada tabel 3.27

**Tabel 3. 27 Skenario *Use Case* Menggerakkan Kursor Tangan**

| Skenario Kasus pada Sistem | |
|---|---|
| ID | UC11 |
| Nama | Menggerakkan Kursor Tangan |
| Tujuan | Untuk mengontrol pergerakan jaring pada *game* AGTA |
| Deskripsi | *Use Case* ini menjelaskan bagaimana *player* menggerakkan jaring pada *game* AGTA |
| Aktor | *Player* |

| Skenario Utama | |
|---|---|
| Kondisi Awal | *Game* sudah berjalan. Aplikasi menampilkan *in game screen* dan elemen-elemen *game.* |
| Aksi Aktor | Reaksi Sistem |
| Pemain melihat *in game screen* dan menggerakkan kursor tangan dengan menggerakkan tangan pada arah sumbu X dan Y. | Aplikasi mendeteksi gerakan tangan pemain dan menggerakkan kursor tangan sesuai dengan gerakan tangan pemain. |
| Kondisi Akhir | Kursor bergerak mengikuti gerakan tangan pemain |

### 3.3.2.2 *Class Diagram*

*Class Diagram* memberikan gambaran pemodelan elemen-elemen *class* yang membentuk sebuah perangkat lunak. *Class* didapatkan dengan menganalisis secara detail terhadap *use case* yang dimodelkan. Deskripsi dari *class diagram game* AGTA ditunjukkan pada Tabel 3.28. Deskripsi dari *interface game* AGTA ditunjukkan pada Tabel 3.29.

**Tabel 3.28 Deskripsi diagram *class game* AGTA**

| No | Class | Keterangan |
|---|---|---|
| 1 | `Game1` | *Class* yang berfungsi sebagai midlet permainan. |
| 2 | `SplashScreen` | *Class* yang berfungsi untuk mengatur *splash Screen* |
| 3 | `CatchTheJelly` | *Class* yang berfungsi untuk mengatur *screen* dan *gameplay game Catch The Jelly* |
| 4 | `FallingParty` | *Class* yang berfungsi untuk mengatur *screen* dan *gameplay game Falling Party* |
| 5 | `MainMenu` | *Class* yang berfungsi untuk mengatur *screen menu* |
| 6 | `Help` | *Class* yang berfungsi untuk mengatur *screen help* |
| 7 | `Info` | *Class* yang berfungsi untuk mengatur *screen Information* |

**Tabel 3.29 Deskripsi *Interface***

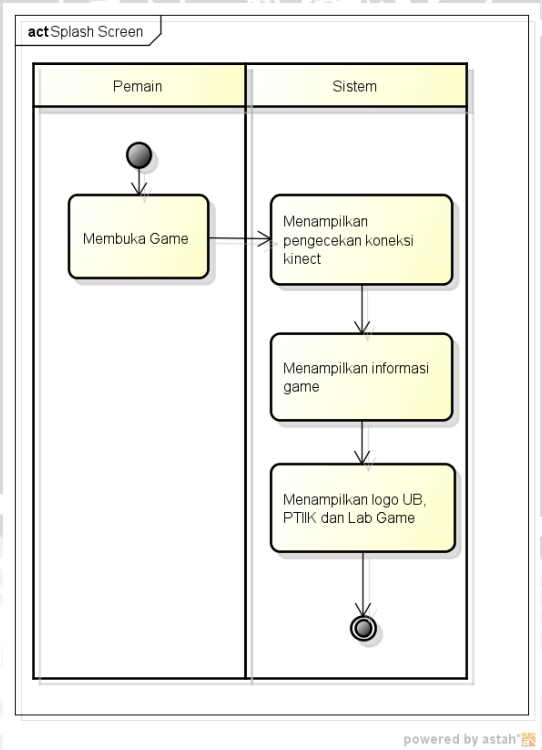
| No | *Interface* | Keterangan |
|---|---|---|
| 1 | `State` | *Interface* yang berfungsi untuk diimplementasikan pada semua *class screen*. |

### *3.3.2.3 Diagram Activity*

Diagram *activity* digunakan untuk menjelaskan urutan dari aktivitas. Diagram *activity* menggambarkan alur kerja mulai dari *starting point* hingga *finish point* dengan memberikan detail alur percabangan yang terdapat pada proses aktivitas.

1. Diagram Activity Melihat *Splash Screen*

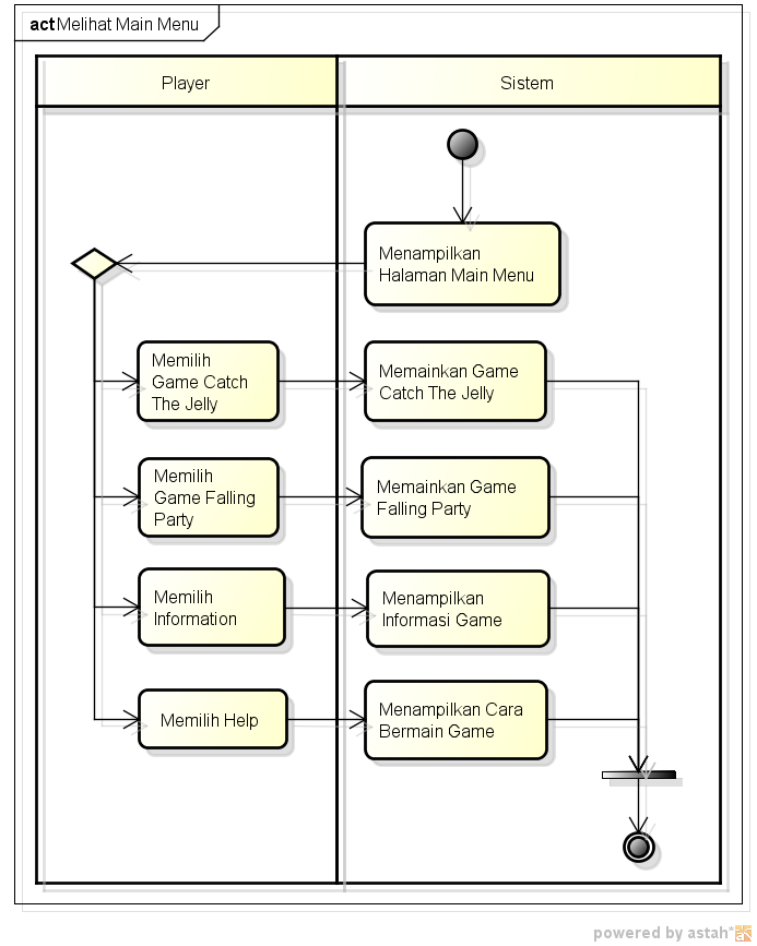
Diagram *activity* melihat *Splash Screen* dimulai dengan menampilkan pengecekan koneksi kinect kemudian menampilkan informasi *game* dan menampilkan logo UB, logo PTIIK dan logo LabGame. Diagram *activity* melihat *Splash Screen* ditunjukkan pada Gambar 3.7.




**Gambar 3. 7 Diagram *Activity* Melihat *Splash Screen***

2. Diagram *Activity* Melihat *Main Menu*

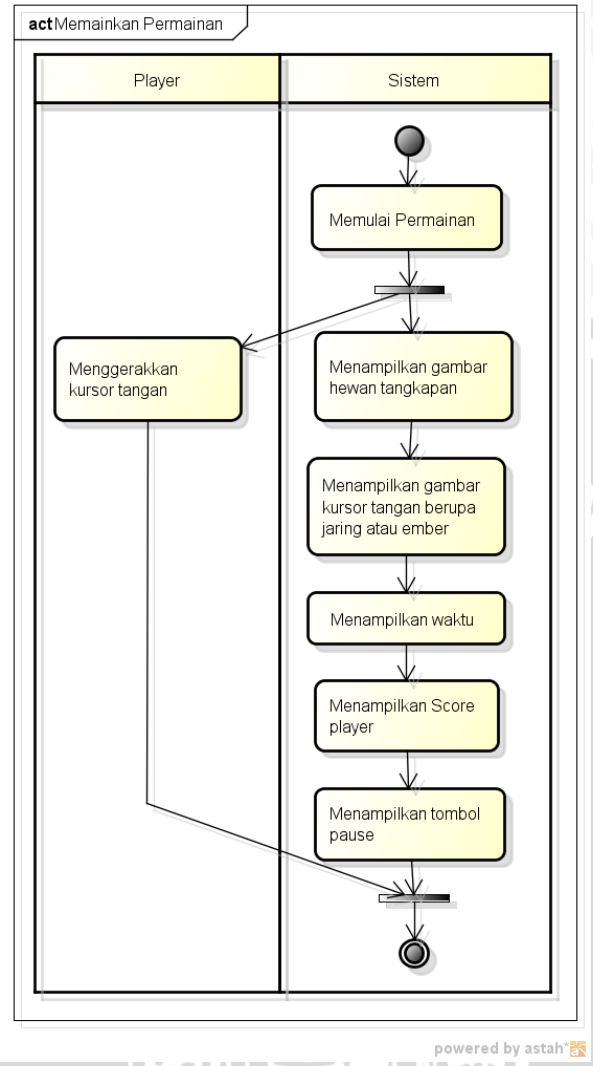
Diagram *activity* melihat *main menu* adalah aktivitas untuk memilih pilihan menu yang ada dalam menu utama. Pemain akan memilih aktivitas yang akan dilakukan dengan memilih ikon *gameplay* 1, *gameplay* 2, *help* dan *information*. Diagram *activity* melihat main menu ditunjukkan pada Gambar 3.8.




**Gambar 3. 8 Diagram *Activity* Melihat *Main Menu***

3. Diagram Activity Menjalankan Permainan Utama

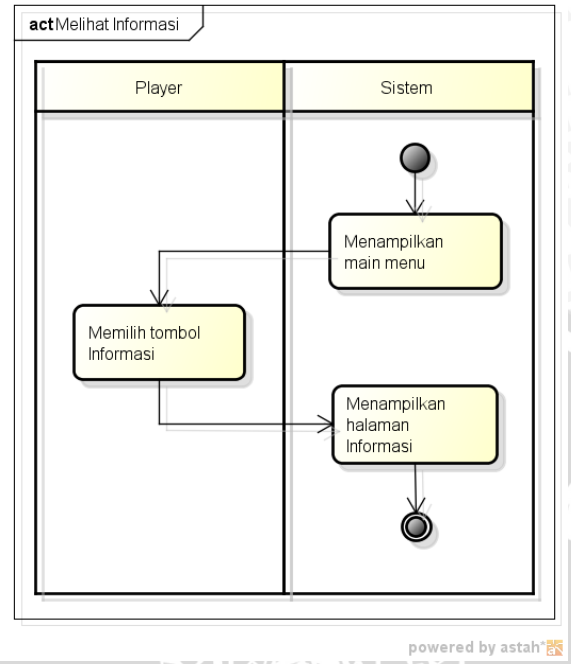
Diagram *activity* menjalankan permainan utama adalah aktivitas untuk mulai bermain dalam permainan utama. Diagram *activity* menjalankan permainan utama ditunjukkan pada Gambar 3.9.




**Gambar 3. 9 Diagram *Activity* Menjalankan Permainan Utama**

4. Diagram *Activity* Melihat Informasi

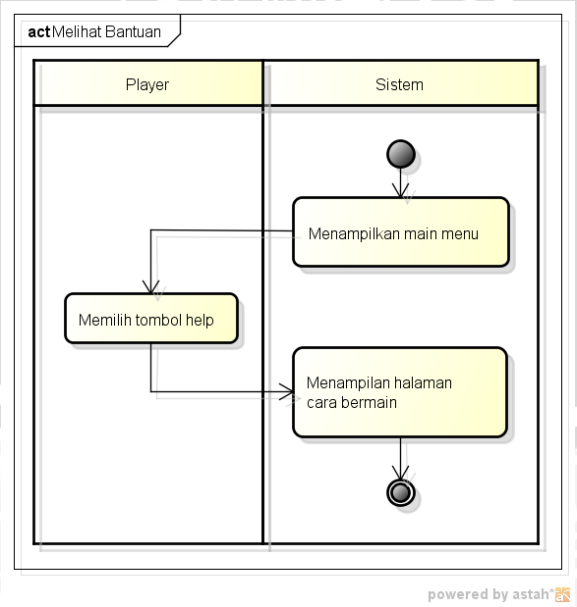
Diagram *activity* melihat Informasi dimulai dengan pemain masuk ke dalam *state* informasi permaian. Informasi berisi tentang semua hal yang berhubungan dengan informasi dan kegunaan permainan. Diagram *activity* melihat informasi ditunjukkan pada Gambar 3.10.




**Gambar 3. 10 Diagram *Activity* Melihat *Information***

5. Diagram Activity Melihat Bantuan

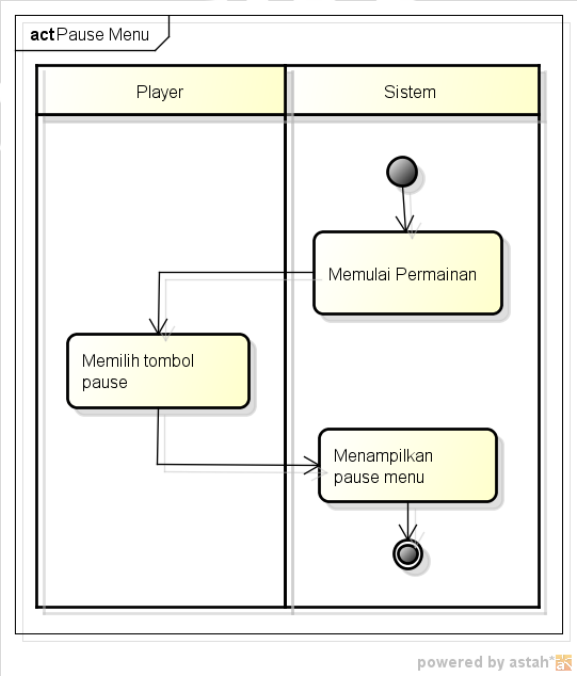
Diagram *activity* melihat bantuan dimulai dengan pemain masuk ke dalam *state* melihat bantuan. Bantuan berisi tentang cara bermain. Diagram *activity* melihat bantuan ditunjukkan pada Gambar 3.11.




**Gambar 3. 11 Diagram *Activity* Melihat Bantuan**

6. Diagram *Activity* Melihat *Pause Menu*

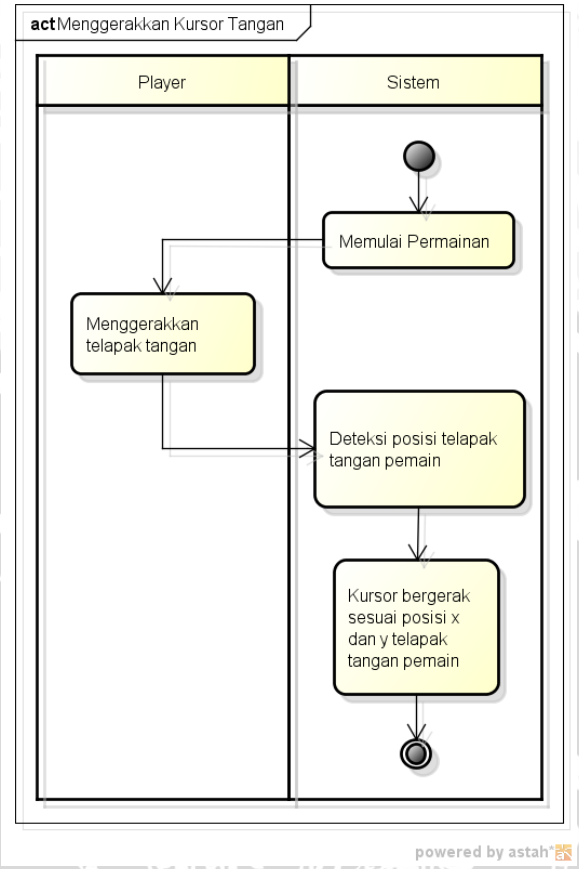
Pada gambar 3.11 menunjukan diagram *activity* melihat *pause menu*, yang menunjukan urutan proses dalam melihat *pause menu*. Penjelasan gambar 3.12 dalam proses menampilkan *pause menu* adalah pemain memilih tombol *pause* pada halaman permainan saat permainan telah dimulai. Sistem akan menampilkan halaman *pause menu* yang akan menghentikan sementara permainan. Halaman *pause menu* memberikan pilihan untuk kembali ke permainan atau ke menu utama.




**Gambar 3. 12 Diagram *Activity* Melihat *Pause Menu***

7. Diagram *Activity* Menggerakkan Kursor Tangan

Pada gambar 3.13 menunjukan *diagram activity* menggerakkan kursor tangan, yang menunjukan urutan proses dalam menggerakkan kursor tangan. Penjelasan gambar 3.13 dalam proses menggerakan kursor tangan adalah ketika sistem telah memulai permainan, pemain dapat menggerakan telapak tanggannya ke arah koordinat sumbu x dan y untuk menggerakkan kursor tangan. Sistem akan mengenali posisi telapak tangan pemain dan membuat kursor tangan bergerak sesuai koordinat posisi telapak tangan pemain.




**Gambar 3. 13 Diagram *Activity* Menggerakkan Kursor Tangan**

## 3.4 Implementasi *Game*

Implementasi pembangunan *game* dilakukan dengan mengacu kepada perancangan *game*. Implementasi pembuatan *user interface* dan karakter 2D menggunakan aplikasi Adobe Photoshop CS5 dan Adobe Ilustrator CS5, sedangkan pembangunan perangkat lunak *game* menggunakan bahasa pemrograman berbasis objek *C#* dengan *IDE Visual Studio 2010.*

Terapi dilakukan dengan memperkenalkan *game* kepada anak penderita ADHD. Penderita hanya perlu memainkan *game* tersebut. *Game* yang diberikan berupa *game* yang dapat meningkatkan fokus pemain mengingat pemain merupakan penderita gangguan perhatian dan hiperaktif.

## 3.5 Pengujian *Game*

Pengujian sistem merupakan hal penting yang bertujuan untuk menemukan kesalahan-kesalahan atau kekurangan-kekurangan pada

perangkat lunak yang diuji. Metode pengujian yang akan digunakan adalah *white-box testing*, *black-box testing* dan pengujian *frame per second* (FPS). Pengujian bermaksud untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak tersebut.

Pengujian juga dilakukan pada anak ADHD untuk mengetahui kefektifan perangkat lunak yang dibuat. Dibutuhkan data yang dapat menjadi ukuran keberhasilan perangkat lunak secara berkala. Penderita ADHD sangat rentan meninggalkan kegiatannya dalam waktu kurang dari 5 menit, oleh karena itu pengambilan data diperoleh dari lama permainan yang dilakukan oleh penderita. Pengambilan data dilakukan secara berkala agar dapat diketahui perkembangan fokus dan tingkat perhatian penderita. Semakin lama penderita bermain, maka semakin tinggi tingkat kefokusannya.

Pengujian terhadap anak ADHD dilakukan di sekolah anak berkebutuhan khusus "Zero to Five" di Jl Anggur 6 Malang. Anak ADHD yang mengikuti pengujian terapi ini sebanyak 3 anak dengan usia yang berbeda. Sebelum melakukan pengujian, *game* diperkenalkan terlebih dahulu kepada anak ADHD selama 1 pertemuan. Tujuannya untuk mengenalkan cara bermain dari *game* AGTA. Metode pengujian dilakukan sebanyak 5 pertemuan. Dalam sekali pertemuan, penderita akan diberikan sebanyak dua macam *game* yang berbeda. *Game* yang diberikan masing-masing hanya berdurasi 3 menit. Di dalam *game* tersebut telah disisipi perhitungan waktu saat pemain mulai bermain hingga pemain meninggalkannya. Setiap *game* yang diberikan semakin lama tingkat kesulitannya semakin tinggi, namun hanya sebatas kemampuan penderita. *Game* ini dimainkan oleh penderita secara berkala setiap tiga hari sekali.

# BAB IV
# IMPLEMENTASI

Bab ini membahas mengenai tahapan implementasi *game* AGTA berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan *game*.

## 4.1 Spesifikasi Sistem

Perangkat lunak ini dikembangkan dalam lingkungan implementasi yang terdiri dari perangkat keras dan perangkat lunak.

### 4.1.1 Spesifikasi Lingkungan Perangkat Keras

Spesifikasi perangkat keras komputer yang dipakai dalam proses pembangunan dijelaskan pada Tabel 4.1.

**Tabel 4. 1 Spesifikasi Lingkungan Perangkat Keras Komputer**

| Perangkat Keras | Spesifikasi |
|---|---|
| *Processor* | Intel(R) Core(TM)i3-2330M / 2.2 GHz |
| *Memory (RAM)* | 2 GB |
| *Harddisk* | 750 GB HDD |
| *VGA* | nVidia GForce 540M / 1GBB |
| *Monitor* | 14.0” HD WXGA |

### 4.1.2 Spesifikasi Lingkungan Perangkat Lunak

Spesifikasi perangkat lunak yang dipakai dalam proses pengembangan perangkat lunak dijelaskan pada Tabel 4.2.

**Tabel 4. 2 Spesifikasi Lingkungan Perangkat Lunak Komputer**

| Perangkat Lunak | Spesifikasi |
|---|---|
| *Operating Sistem* | Microsoft Windows 8 Enterprise 64-bit |
| *Integrated Development Environment* | Visual Studio 2010 |
| *Programming Language* | *C#* |
| *Framework* | XNA |
| *Graphic Editor* | Adobe Photoshop CS 5, Adobe Illustrator CS 5 |

## 4.2 Batasan-Batasan Dalam Implementasi

Beberapa batasan dalam mengimplementasikan perangkat lunak ini adalah sebagai berikut:

1. *Game single player*.
2. *Game* menggunakan bahasa Indonesia.
3. *Game* yang dibangun merupakan *game* dua dimensi (2D)
4. Pembuatan *game* AGTA meggunakan *framework* XNA
5. Pemain harus didampingi oleh pendamping saat memainkan *game* ini
6. Pencatatan hasil permainan dilakukan secara manual

## 4.3 Implementasi Prosedur Program

*Game* AGTA memiliki beberapa proses atau *method* yang terdapat pada beberapa *class*. Pada penulisan skripsi ini hanya dicantumkan prosedur dari beberapa proses (operasi) utama saja sehingga tidak semua *method* akan dicantumkan. Implementasi prosedur ini akan direpresentasikan dalam bentuk *pseudocode*.

### 4.3.2 Algoritma *Generate Random Item*

*Generate random item* pada *Catch the Jelly* merupakan proses yang akan merandom item yang akan muncul dalam *game*. Item yang muncul berupa ubur-ubur yang muncul dari berbagai sisi layar, baik atas, bawah, kiri dan kanan. Implementasi algoritma *generate random item* pada *Catch the Jelly* ditunjukkan pada tabel 4.3.

**Tabel 4. 3 Implementasi Algoritma *Generate Random Item***

| *Pseudocode Generate random item* |
|---|
| DECLARATION:<br>TYPE a IS int<br>used IS int[]<br>random IS Random<br>search IS boolean<br><br>DESCRIPTION:<br>1 a<- 0<br>2 search <- TRUE<br>3 WHILE(search)<br>4 IF (!used[a]) THEN |

```
5                   search <- FALSE
6           ELSE
7                   IF (a<19) THEN
8                           a <- a+1
9                   ENDIF
10          ENDIF
11  ENDWHILE

12  CASE (CALL random.Next(3)) OF
13          0 : CALL setNewPosition(CREATE OBJECT Vector2(0,CALL
    random.Next(640)),CREATE OBJECT Vector2(480,CALL random.Next(640)
    ),480,FALSE,a)
14          1 : CALL setNewPosition(CREATE OBJECT Vector2(480,CALL
    random.Next(640)),CREATE OBJECT Vector2(0,CALL random.Next(640)),
    0,FALSE,a)
15          2 : CALL setNewPosition(CREATE OBJECT Vector2(CALL
    random.Next(480),0),CREATE OBJECT Vector2(CALL random.Next(480),
    640),640, TRUE,a)
16          Default : CALL setNewPosition(CREATE OBJECT Vector2(CALL
    random.Next(480),640),CREATE OBJECT Vector2(CALL
    random.Next(480),0),0,TRUE, a)
17  ENDCASE
```

Penjelasan dari algoritma *generate random item* untuk pada tabel 4.3, yaitu

1. Baris 1 menjelaskan pemberian nilai 0 kepada variabel a. Fungsi variabel ini adalah untuk melakukan pengecekan id item yang tidak terpakai dari item yang tersedia.
2. Baris 2 menjelaskan pemberian nilai *true* pada variabel search sebagai penanda pencarian item. Jika ditemukan item yang tidak terpakai, nilai search akan diubah menjadi *false* untuk menghentikan pencarian. Penjelasan diatas merupakan penjelasan dari baris 2-11.
3. Baris 12 melakukan pengacakan nilai dari 0-2 untuk menentukan posisi item yang baru. Baris 13-16 menjelaskan pilihan posisi awal yang ditentukan oleh hasil pengacakan.

### 4.3.3 Algoritma Menentukan Posisi Awal Item

Pada *game Catch the Jelly*, item ubur-ubur berjalan ke segala arah secara acak. Pengacakan telah dilakukan pada algoritma *generate random item*. Pengacakan dilakukan pada posisi awal dan arah yang dimasukkan ke *method* setNewPosition. Implementasi algoritma menentukan posisi awal item ubur-ubur pada *game Catch the Jelly* ditunjukkan pada tabel 4.4.

**Tabel 4. 4 Algoritma Menentukan Posisi Awal Item**

| *Pseudocode* Menentukan Posisi Awal Item |
|---|

```
Methode : setNewPosition PARAMETER position, tujuan, hilang,
vertikal, indexOfObject

DECLARATION :
     TYPE position IS Vector2
     tujuan IS Vector2
     hilang IS int
     vertikal IS boolean
     indexOfObject IS int
     pos IS Vector2[]
     posTujuan IS Vector2[]
     posHilang IS int[]
     isVertikal IS int[]
DESCRIPTION:
  1  pos[indexOfObject]<- position
  2  posTujuan[indexOfObject] <- tujuan
  3  posHilang[indexOfObject] <- hilang
  4  isVertikal[indexOfObject] <- isVertikal
```

Penjelasan dari algoritma membuat item karakter untuk setNewPosition pada tabel 4.4, yaitu

1. Baris 1 menjelaskan pemberian nilai pos[] pada *index* ke- `indexOfObject`. Nilai ini merupakan posisi awal dari item ubur-ubur yang baru.
2. Baris 2 menjelaskan pemberian nilai posTujuan[] pada *index* ke- `indexOfObject`. Nilai ini merupakan posisi tujuan dari item ubur-ubur yang baru.
3. Baris 3 menjelaskan pemberian nilai posHilang[] pada *index* ke- `indexOfObject`. Nilai ini merupakan posisi hilangnya item ubur-ubur yang baru nanti.

4. Baris 4 menjelaskan pemberian nilai isVertikal[] pada *index* ke-`indexOfObject`. Nilai ini menentukan arah jalannya ubur-ubur yang baru, apakah ubur-ubur berjalan secara vertikal atau horizontal.

### 4.3.4 Algoritma Pergerakan Item

Algoritma Pergerakan Item merupakan proses yang akan menjelaskan langkah-langkah pergerakan item sesuai dengan arah dan posisi tujuan item. Implementasi Algoritma Pergerakan Pemain ditunjukkan pada tabel 4.5.

**Tabel 4. 5 Implementasi Algoritma Pergerakan Item**

| *Pseudocode* Pergerakan Item |
|---|
| DECLARATION:<br>TYPE i IS int<br>arahMusuh IS Vector2<br>posTujuan IS Vector2<br>pos IS Vector2<br><br>DESCRIPTION:<br>1 arahMusuh <- posTujuan[i]-pos[i]<br>2 pos[i] <- pos[i] + (arahMusuh * 2f) |

Penjelasan dari algoritma pergerakan item pada tabel 4.5, yaitu

1. Baris 1 menjelaskan inisialisasi nilai arahMusuh dengan cara mengurangi posis tujuan item dengan posisi item saat ini.
2. Baris 2 mengubah nilai posisi item dengan cara menambahkannya dengan 2*arahMusuh.

### 4.3.5 Algoritma Gerakan Rotasi Jaring

Algoritma gerakan kursor pemain merupakan proses yang dijalankan untuk merubah posisi kursor pemain sesuai koordinat x dan y yang diberikan pada parameter proses. Implementasi algoritma gerakan kursor pemain ditunjukkan pada tabel 4.6

**Tabel 4. 6 Implementasi Algoritma Gerakan Rotasi Jaring**

| *Pseudocode* Gerakan Rotasi Jaring |
|---|
| DECLARATION:<br>TYPE tangan IS Vector2<br>bahu IS Vector2<br>arah IS Vector2 |

```
    rightHandX IS int
    rightHandY IS int
    rightElbowX IS int
    rightElbowY IS int
    shrot IS float

DESCRIPTION:
 1   tangan <- CREATE OBJECT Vector2(rightHandX, rightHandY)
 2   bahu <- CREATE OBJECT Vector2(rightElbowX, rightElbowY)
 3   arah <- tangan-bahu
 4   IF(CALL arah.LengthSquared() > 100) THEN
 5         shrot <- CALL Math.Atan2(arah.Y, arah.X)
 6   ENDIF
```

Penjelasan dari algoritma gerakan rotasi jaring pada tabel 4.6., yaitu

1. Baris 1 dan 2 menjelaskan pemberian nilai variabel tangan dan bahu yang berisi posisi tangan dan posisi bahu.
2. Baris 3 memberikan nilai variabel arah dari nilai pengurangan tangan dengan bahu
3. Baris 5 memberikan nilai shrot sebagai besar rotasi jaring sesuai dengan gerakan tangan dari perhitungan nilai tan dari *vector* arah.

## 4.4 Implementasi *Game Elements*

Karakter atau elemen-elemen yang dibutuhkan pada *game* AGTA yang telah dijelaskan di *game design document* pada BAB Perancangan. Implementasi dari elemen permainan ditunjukkan pada tabel 4.7.

**Tabel 4. 7 Implementasi Karakter *Game* AGTA**

| *Game Object* | **Profil** |
|---|---|
|  | **Nama :** Gambar ubur-ubur<br>**Keterangan :** Beberapa gambar ubur-ubur dengan bentuk dan warna yang berbeda yang harus ditangkap pada *game Catch the Jelly* |

| | |
|---|---|
|  | **Nama :** Gambar kuda laut<br>**Keterangan :** Beberapa gambar kuda laut dalam berbagai warna yang harus ditangkap pada *game Falling Party* |
|  | **Nama :** Gambar bintang laut<br>**Keterangan :** Beberapa gambar bintang laut dalam berbagai warna yang harus ditangkap pada *game Falling Party* |
|  | **Nama :** *Hand Cursor*<br>**Keterangan :** *Hand Cursor* ini yang akan digerakan oleh gerakan tangan pemain pada *main menu* untuk memilih pilihan menu |
|  | **Nama :** Jaring<br>**Keterangan :** Jaring ini yang akan digerakan oleh gerakan tangan pemain untuk menangkap ubur-ubur dalam *game Catch the Jelly* |
|  | **Nama :** Ember<br>**Keterangan :** Ember ini yang akan digerakan oleh gerakan tangan pemain untuk menangkap hewan laut yang jatuh dalam *game Falling Party* |

## 4.5 Implementasi Antarmuka dan HUD

Implementasi antarmuka *game* AGTA ini terdiri dari *splash screen*, menu utama, *gameplay 1 (Catch the Jelly), gameplay 2 (Falling Party), information, help, pause menu* dan *end game screen.*

### *4.5.1 Splash Screen*

Halaman *Splash Screen* merupakan halaman awal yang diakses oleh *user*. Halaman ini dimulai dengan menampilkan pengecekan koneksi Kinect seperti pada gambar 4.1 kemudian menampilkan logo UB, logo PTIIK dan logo LabGame seperti pada gambar 4.2.

### *4.5.2 Main Menu*

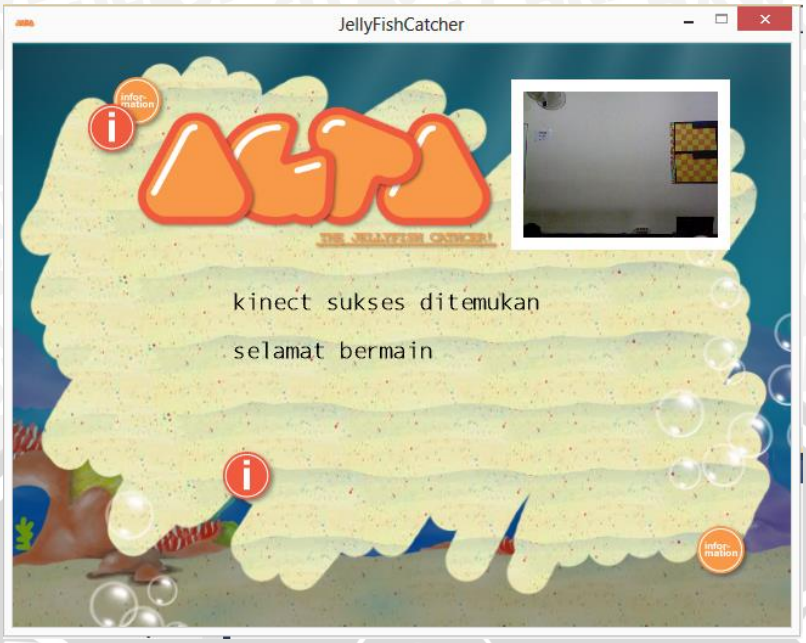
*Main menu* merupakan halaman menu yang akan tampil pertama kali ketika *game* dijalankan. Di halaman *main menu*, *button* yang bisa dipilih untuk menuju menu selanjutnya yaitu :

- ***Catch the Jelly,*** *button* ini digunakan untuk masuk ke *game Catch the Jelly*
- ***Falling Party,*** *button* ini digunakan untuk masuk ke *game Falling Party*
- ***Help,*** ini digunakan untuk masuk ke *help screen*
- ***Information,*** ini digunakan untuk masuk ke *information screen*
- ***Exit,*** *button* ini digunakan untuk keluar dari *game* AGTA

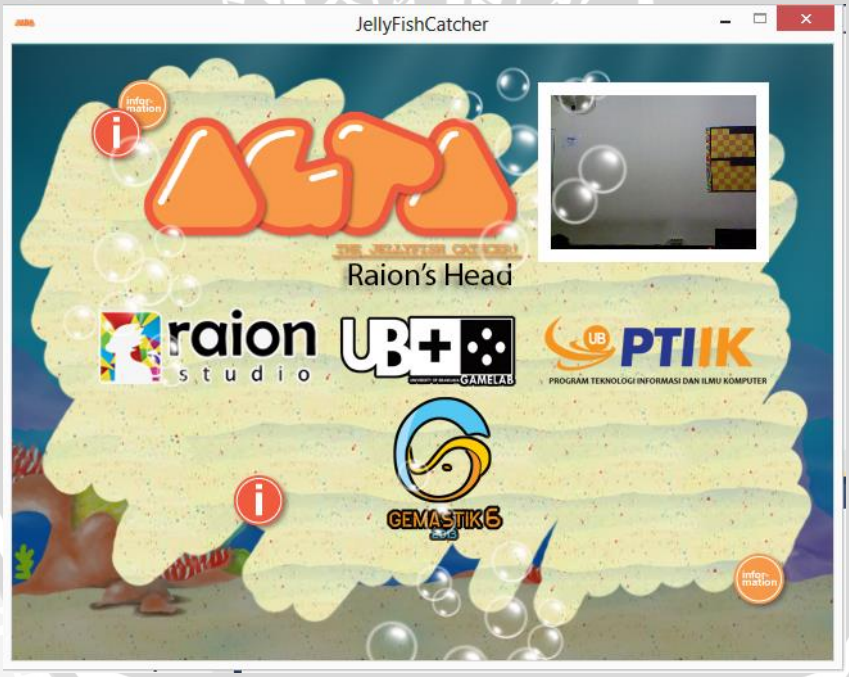
Implementasi *main menu* ditunjukkan oleh gambar 4.3.

### 4.5.3 Halaman *Help*

Tampilan halaman *Help* yang memberikan informasi cara bermain ditunjukkan pada gambar 4.4, 4.5 dan 4.6. Untuk berpindah ke halaman selanjutnya, *user* hanya perlu memilih tombol arah panah ke kanan. Dan memilih halaman sebelumnya dengan memilih tombol arah panah ke kiri.




**Gambar 4. 1 Implementasi halaman pengecekan koneksi Kinect**




**Gambar 4. 2 Implementasi Halaman logo institusi**


**Gambar 4. 3 Implementasi Halaman *Main Menu***

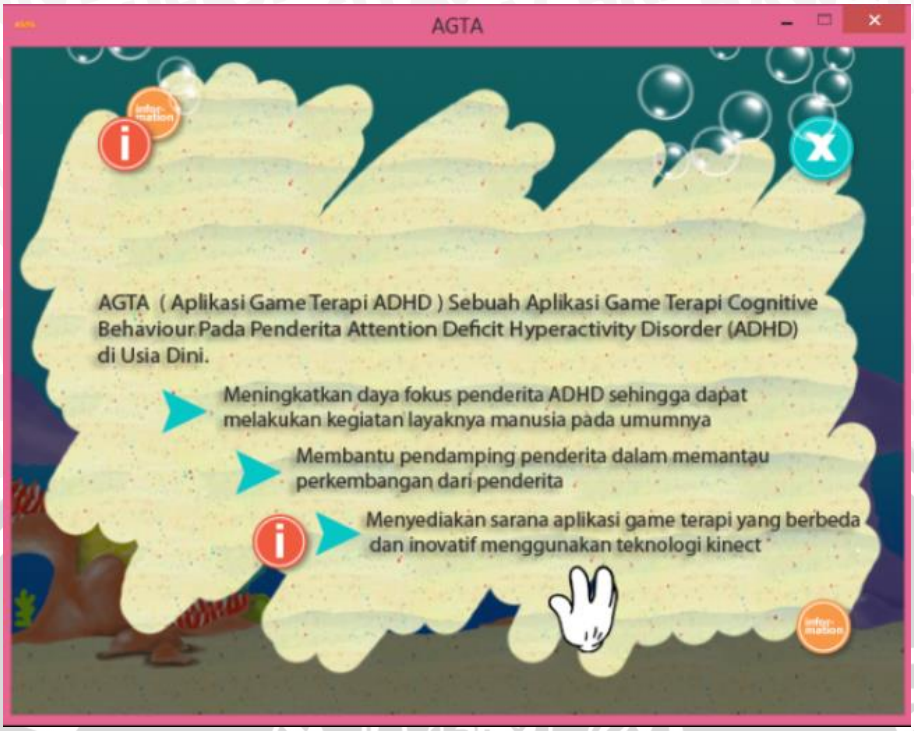

**Gambar 4. 4 Tampilan Halaman pertama *Help Screen***


**Gambar 4. 5 Tampilan Halaman kedua *Help Screen***


**Gambar 4. 6 Tampilan Halaman ketiga *Help Screen***

### 4.5.4 Halaman *Information*

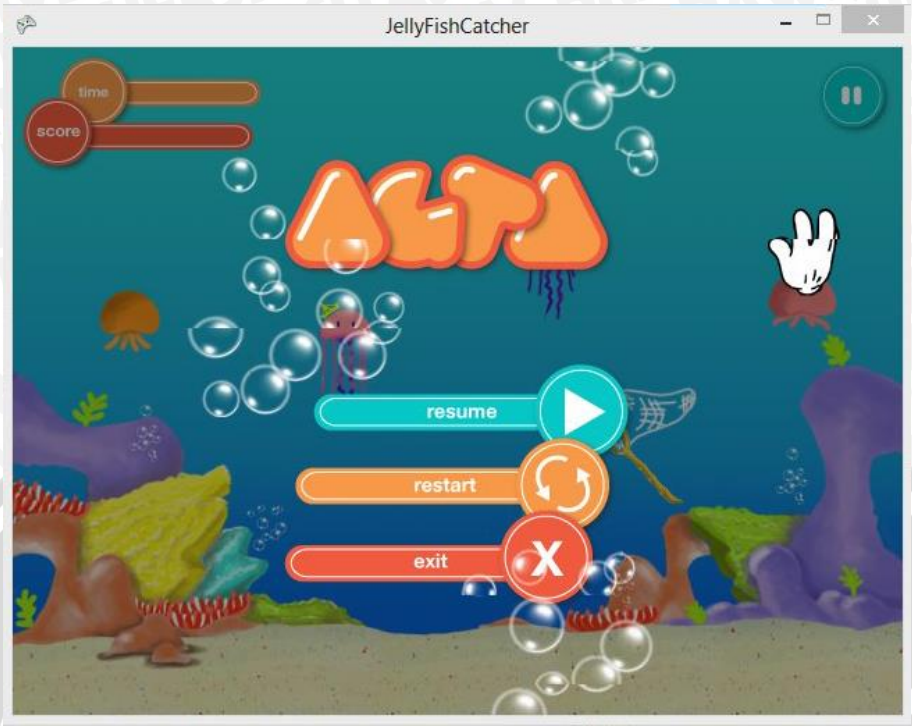
Tampilan halaman *Information* yang memberikan informasi dan kegunaan permainan ditunjukkan pada gambar 4.7.




**Gambar 4. 7 Tampilan Halaman *Information***

### 4.5.5 Halaman *Pause Menu*

Halaman *pause menu* merupakan halaman yang ditampilkan ketika *game* pada *state pause*. Pada *pause menu* untuk kembali melanjutkan permainan dengan memilih tombol *resume*, untuk memulai permainan dari awal memilih tombol *restart* sedangkan untuk keluar ke *main menu* dengan memilih tombol *exit*. Tampilan halaman *pause menu* ditunjukkan pada gambar 4.8.




**Gambar 4. 8 Tampilan Halaman *Pause Menu***

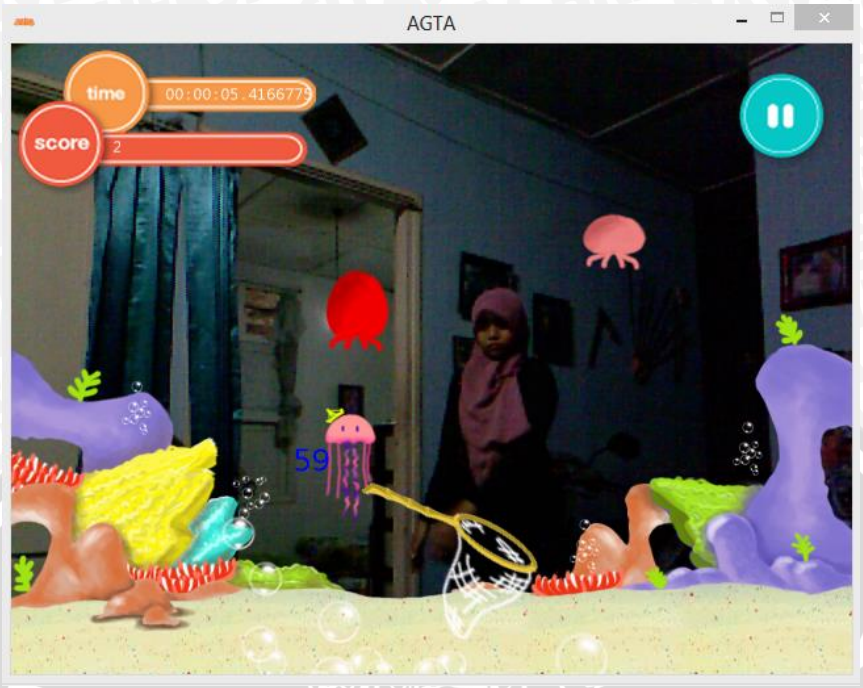
### 4.5.6 *Implementasi HUD* ( *Heads-Up Display* )

HUD berisi informasi mengenai waktu dan *score* pemain. Implementasi HUD ditunjukan pada gambar 4.9.


**Gambar 4. 9 Tampilan HUD**

### 4.5.7 *Game Arena*

Arena permainan AGTA berada di dalam laut. *Game* arena *Catch the Jelly* ditunjukan oleh gambar 4.10. *Game* arena *Falling Party* ditunjukan oleh gambar 4.11.




**Gambar 4. 10 *Game Arena Catch the Jelly***


**Gambar 4. 11 *Game* Arena *Falling Party***

# BAB V
# PENGUJIAN DAN ANALISIS

Bab ini membahas mengenai tahapan pengujian dan analisis *game* AGTA yang telah dikembangkan. Proses pengujian dilakukan melalui empat tahapan yaitu pengujian unit, pengujian integrasi, pengujian validasi, dan pengujian performa. Pada pengujian validasi akan digunakan teknik pengujian *Black-Box* (*Black-Box Testing*). Pada pengujian performa, subjek yang diuji adalah nilai *frames per second*.

## 5.1 Pengujian

Proses pengujian yang dilakukan melalui empat tahapan yaitu pengujian unit, pengujian integrasi, pengujian unit, pengujian integrasi, pengujian validasi dan pengujian performa.

### 5.1.1 Pengujian Unit

Pada pengujian unit digunakan metode *White Box Testing* dengan teknik *Basis Path Testing*. Pada teknik *Basis Path Testing* proses pengujian dilakukan dengan memodelkan algoritma pada sebuah *flowgraph*, menentukan *cyclometic complexcity* dan melakukan uji kasus untuk setiap *path* yang ada.

#### *5.1.1.1* Pengujian Unit Operasi updatePosisiMusuh()

Operasi updatePosisiMusuh() merupakan implementasi algoritma *update* posisi musuh. Operasi ini terdapat dalam kelas *CatchTheJelly*. Permodelan operasi updatePosisiMusuh() dalam bentuk *flowgraph* ditunjukkan pada tabel 5.1.

**Tabel 5. 1 Pemodelan *Flow Graph* Operasi updatePosisiMusuh()**

| *Pseudocode* | *Flow Graph* |
|---|---|
| Methode : updatePosisiMusuh PARAMETER i IS int<br>DECLARATION:<br>TYPE arahJalanMusuh IS Vector2<br>posTujuan IS Vector2[]<br>pos IS Vector2[]<br><br>DESCRIPTION :<br>1 arahJalanMusuh <- posTujuan[i] – pos[i]<br>2 CALL arahJalanMusuh.Normalize<br>3 Pos[i] <- pos[i] + (arahJalanMusuh*2)<br>4 END updatePosisiMusuh | Node (N)= 4<br>Edge (E)= 3<br>  |

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap operasi updatePosisiMusuh() menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2 .

V(G) = E – N + 2

= 3 – 4 + 2 = 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur *independent* yaitu:

Jalur 1: 1 – 2 – 3 - 4

Penentuan kasus uji untuk jalur *independent* tersebut dan hasil eksekusinya dijelaskan pada Tabel 5.2.

**Tabel 5. 2 *Test Case* Pengujian Operasi updatePosisiMusuh()**

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Mengupdate posisi item pada *index* ke-i | Posisi item pada *index* ke-i bertambah sesuai arah jalan item | Posisi item pada *index* ke-i bertambah sesuai arah jalan item |

### *5.1.1.2* Pengujian *Unit* Operasi backToDefault()

Operasi backToDefault() merupakan implementasi algoritma mengembalikan nilai *default* item yang tidak terpakai. Operasi ini terdapat pada kelas *CatchTheJelly*. Permodelan operasi backToDefault() dalam bentuk *flowgraph* ditunjukkan pada tabel 5.3.

Pemodelan ke dalam *flowgraph* yang telah dilakukan terhadap operasi backToDefault() menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2 .

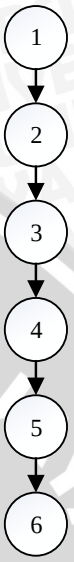
V(G) = E – N + 2

= 5 – 6 + 2 = 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur *independent* yaitu:

Jalur 1: 1 – 2 – 3 – 4 – 5 – 6

**Tabel 5. 3 Pemodelan *Flowgraph* Operasi backToDefault()**

| *Pseudocode* | *Flow Graph* |
|---|---|
| Methode : backToDefault PARAMETER i IS int<br><br>DECLARATION:<br>TYPE used IS boolean<br>isExplode IS boolean<br>pos IS Vector2[]<br>explodePosx IS int<br>explodePosy IS int<br><br>DESCRIPTION :<br>1 used[i] <- FALSE<br>2 pos[i] <- CREATE OBJECT Vector2(-100,-100)<br>3 explodePosx <- pos[i].X<br>4 explodePosy <- pos[i].Y<br>5 isExplode <- TRUE<br>6 END backToDefault | Node (N)= 6<br>Edge (E)= 5<br>  |

Penentuan kasus uji untuk jalur *independent* tersebut dan hasil eksekusinya dijelaskan pada Tabel 5.4.

**Tabel 5. 4 *Test Case* Pengujian operasi backToDefault()**

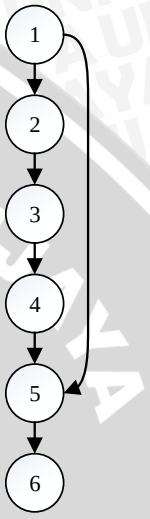
| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Mengembalikan nilai *default* item pada *index* ke-i | Nilai used item pada index ke-i menjadi *false*, posisi item pada index ke-i pada titik (-100,-100) , posisi ledakan sama dengan posisi item terakhir dan nilai isExplode menjadi *true* | Nilai used item pada *index* ke-i menjadi *false*, posisi item pada *index* ke-i pada titik -100,-100 , posisi ledakan sama dengan posisi item terakhir dan nilai isExplode menjadi *true* |

### *5.1.1.3* Pengujian *Unit* Operasi *collidesWJaring()*

Operasi collidesWJaring() merupakan implementasi algoritma pengecekan tabrakan item dengan jaring. Operasi ini terdapat dalam kelas *CatchTheJelly*.

Permodelan operasi collidesWJaring() dalam bentuk *flowgraph* ditunjukkan pada tabel 5.5.

**Tabel 5. 5 Pemodelan *Flow Graph* Operasi collidesWJaring()**

| *Pseudocode* | *Flow Graph* |
|---|---|
| Methode : collidesWJaring PARAMETER i IS int<br><br>DECLARATION:<br>TYPE gbrLengan IS Rectangle<br>score IS int<br><br>DESCRIPTION :<br>1 IF(CREATE OBJECT Rectangle(pos[i].X,pos[i].Y,75,75).Intersects(gbrLengan)) THEN<br>2 CALL backToDefault(i)<br>3 Score <- score+1<br>4 CALL gameMaster.mainKena()<br>5 ENDIF<br>6 END collidesWJaring | Node (N)= 6<br>Edge (E)= 6<br>  |

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap operasi collidesWJaring() menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2 .

V(G) = E – N + 2

= 6 - 6 + 2 = 2

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan dua buah basis set dari jalur *independent* yaitu:

Jalur 1: 1 – 5 – 6

Jalur 2: 1 – 2 – 3 – 4 – 5 – 6

Penentuan kasus uji untuk jalur *independent* tersebut dan hasil eksekusinya dijelaskan pada Tabel 5.6

**Tabel 5. 6 *Test Case* Pengujian Operasi collidesWJaring()**

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Jika item tidak bertabrakan dengan jaring | Tidak dilakukan penambahan score dan membuat item menjadi item yang tidak digunakan | Tidak dilakukan penambahan score dan item menjadi item yang tidak digunakan |
| 2 | Jika Item bertabrakan dengan jaring | Dilakukan penambahan score dan membuat item menjadi item yang tidak digunakan | Dilakukan penambahan score dan item menjadi item yang tidak digunakan |

### *5.1.1.4 Pengujian Unit Operasi updateRotJaring()*

Operasi updateRotJaring() merupakan implementasi algoritma penghitungan rotasi jaring. Operasi ini terdapat dalam kelas *CatchTheJelly*. Permodelan operasi updateRotJaring () dalam bentuk *flowgraph* ditunjukkan pada tabel 5.7.

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap operasi updateRotJaring() menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2 .

V(G) = E – N + 2

= 4 -4 + 2 = 2

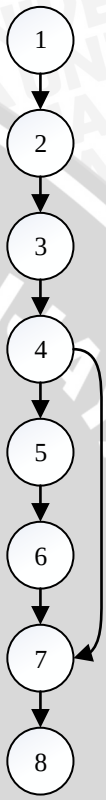
Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan dua buah basis set dari jalur *independent* yaitu:

Jalur 1: 1 – 2 – 3 – 4 – 7 – 8

Jalur 2: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8

Penentuan kasus uji untuk jalur *independent* tersebut dan hasil eksekusinya dijelaskan pada Tabel 5.8.

**Tabel 5. 7 Pemodelan *Flowgraph* Operasi updateRotJaring()**

| *Pseudocode* | *Flow Graph* |
|---|---|
| METHODE: updateRotJaring PARAMETER rHndX, rHndY, rElbwX, rElbwY<br><br>DECLARATION:<br>TYPE rHndX IS int<br>rHndY IS int<br>rElbwX IS int<br>rElbwY IS int<br>tangan IS Vector2<br>bahu IS Vector2<br>arahJaring IS Vector2<br>shrot IS float<br><br>DESCRIPTION:<br>1 Tangan <- CREATE OBJECT Vector2(rHndX, rHndY)<br>2 bahu <- CREATE OBJECT Vector2(rElbwX, rElbwY)<br>3 arahJaring <- tangan-bahu<br>4 IF(CALL arahJaring.LengthSquared() > 100) THEN<br>5 CALL arahJaring.Normalize()<br>6 Shrot <- CALL Math.Atan2(arahJaring.Y, arahJaring.X)<br>7 END IF<br>8 END updateRotJaring | Node (N)= 8<br>Edge (E)= 8<br>  |

**Tabel 5. 8 *Test Case* Pengujian Operasi updateRotJaring()**

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Parameter posisi tangan dan siku yang diberikan benar, namun hasil perhitungan *vector* arahJaring tidak lebih besar dari 100 | Tidak dilakukan penghitungan rotasi jaring kembali. Sehingga nilai rotasi jaring tetap seperti sebelumnya | Tidak dilakukan penghitungan rotasi jaring kembali. Nilai rotasi jaring tetap seperti sebelumnya |

| | | | |
|---|---|---|---|
| 2 | Parameter posisi tangan dan siku yang diberikan benar dan hasil perhitungan vector arahJaring lebih besar dari 100 | Dilakukan penghitungan rotasi jaring kembali | Dilakukan penghitungan rotasi jaring kembali |

**5.1.1.5 Pengujian Unit Operasi spawningObject()**

Operasi spawningObject () merupakan implementasi merandom item yang akan muncul dalam *game*. Operasi ini terdapat dalam kelas *CatchTheJelly*. Permodelan operasi spawningObject() dalam bentuk *flowgraph* ditunjukkan pada tabel 5.9.

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap operasi generateRandomItem() menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2 .

V(G) = E – N + 2

= 24 - 19 + 2 = 7

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan tujuh buah basis set dari jalur *independent* yaitu:

Jalur 1: 1 – 2 – 3 – 4 – 5 – 8 – 3 – 9 – 10 – 11 – 12 – 19

Jalur 2: 1 – 2 – 3 – 4 – 6 – 7 – 8 – 3 – 9 – 10 – 11 – 12 – 19

Jalur 3: 1 – 2 – 3 – 4 – 6 – 8 – 3 – 9 – 10 – 11 – 12 – 19

Jalur 4: 1 – 2 – 3 – 9 – 10 – 11 – 12 – 19

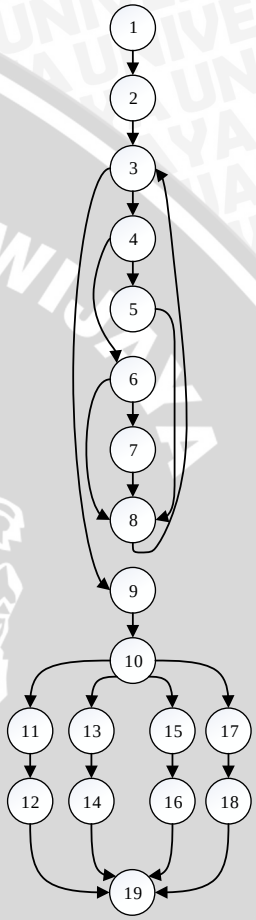
Jalur 5: 1 – 2 – 3 – 9 – 10 – 13 – 14 – 19

Jalur 6: 1 – 2 – 3 – 9 – 10 – 15 – 16 – 19

Jalur 7: 1 – 2 – 3 – 9 – 10 – 17 – 18 – 19

Penentuan kasus uji untuk jalur *independent* tersebut dan hasil eksekusinya dijelaskan pada Tabel 5.10

**Tabel 5. 9 Pemodelan *Flowgraph* Operasi spawningObject()**

| *Pseudocode* | *Flow Graph* |
|---|---|
| Methode : spawningObject<br><br>DECLARATION :<br>TYPE a IS int<br>search IS boolean<br>used[] IS boolean<br><br>DESCRIPTION:<br>1 a<- 0<br>2 search <- TRUE<br>3 WHILE(search)<br>4 IF (!used[a]) THEN<br>5 search <- FALSE<br>6 ELSE IF (a<19) THEN<br>7 a <- a+1<br>8 ENDIF<br>9 ENDWHILE<br>10 CASE (CALL random.Next(3)) OF<br>11 CASE(0) :<br>12 CALL setNewPosition(CREATE OBJECT Vector2(0,CALL random.Next(640)),CREATE OBJECT Vector2(480,CALL random.Next(640) ),480,FALSE,a)<br>13 CASE(1) :<br>14 CALL setNewPosition(CREATE OBJECT Vector2(480,CALL random.Next(640)),CREATE OBJECT Vector2(0,CALL random.Next(640)), 0,FALSE,a)<br>15 CASE(2) :<br>16 CALL setNewPosition(CREATE OBJECT Vector2(CALL random.Next(480),0),CREATE OBJECT Vector2(CALL random.Next(480), 640),640, TRUE,a)<br>17 Default :<br>18 CALL setNewPosition(CREATE OBJECT Vector2(CALL random.Next(480),640),CREATE OBJECT Vector2(CALL random.Next(480),0),0,TRUE, a)<br>19 ENDCASE | Node (N)= 19<br>Edge (E)= 24<br>  |

**Tabel 5. 10 *Test Case* Pengujian Operasi spawningObject()**

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Nilai search bernilai *true*. Nilai used pada index ke-a bernilai *false*. Nilai *random* pada *switch* bernilai 0 | Sistem akan menyelesaikan pencarian item yang tidak terpakai dan menentukan posisi dan arah berjalan item pada *index* ke-a yaitu berjalan horizontal dari kiri ke kanan. | Sistem menyelesaikan pencarian *index* yang tidak terpakai dan menentukan posisi dan arah berjalan item pada *index* ke-a yaitu berjalan horizontal dari kiri ke kanan. |
| 2 | Nilai search bernilai *true*. Nilai used pada *index* ke-a bernilai *true* dan nilai a kurang dari 19. Nilai *random* pada *switch* bernilai 0 | Sistem akan melakukan pencarian item yang tidak terpakai pada *index* berikutnya dan menentukan posisi dan arah berjalan item pada *index* ke-a yaitu berjalan horizontal dari kiri ke kanan. | Sistem melakukan pencarian item yang tidak terpakai pada *index* berikutnya dan menentukan posisi dan arah berjalan item pada *index* ke-a yaitu berjalan horizontal dari kiri ke kanan. |
| 3 | Nilai search bernilai *true*. Nilai used pada index ke-a bernilai *true* dan nilai a lebih dari 19. Nilai *random* pada switch bernilai 0 | Sistem akan menentukan posisi dan arah berjalan item pada *index* ke-a yaitu berjalan horizontal dari kiri ke kanan. | Sistem menentukan posisi dan arah berjalan item pada *index* ke-a yaitu berjalan horizontal dari kiri ke kanan. |
| 4 | Nilai search bernilai *false*. Nilai *random* | Sistem akan menentukan posisi dan arah berjalan item pada *index* ke-a | Sistem menentukan posisi dan arah berjalan item pada *index* ke-a |

| | | | |
|---|---|---|---|
| | pada *switch* bernilai 0 | yaitu berjalan horizontal dari kanan ke kiri. | yaitu berjalan horizontal dari kanan ke kiri. |
| 5 | Nilai search bernilai *false*. Nilai *random* pada *switch* bernilai 1 | Sistem akan menentukan posisi dan arah berjalan item pada *index* ke-a yaitu berjalan horizontal dari kiri ke kanan. | Sistem menentukan posisi dan arah berjalan item pada *index* ke-a yaitu berjalan horizontal dari kiri ke kanan. |
| 6 | Nilai search bernilai *false*. Nilai *random* pada *switch* bernilai 2 | Sistem akan menentukan posisi dan arah berjalan item pada index ke-a yaitu berjalan vertikal dari atas ke bawah. | Sistem menentukan posisi dan arah berjalan item pada index ke-a yaitu berjalan vertikal dari atas ke bawah. |
| 7 | Nilai search bernilai *false*. Nilai *random* pada *switch* bernilai 3 | Sistem akan menentukan posisi dan arah berjalan item pada index ke-a yaitu berjalan vertikal dari bawah ke atas. | Sistem menentukan posisi dan arah berjalan item pada index ke-a yaitu berjalan vertikal dari bawah ke atas. |

#### 5.1.1.6 Pengujian Unit Operasi checkingLimit()

Operasi checkingLimit() merupakan implementasi pengecekan item yang melewati garis batas atau keluar dari layar. Operasi ini terdapat dalam kelas *CatchTheJelly*. Permodelan operasi checkingLimit() dalam bentuk *flowgraph* ditunjukkan pada tabel 5.11.

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap operasi checkingLimit() menghasilkan jumlah kompleksitas siklomatis melalui persamaan V(G) = E – N + 2

V(G) = E – N + 2

= 30 - 21 + 2 = 11

**Tabel 5. 11 Pemodelan *Flowgraph* Operasi checkingLimit()**

| *Pseudocode* | *Flow Graph* |
|---|---|

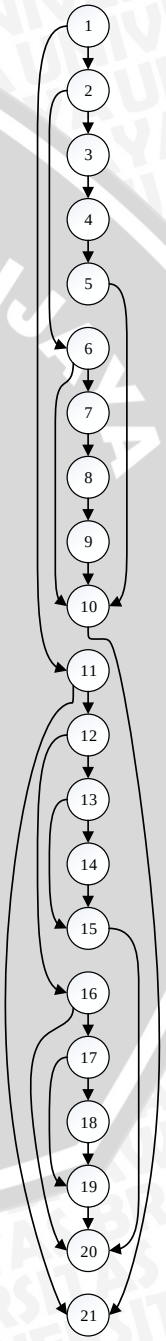
```
Methode : checkingLimit PARAMETER i IS int

DECLARATION :
     TYPE isVertikal[] IS boolean
     posHilang[] IS Vector2
     pos[] IS Vector2
     pinggirBawah IS Rectangle
     pinggirAtas IS Rectangle
     pinggirKanan IS Rectangle
     pinggirKiri IS Rectangle

DESCRIPTION:
 1  IF (isVertikal[i]) THEN
 2    IF (posHilang[i] = 640) THEN
 3      IF (CALL(CREATE OBJECT Rectangle(pos[i].X,
    pos[i].Y, 75, 75).Intersects(pinggirBawah))) THEN
 4        CALL backToDefault(i)
 5      ENDIF
 6    ELSE IF (posHilang[i] = 0)THEN
 7      IF (CALL(CREATE OBJECT Rectangle(pos[i].X,
    pos[i].Y, 75, 75).Intersects(pinggirAtas))) THEN
 8        CALL backToDefault(i)
 9      ENDIF
10    ENDIF
11  ELSE IF (!isVertikal[i]) THEN
12    IF (posHilang[i] = 480) THEN
13      IF (CALL(CREATE OBJECT Rectangle(pos[i].X,
    pos[i].Y, 75, 75).Intersects(pinggirKanan))) THEN
14        CALL backToDefault(i)
15      ENDIF
16    ELSE IF (posHilang[i] == 0) THEN
17      IF (CALL(CREATE OBJECT Rectangle(pos[i].X,
    pos[i].Y, 75, 75).Intersects(pinggirKiri))) THEN
18        CALL backToDefault(i)
19      ENDIF
20    ENDIF
21  ENDIF
```

Node (N)= 21

Edge (E)= 30



Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan sebelas buah basis set dari jalur *independent* yaitu:

Jalur 1: 1 – 2 – 3 – 4 – 5 – 10 – 21

Jalur 2: 1 – 2 – 3 – 5 – 10 – 21

Jalur 3: 1 – 2 – 6 – 7 – 8 – 9 – 10 – 21

Jalur 4: 1 – 2 – 6 – 7 – 9 – 10 – 21

Jalur 5: 1 – 2 – 6 – 10 – 21

Jalur 6: 1 – 11 – 12 – 13 – 14 – 15 – 20 – 21

Jalur 7: 1 – 11 – 12 – 13 – 15 – 20 – 21

Jalur 8: 1 – 11 – 12 – 16 – 17 – 18 – 19 – 20 – 21

Jalur 9: 1 – 11 – 12 – 16 – 17 – 19 – 20 – 21

Jalur 10: 1 – 11 – 12 – 16 – 20 – 21

Jalur 11: 1 – 11 – 21

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing – masing kasus uji dijelaskan pada tabel 5.12

**Tabel 5. 12 *Test Case* Pengujian Operasi checkingLimit()**

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Nilai isVertikal pada *index* yang ditentukan melalui parameter i bernilai *true*. Nilai posHilang pada *index* ke-i bernilai sama dengan 640 dan *rectangle* item bersentuhan dengan *rectangle* pinggirBawah. | Sistem akan melakukan pemanggilan backToDefault dengan parameter i. Sehingga item kembali ke posisi *default* dan variabel used ke-i bernilai *false* | Sistem melakukan pemanggilan backToDefault dengan parameter i. Item kembali ke posisi *default* dan variabel used ke-i bernilai *false* |
| 2 | Nilai isVertikal pada *index* yang ditentukan melalui parameter i | Sistem tidak melakukan pemanggilan | Sistem tidak melakukan pemanggilan |

| | | | |
|---|---|---|---|
| | bernilai *true*. Nilai posHilang pada *index* ke-i bernilai sama dengan 640 dan *rectangle* item tidak bersentuhan dengan *rectangle* pinggirBawah | backToDefault dengan parameter i. Sehingga item tetap pada posisinya | backToDefault dengan parameter i. Sehingga item tetap pada posisinya |
| 3 | Nilai isVertikal pada *index* yang ditentukan melalui parameter i bernilai *true*. Nilai posHilang pada *index* ke-i bernilai sama dengan 0 dan *rectangle* item bersentuhan dengan *rectangle* pinggirAtas. | Sistem akan melakukan pemanggilan backToDefault dengan parameter i. Sehingga item kembali ke posisi *default* dan variabel used ke-i bernilai *false* | Sistem melakukan pemanggilan backToDefault dengan parameter i. Item kembali ke posisi *default* dan variabel used ke-i bernilai *false* |
| 4 | Nilai isVertikal pada *index* yang ditentukan melalui parameter i bernilai *true*. Nilai posHilang pada *index* ke-i bernilai sama dengan 0 dan *rectangle* item tidak bersentuhan dengan *rectangle* pinggirAtas | Sistem tidak melakukan pemanggilan backToDefault dengan parameter i. Sehingga item tetap pada posisinya | Sistem tidak melakukan pemanggilan backToDefault dengan parameter i. Sehingga item tetap pada posisinya |
| 5 | Nilai isVertikal pada *index* yang ditentukan melalui parameter i bernilai *true*. Nilai posHilang pada *index* ke-i bernilai tidak sama | Sistem tidak melakukan pemanggilan backToDefault dengan parameter i. Sehingga item tetap | Sistem tidak melakukan pemanggilan backToDefault dengan parameter i. Sehingga item tetap |

| | | | |
|---|---|---|---|
| | dengan 0 dan 640 | pada posisinya | pada posisinya |
| 6 | Nilai isVertikal pada *index* yang ditentukan melalui parameter i bernilai *false*. Nilai posHilang pada *index* ke-i bernilai sama dengan 480 dan *rectangle* item bersentuhan dengan *rectangle* pinggirKanan. | Sistem akan melakukan pemanggilan backToDefault dengan parameter i. Sehingga item kembali ke posisi *default* dan variabel used ke-i bernilai *false* | Sistem melakukan pemanggilan backToDefault dengan parameter i. Item kembali ke posisi *default* dan variabel used ke-i bernilai *false* |
| 7 | Nilai isVertikal pada *index* yang ditentukan melalui parameter i bernilai *false*. Nilai posHilang pada *index* ke-i bernilai sama dengan 480 dan *rectangle* item tidak bersentuhan dengan *rectangle* pinggirKanan | Sistem tidak melakukan pemanggilan backToDefault dengan parameter i. Sehingga item tetap pada posisinya | Sistem tidak melakukan pemanggilan backToDefault dengan parameter i. Sehingga item tetap pada posisinya |
| 8 | Nilai isVertikal pada *index* yang ditentukan melalui parameter i bernilai *false*. Nilai posHilang pada *index* ke-i bernilai sama dengan 0 dan *rectangle* item bersentuhan dengan *rectangle* pinggirKiri. | Sistem akan melakukan pemanggilan backToDefault dengan parameter i. Sehingga item kembali ke posisi *default* dan variabel used ke-i bernilai *false* | Sistem melakukan pemanggilan backToDefault dengan parameter i. Item kembali ke posisi *default* dan variabel used ke-i bernilai *false* |
| 9 | Nilai isVertikal pada | Sistem tidak | Sistem tidak |

| | | | |
|---|---|---|---|
| | *index* yang ditentukan melalui parameter i bernilai *false*. Nilai posHilang pada *index* ke-i bernilai sama dengan 0 dan *rectangle* item tidak bersentuhan dengan *rectangle* pinggirKiri | melakukan pemanggilan backToDefault dengan parameter i. Sehingga item tetap pada posisinya | melakukan pemanggilan backToDefault dengan parameter i. Sehingga item tetap pada posisinya |
| 10 | Nilai isVertikal pada *index* yang ditentukan melalui parameter i bernilai *false*. Nilai posHilang pada *index* ke-i bernilai tidak sama dengan 0 dan 640 | Sistem tidak melakukan pemanggilan backToDefault dengan parameter i. Sehingga item tetap pada posisinya | Sistem tidak melakukan pemanggilan backToDefault dengan parameter i. Sehingga item tetap pada posisinya |
| 11 | Nilai isVertikal pada *index* yang ditentukan melalui parameter i tidak bernilai *true* maupun *false*. | Sistem tidak melakukan apapun. Sehingga item tetap pada posisinya | Sistem tidak melakukan apapun. Sehingga item tetap pada posisinya |

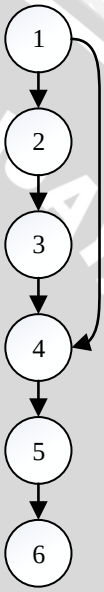
### 5.1.2 Pengujian Integrasi

Pengujian integrasi diterapkan pada proses yang mengintegrasikan fungsionalitas dari beberapa *class* untuk melakukan sebuah operasi tertentu. Pada pengujian integrasi yang dijadikan sebagai objek uji adalah *class* inti yang menggabungkan kinerja dari *class* – *class* lain. Pengujian integrasi yang dilakukan terhadap *class* inti ini menggunakan strategi *bottom-up*, dimana modul - modul yang diintegrasikan masing - masing diuji terlebih dahulu dalam pengujian unit dan kemudian bergerak menuju ke pengujian modul - modul kontrol yang mengintegrasikannya

### 5.1.2.1 Pengujian Integrasi Operasi pullingObject()

Operasi pullingObject() merupakan implementasi algoritma pindah ke halaman berikutnya. Operasi ini terdapat pada kelas *CatchTheJelly*. Pemodelan operasi pullingObject() dalam bentuk *flowgraph* ditunjukkan pada tabel 5.13.

**Tabel 5. 13 Pemodelan *Flowgraph* Operasi pullingObject()**

| *Pseudocode* | *Flow Graph* |
|---|---|
| METHODE: pullingObject PARAMETER i, rHandX, rHandY, rElbowX, rElbowY<br><br>DECLARATION:<br>TYPE i IS int<br>rHandX IS int<br>rHandY IS int<br>rElbowX IS int<br>rElbowY IS int<br>used IS boolean[]<br><br>DESCRIPTION:<br>1 IF (used[i]) THEN<br>2 CALL updateRotJaring(rHandX, rHandY, rElbowX, rElbowY)<br>3 CALL updatePosisiMusuh(i)<br>4 ENDIF<br>5 CALL collidesWJaring(i)<br>6 CALL checkingLimit(i) | Node (N)= 6<br>Edge (E)= 6<br>  <br> |

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap operasi updateRotJaring() menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2 .

V(G) = E – N + 2

= 4 -4 + 2 = 2

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan dua buah basis set dari jalur *independent* yaitu:

Jalur 1: 1 – 4 – 5 - 6

Jalur 2: 1 – 2 – 3 – 4 – 5 – 6

Penentuan kasus uji untuk jalur *independent* tersebut dan hasil eksekusinya dijelaskan pada Tabel 5.14.

**Tabel 5. 14 *Test Case* Pengujian Operasi pullingObject()**

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Parameter *index* item, posisi tangan dan siku yang diberikan benar, namun nilai used pada *index* ke-i bernilai *false* | posisiMusuh tidak di-*update* dan menjalankan pengecekan tubrukan dengan jaring dan pengecekan batas | posisiMusuh tidak di-*update* dan menjalankan pengecekan tubrukan dengan jaring dan pengecekan batas |
| 2 | Parameter *index* item, posisi tangan dan siku yang diberikan benar, nilai used pada *index* ke-i bernilai *true* | posisiMusuh di-*update* dan menjalankan pengecekan tubrukan dengan jaring dan pengecekan batas | posisiMusuh di-*update* dan menjalankan pengecekan tubrukan dengan jaring dan pengecekan batas |

### 5.1.3 Pengujian Validasi

Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang dibutuhkan. *Item - item* yang telah dirumuskan dalam daftar kebutuhan dan merupakan hasil analisis kebutuhan akan menjadi acuan untuk melakukan pengujian validasi. Pengujian validasi menggunakan metode pengujian *Black Box*, karena tidak diperlukan konsentrasi terhadap alur jalannya *Prosedur* program dan lebih ditekankan untuk menemukan konformitas antara kinerja sistem dengan daftar kebutuhan.

#### *5.1.3.1 Kasus Uji Validasi*

Keseluruhan kasus uji validasi dijelaskan pada tabel 5.15.

**Tabel 5. 15 Kasus Uji Validasi**

| No | Kasus Uji | Objek Uji | Prosedur dan Input | Kondisi yang diharapkan | Kondisi Gagal |
|---|---|---|---|---|---|
| 1 | Melihat *Splash Screen* | F01 | Pemain menekan ikon | Aplikasi dapat menampilkan *splash screen* | Aplikasi tidak dapat menampilkan |

| | | | | | |
|---|---|---|---|---|---|
| | | | permainan untuk memulai eksekusi *game*. | berupa pengecekan koneksi Kinect, informasi *game* dan logo UB, PTIIK dan GameLab | *splash screen*. Atau dapat menampilkan tetapi tidak sesuai urutan. |
| 2 | Melihat menu utama | F02 | *Game* memulai eksekusi menu utama setelah splash screen selesai ditampilkan | Aplikasi dapat menampilkan halaman menu utama. | Aplikasi tidak dapat menampilkan halaman menu utama. |
| 3 | Memulai permainan | F03 | Pemain memilih *icon Catch the Jelly* pada menu utama | Aplikasi dapat menampilkan *gameplay Catch the Jelly* | Aplikasi tidak dapat menampilkan *gameplay Catch the Jelly* |
| | | | Pemain memilih *icon Falling Party* pada menu utama | Aplikasi dapat menampilkan *gameplay Falling Party* | Aplikasi tidak dapat menampilkan *gameplay Falling Party* |
| 4 | Melihat *loading screen.* | F04 | Pemain memilih *icon Catch the Jelly* pada menu utama | Aplikasi menampilkan *loading screen* setelah *icon Catch the Jelly* terpilih dan | Aplikasi tidak menampilkan *loading screen* setelah *icon Catch the Jelly* terpilih, |

| | | | | | |
|---|---|---|---|---|---|
| | | | | aplikasi menampilkan *gameplay Catch the Jelly* setelah *loading screen* berakhir. | sehingga aplikasi juga tidak menampilkan *gameplay Catch the Jelly* setelah *loading screen* berakhir. |
| | | | Pemain memilih *icon Falling Party* pada menu utama | Aplikasi menampilkan *loading screen* setelah *icon Falling Party* terpilih dan aplikasi menampilkan *gameplay Falling Party* setelah *loading screen* berakhir. | Aplikasi tidak menampilkan *loading screen* setelah *icon Falling Party* terpilih, sehingga aplikasi juga tidak menampilkan *gameplay Falling Party* setelah *loading screen* berakhir. |
| | | | Pemain memilih *icon help* pada menu utama | Aplikasi menampilkan *loading screen* setelah *icon Help* terpilih dan aplikasi menampilkan | Aplikasi tidak menampilkan *loading screen* setelah *icon Help* terpilih, sehingga aplikasi juga |

| | | | | | |
|---|---|---|---|---|---|
| | | | | layar bantuan setelah *loading screen* berakhir. | tidak menampilkan layar bantuan setelah *loading screen* berakhir |
| | | | Pemain memilih *icon information* pada menu utama | Aplikasi menampilkan *loading screen* setelah *icon information* terpilih dan aplikasi menampilkan *information screen* setelah *loading screen* berakhir. | Aplikasi tidak menampilkan *loading screen* setelah *icon information* terpilih, sehingga aplikasi juga tidak menampilkan *information screen* setelah *loading screen* berakhir |
| 5. | Melakukan proses *pause*. | F05 | Pemain memilih tombol *pause* pada *in game.* | Aplikasi menghentikan sementara permainan yang berlangsung lalu menampilkan menu *pause* yang memberikan | Aplikasi tidak menghentikan sementara permainan yang berlangsung sehingga tidak menampilkan menu *pause* yang memberikan |

| | | | | | |
|---|---|---|---|---|---|
| | | | | pilihan untuk kembali ke permainan, *reset* atau keluar dari *game*. | pilihan untuk kembali ke permainan atau keluar dari *game*. |
| 6 | Menggerakkan kursor tangan. | F6 | Pemain menggerakan salah satu telapak tangan pada arah sumbu X dan Y saat *in game state.* | Kursor tangan pemain mengikuti arah telapak tangan pemain pada arah sumbu X dan Y. | Kursor tangan pemain diam atau tidak mengikuti arah telapak tangan pemainpada arah sumbu X dan Y. |
| 7 | Melihat skor pemain. | F7 | Pemain memainkan permainan dan masuk *in game state.* | Aplikasi dapat menampilkan informasi skor pemain pada HUD *screen.* | Aplikasi tidak dapat menampilkan informasi skor pemain pada HUD *screen.* |
| 8 | Melihat sisa waktu. | F8 | Pemain memainkan permainan dan masuk *in game state.* | Aplikasi dapat menampilkan informasi sisa waktu permainan pada HUD *screen.* | Aplikasi tidak dapat menampilkan informasi sisa waktu permainan pada HUD *screen* |
| 9 | Melihat gambar hewan | F9 | Pemain memainkan permainan | Aplikasi dapat menampilkan gambar hewan | Aplikasi tidak dapat menampilkan |

| | | | | | |
|---|---|---|---|---|---|
| | tangkapan | | dan masuk *in game state.* | tangkapan secara acak pada layar permainan | gambar hewan tangkapan secara acak pada layar permainan |
| 10 | Menghancurkan target item. | F10 | Pemain mengarahkan kursor pemain menyentuh *target item* pada *in game state.* | Aplikasi menghilangkan *target item* yang bertabrakan dengan kursor tangan *player*. | Aplikasi tidak menghilangkan *target item* yang bertabrakan dengan kursor tangan *player*. |
| 11 | Melihat cara bermain. | F11 | Pemain memilih menu melihat cara bermain dengan memilih *icon help*. | Aplikasi menampilkan halaman cara bermain. | Aplikasi tidak dapat menampilkan halaman cara bermain. |
| 12 | Melihat informasi *game*. | F12 | Pemain memilih menu melihat informasi dengan memilih *icon Information*. | Aplikasi menampilkan halaman informasi *game*. | Aplikasi tidak dapat menampilkan halaman informasi *game*. |
| 13 | Keluar permainan | F13 | Pemain memilih *icon* keluar. | Keluar dari aplikasi. | Tidak keluar dari aplikasi. |

### *5.1.3.2 Hasil Pengujian Validasi*

Hasil pengujian validasi dijelaskan pada Tabel 5.16.

**Tabel 5. 16 Hasil Uji Validasi**

| No | Kasus Uji | Objek Uji | Prosedur dan Input | Kondisi yang diharapkan | Hasil |
|---|---|---|---|---|---|
| 1 | Melihat *Splash Screen* | F01 | Pemain menekan ikon permainan untuk memulai eksekusi *game*. | Aplikasi dapat menampilkan *splash screen* berupa pengecekan koneksi Kinect, informasi *game* dan logo UB, PTIIK dan GameLab secara berurutan. | Valid |
| 2 | Melihat menu utama | F02 | *Game* memulai eksekusi menu utama setelah splash screen selesai ditampilkan | Aplikasi dapat menampilkan halaman menu utama. | Valid |
| 3 | Memulai permainan | F03 | Pemain memilih *icon Catch the Jelly* pada menu utama | Aplikasi dapat menampilkan *gameplay Catch the Jelly* | Valid |
| | | | Pemain memilih *icon Falling Party* pada menu utama | Aplikasi dapat menampilkan *gameplay Falling Party* | Valid |

| | | | | | |
|---|---|---|---|---|---|
| 4 | Melihat *loading screen.* | F04 | Pemain memilih *icon Catch the Jelly* pada menu utama | Aplikasi menampilkan *loading screen* setelah *icon Catch the Jelly* terpilih dan aplikasi menampilkan *gameplay Catch the Jelly* setelah *loading screen* berakhir. | Valid |
| | | | Pemain memilih *icon Falling Party* pada menu utama | Aplikasi menampilkan *loading screen* setelah *icon Falling Party* terpilih dan aplikasi menampilkan *gameplay Falling Party* setelah *loading screen* berakhir. | Valid |
| | | | Pemain memilih *icon help* pada menu utama | Aplikasi menampilkan *loading screen* setelah *icon Help* terpilih dan aplikasi menampilkan layar bantuan setelah *loading screen* berakhir. | Valid |
| | | | Pemain memilih *icon information* pada menu utama | Aplikasi menampilkan *loading screen* setelah *icon information* terpilih dan aplikasi menampilkan *information screen* setelah *loading screen* berakhir. | Valid |

| | | | | | |
|---|---|---|---|---|---|
| 5. | Melakukan proses *pause*. | F05 | Pemain memilih tombol *pause* pada *in game*. | Aplikasi menghentikan sementara permainan yang berlangsung lalu menampilkan menu *pause* yang memberikan pilihan untuk kembali ke permainan, *reset* atau keluar dari *game*. | Valid |
| 6 | Menggerakkan kursor tangan. | F6 | Pemain menggerakan salah satu telapak tangan pada arah sumbu X dan Y saat *in game state*. | Kursor tangan pemain mengikuti arah telapak tangan pemain pada arah sumbu X dan Y. | Valid |
| 7 | Melihat skor pemain. | F7 | Pemain memainkan permainan dan masuk *in game state*. | Aplikasi dapat menampilkan informasi skor pemain pada HUD *screen*. | Valid |
| 8 | Melihat sisa waktu. | F8 | Pemain memainkan permainan dan masuk *in game state*. | Aplikasi dapat menampilkan informasi waktu permainan pada HUD *screen*. | Valid |
| 9 | Melihat gambar hewan tangkapan | F9 | Pemain memainkan permainan dan masuk *in game state*. | Aplikasi dapat menampilkan gambar hewan tangkapan secara acak pada layar permainan | Valid |

| | | | | | |
|---|---|---|---|---|---|
| 10 | Menghancurkan target item. | F10 | Pemain mengarahkan kursor pemain menyentuh *target item* pada *in game state.* | Aplikasi menghilangkan *target item* yang bertabrakan dengan kursor tangan player. | Valid |
| 11 | Melihat cara bermain. | F11 | Pemain memilih menu melihat cara bermain dengan memilih *icon help*. | Aplikasi menampilkan halaman cara bermain. | Valid |
| 12 | Melihat informasi *game*. | F12 | Pemain memilih menu melihat informasi dengan memilih *icon Information.* | Aplikasi menampilkan halaman informasi *game*. | Valid |
| 13 | Keluar permainan | F13 | Pemain memilih *icon* keluar. | Keluar dari aplikasi. | Valid |

### 5.1.4 Pengujian Performa

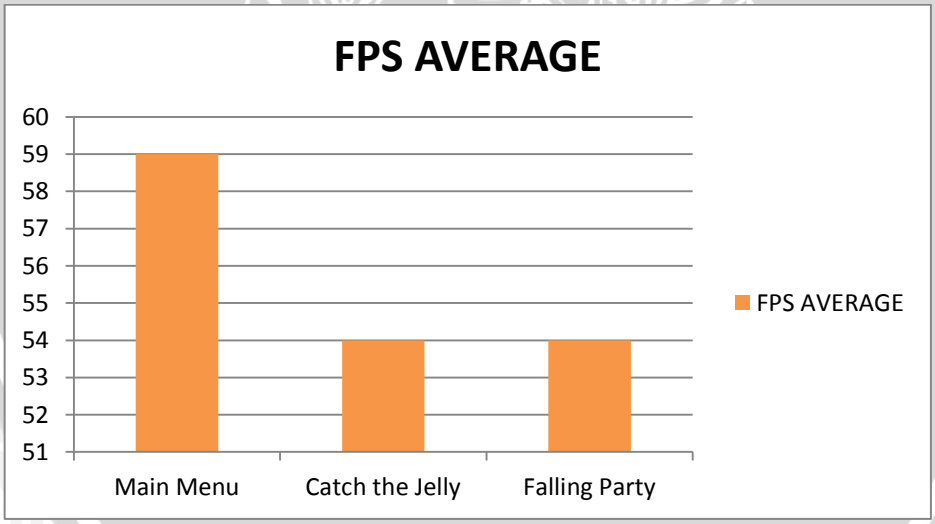
Pengujian performa *game* AGTA dilakukan dengan melihat FPS (*Frame Per Second*) yang dihasilkan oleh *game*. Pengujian FPS dilakukan pada perangkat komputer dengan. Pengujian kinerja dilakukan dengan melihat FPS yang dihasilkan pada setiap halaman. Adapun halaman yang digunakan dalam pengujian ini adalah *main menu*, halaman *in Game Catch the Jelly* dan halaman *in Game Falling Party*. Hasil pengujian kinerja *game* ditunjukkan pada Tabel 5.17. Tabel 5.18 menunjukan pengaruh FPS terhadap kinerja *game*. Gambar 5.1 menunjukan grafik FPS dari tiap halaman *game* yang dihasilkan dari pengujian.

**Tabel 5. 17 Hasil pengujian kinerja dengan melihat FPS**

| Spesifikasi Sistem | FPS | | |
|---|---|---|---|
| | ***Main Menu*** | ***Game Catch the Jelly*** | ***Game Falling Party*** |
| Intel(R) Core(TM)i3-2330M / 2.2 GHz, RAM 2 GB, 750 GB HDD, VGA nVidia GForce 540M / 1GBB, Monitor 14.0" HD WXGA | 59 | 54 | 54 |

**Tabel 5. 18 Pengaruh FPS Terhadap kinerja *Game***

| FPS | Kinerja *Game* |
|---|---|
| >= 30 FPS | Tampilan *game* halus dan lancar. |
| < 30FPS | Tampilan *game* patah – patah. |




**Gambar 5. 1 Grafik FPS *Average Game***

## 5.2 Pengujian pada Anak ADHD

Tujuan dari aplikasi ini adalah terciptanya aplikasi *game* terapi *cognitive behavior* untuk penderita ADHD yang dijadikan sarana bermain, beradaptasi, memperbaiki kemampuan untuk memecahkan masalah dan berkonsentrasi pada kegiatan yang sedang dilakukannya.

Hasil yang diperoleh adalah terciptanya *game* terapi *cognitive behaviour* yang dapat meningkatkan konsentrasi penderita ADHD.

Untuk mengetahui ketercapaian dari tujuan yang diharapkan yaitu dengan melakukan pengujian terhadap anak penderita ADHD secara berkala. Hasil pengujian *game* AGTA pada anak ADHD dapat dilihat pada tabel 5.19.

**Tabel 5. 19 Hasil Pengujian Lama Permainan Anak ADHD**

| PERTEMUAN KE- | LAMA PERMAINAN (detik) | | | | | |
|---|---|---|---|---|---|---|
| | ANAK KE-1 | | ANAK KE-2 | | ANAK KE-3 | |
| | G1 | G2 | G1 | G2 | G1 | G2 |
| 1 | 69 | 36 | 43 | 25 | 61 | 43 |
| 2 | 82 | 70 | 54 | 23 | 78 | 79 |
| 3 | 78 | 82 | 84 | 61 | 119 | 93 |
| 4 | 103 | 103 | 122 | 82 | 132 | 95 |
| 5 | 126 | 112 | 134 | 106 | 152 | 127 |

**Sumber :** Pengujian dan Analisis

Keterangan: G1: *Game* 1
G2 : *Game* 2

## 5.3 Analisis

Proses analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian *game* AGTA yang telah dilakukan. Proses analisis mengacu pada dasar teori sesuai dengan hasil pengujian yang didapatkan. Analisis dilakukan terhadap hasil pengujian di setiap tahap pengujian. Proses analisis yang dilakukan meliputi analisis hasil pengujian validasi, dan analisis hasil pengujian performa.

### 5.3.1 Analisis Hasil Pengujian Validasi

Berdasarkan kesesuaian antara hasil uji terhadap implementasi dan fungsionalitas *game* AGTA dengan hasil yang diharapkan dalam daftar kebutuhan *game*, dapat disimpulkan bahwa implementasi dan fungsionalitas *game* telah memenuhi kebutuhan yang telah dijabarkan dalam daftar kebutuhan.

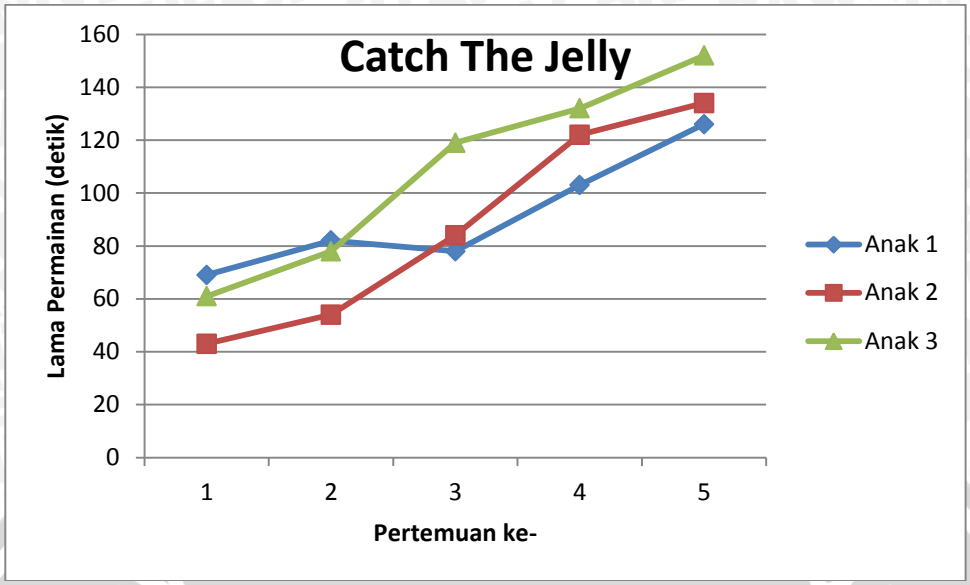
### 5.3.2 Analisis Hasil Pengujian Performa

Hasil analisa yang didapatkan dari pengujian integrasi yaitu :

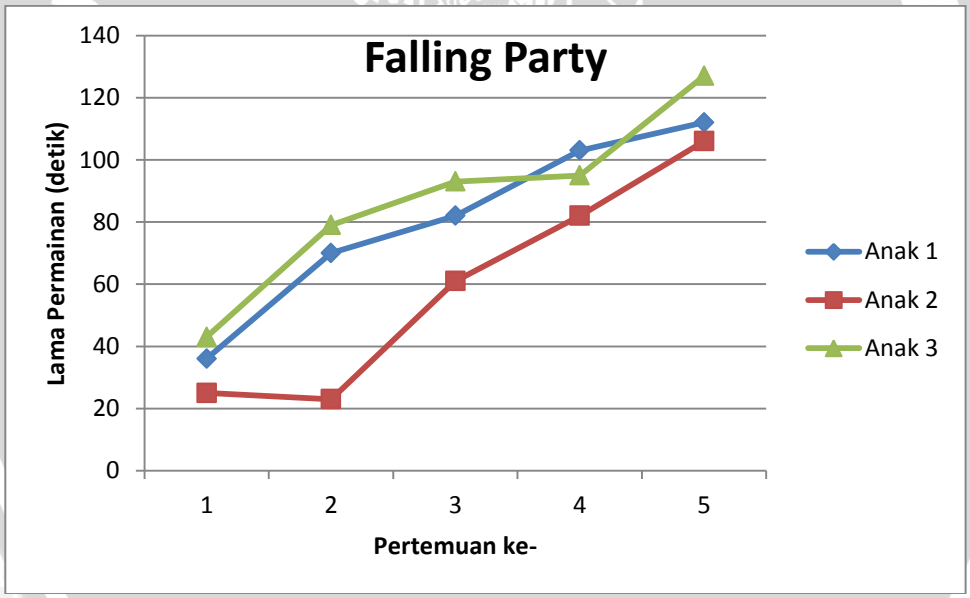
1. Proses analisa terhadap hasil pengujian performa dilakukan dengan melihat FPS yang dihasilkan pada halaman main menu, halaman *in Game Catch the Jelly* dan halaman *in Game Falling Party*.
2. Berdasarkan hasil pengujian peforma yang telah dilakukan, didapatkan rata - rata FPS terendah ada pada halaman *in game Catch the Jelly* dan *in-game Falling Party* yaitu sebesar 54 FPS, hal ini disebabkan karena pada halaman *in game* menampilkan banyak komponen gambar bergerak dan proses algoritma alur *game* diproses di halaman ini. FPS tertinggi didapatkan rata-rata sebesar 59 pada halaman main menu, hal ini disebabkan karena halaman main menu tidak menampilkan banyak komponen gambar bergerak dan algoritma yang dibutuhkan hanya sedikit. Sehingga dari hasil pengujian di dapatkan bahwa semakin kecil komponen yang diproses semakin besar pula hasil FPS yang di dapat dan semakin halus tampilan *game*.
3. Spesifikasi komputer yang disarankan untuk dapat menjalankan *game* AGTA adalah :
    a. Prosesor Intel Core I3 2.0 GHz atau yang lebih cepat.
    b. Memori minimal 2 GB RAM.
    c. VGA *Dedicated* dengan memori 752 MB.

### 5.3.3 Analisis Hasil Pengujian pada Anak ADHD

Dari tabel 5.15 dapat dilihat perkembangan penderita ADHD melalui lama permainan yang dilakukan. Perkembangan penderita dapat dilihat dari gambar 5.2 dan 5.3. Pada *game Catch the Jelly*, anak ke-1 lama bermainnya rata-rata meningkat, meskipun pada pertemuan ke-3 menurun. Hal ini dikarenakan anak ke-1 bermain *game* ini setelah melakukan banyak aktivitas. Sehingga anak kekurangan konsentrasi. Sedangkan anak ke-2 dan ke-3 lama bermainnya semakin meningkat.




**Gambar 5. 2 Grafik Hasil Pengujian *Game Catch The Jelly* pada Anak ADHD**




**Gambar 5. 3 Grafik Pengujian *Game Falling Party* pada Anak ADHD**

Pada *game Falling Party*, anak ke-1 dan ke-3 lama bermainnya semakin meningkat. Sedangkan anak ke-2 juga rata-rata meningkat, namun pada pertemuan ke-2 lama bermainnya hampir sama.

Dari setiap pertemuan, anak ADHD rata-rata semakin lama dalam bermain. Baik pada *game Catch the Jelly* maupun *game Falling Party*. Semakin lama penderita bermain, maka semakin baik tingkat fokus dan perhatian penderita.

Pada saat pengujian *game* pada anak ADHD terdapat kendala yang ditemukan pada *game* antara lain jarak tangkap kinect yang terbatas, yaitu antara 1,2 sampai dengan 3,5 meter saja, *game* hanya dapat dimainkan oleh 1 orang saja karena kemampuan kinect mendeteksi tubuh manusia hanya terbatas 1 orang saja dan *game* akan terganggu jika dimainkan dalam ruangan yang gelap karena Kinect tidak dapat mendeteksi bagian tubuh manusia di tempat yang gelap.

# BAB VI
# PENUTUP

## 6.1 Kesimpulan

Dari hasil pengamatan selama perancangan, implementasi, dan proses pengujian *game* yang dilakukan, diambil kesimpulan sebagai berikut:

1. *Game* AGTA telah berhasil diimplementasikan sebagai *game* terapi *cognitive behaviour* untuk anak ADHD. Unsur terapi dari *game* ini terdapat pada *gameplay* yang memerlukan konsentrasi dan fokus untuk menangkap item-item bergerak pada *game*. AGTA menerapkan teknik interaksi natural untuk menggerakkan kursor. Dimana kursor digunakan dalam *gameplay* sebagai penangkap item-item bergerak yang berupa hewan-hewan laut pada *game*. *Game* AGTA dirancang dari *game concept design* dan *technical design*. Implementasi *game* AGTA menggunakan *framework* XNA dan *library* Kinect untuk XNA digunakan untuk mengimplementasikan sensor kinect dalam *game*.
2. Berdasarkan hasil pengujian unit dan pengujian integrasi dengan menggunakan metode *whitebox testing*, dari 6 unit operasi dan 1 integrasi operasi didapatkan jumlah jalur pada logika setiap *method* dan prosedur telah sesuai dengan perhitungan *cyclomatic complexity* dan setiap jalur *independent* dengan hasil pengujian seluruh operasi sesuai dengan hasil yang diharapkan.
3. Berdasarkan hasil pengujian validasi menggunakan metode *blackbox testing*, dari 13 kasus uji keseluruhan fungsional aplikasi permainan dapat berjalan sesuai daftar kebutuhan yang telah dibuat dan dimodelkan dalam *use case*.
4. Pengujian kinerja *game* AGTA menghasilkan kesimpulan bahwa rata – rata FPS terendah ada pada halaman *in game Catch the Jelly* dan *in game Falling Party* yaitu 54. Sehingga *game* AGTA dapat berjalan optimal ketika *game* berjalan pada komputer atau *laptop* yang dapat menghasilkan *frame per second* lebih besar atau sama dengan 30FPS ketika memainkan *game* AGTA.
5. Berdasarkan hasil pengujian perkembangan penderita ADHD melalui lama permainan dari ketiga anak dan dari lima pertemuan yang dilakukan, didapatkan rata-rata setiap anak semakin lama dalam bermain. Dapat

disimpulkan bahwa fokus penderita dapat meningkat karena semakin lama pederita bermain, semakin baik tingkat fokus dan perhatian penderita.

## 6.2 Saran

Saran untuk pengembangan *game* AGTA lebih lanjut antara lain:

1. Penambahan fitur *Voice Recognition* dengan memanfaatkan *microphone* yang sudah *include* di dalam kinect.
2. Penambahan *gameplay* untuk meningkatkan aktivitas sensorik anak ADHD dengan menggunakan perintah suara
3. Penambahan fitur analisa perkembangan fokus anak ADHD dengan memanfaatkan data hasil bermain
4. Penambahan fitur *highscore* dan *reward* agar pemain lebih tertarik dengan *game* AGTA.

## DAFTAR PUSTAKA


[ANO-11] Anonim. 2011. "Kinect Nat Geo TV".www.relentless.co.uk [7 Agustus 2013].

[ARA-12] Arisandi, Adhi dkk.2012. "Visualisasi Gerakan Objek 3D pada *Augmented Reality* dangan Deteksi Tumbukan Berbasis *Bounding Box*" . Surabaya:Institut Teknologi Sepuluh Nopember

[BOU-11] Bouvrie , Bas des. 2011. *Improving RGBD Indoor Mapping with IMU data. Thesis*, Netherlands : Master of Science, Delft University of Technology.

[CHA-11] Chandler, Heather Maxwell & Chandler, Rafael. 2011. *Fundamentals of Game Development*. Jones & Bartlett Learning.

[ELJ-08] Ellis O., Jeanne.2008.Psikologi Pendidikan Membantu Siswa Tumbuh dan Berkembang. Jakarta: Erlangga

[HAM-06] Hamilton, Kim & Miles, Russell. 2006. *Learning UML* 2.0. O'Reilly Media, Inc

[HAR-04] Hariyanto, Bambang. 2004. Rekayasa Sistem Berorientasi Objek. Informatika Bandung.

[IND-07] Indah Wahyuni, Lina. 2007. Terapi Bermain Untuk Meningkatkan Konsentrasi Anak Dengan Gangguan Adhd. Malang:Universitas Muhammadiyah Malang.

[KLE-08] Klopfer, Eric.2008.*Augmented Learning Research and Design of Mobile Educational Games*.Massachusetts: Massachusetts Institute of Technology

[LIU-06] Liu, Liping & Roussev, Boris. 2006. *Management of The Object-Oriented Development Process*. Idea Group Publishing.

[MIL-94] P. Milgram and A. F. Kishino, 1994, Taxonomy of Mixed Reality Visual Displays, IEICE Transactions on Information and Systems

[PED-08] Pedersen, Roger E. 2008. *Game Design Foundation, 2ndEdition*. Wordware Publishing, Inc

[PRE-01] Pressman, Roger S. 2001. *Software Engineering: a practitioner's approach,7th edition*. Mc Graw Hill.

[ROG-10] Rogers, Scott. 2010. *Level Up, The Guide To Great Video Game Design*. John Wiley & Sons, Ltd

[ROM-11] Miles, Rob. 2011. C# *Programming*. Cottingham Road: The University of Hull

[ROM-12] Miles, Rob. 2012. *Using Kinect for Windows with XNA*

[ROU-10] Rouse, Richard. 2010. *Game Design: Theory and Practice*, Second Edition. Jones & Bartlett Learning.

[SHB-05] Sheldon, Brian.2005. *Cognitive Behavioural Therapy Research, practice and philosophy* . New York:Routledge

# LAMPIRAN

## *L.1. User Guide*

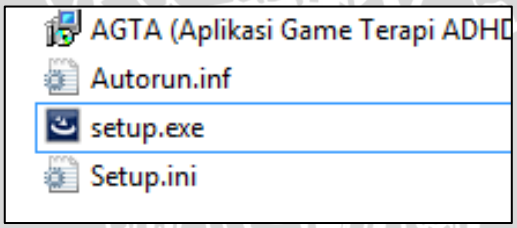
### L.1.1 Minimal Kelengkapan Sistem

- Microsoft Kinect
- PC/Laptop dengan kelengkapan :
  - OS : Windows 8 / Windows 7
  - Processor : Dual Core 2.0 GHz
  - Memory : 1 GB untuk lainnya
  - Hard Drive : 100 MB space kosong
  - DirectX 9.0c

### L.1.2 Installasi AGTA

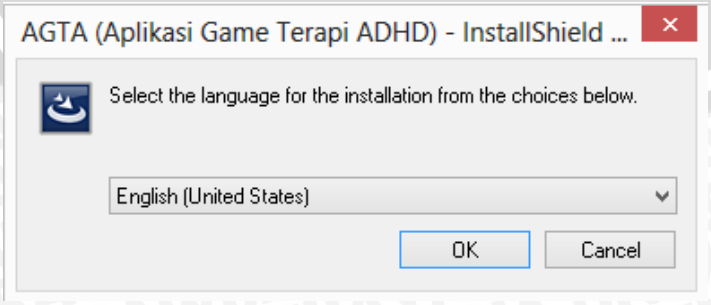
Langkah-langkah Installasi :

1. Siapkan CD *Installer* AGTA
2. Jalankan 'setup.exe' untuk memulai installasi denngan mengklik dua kali *file* setup.exe seperti pada gambar L.1



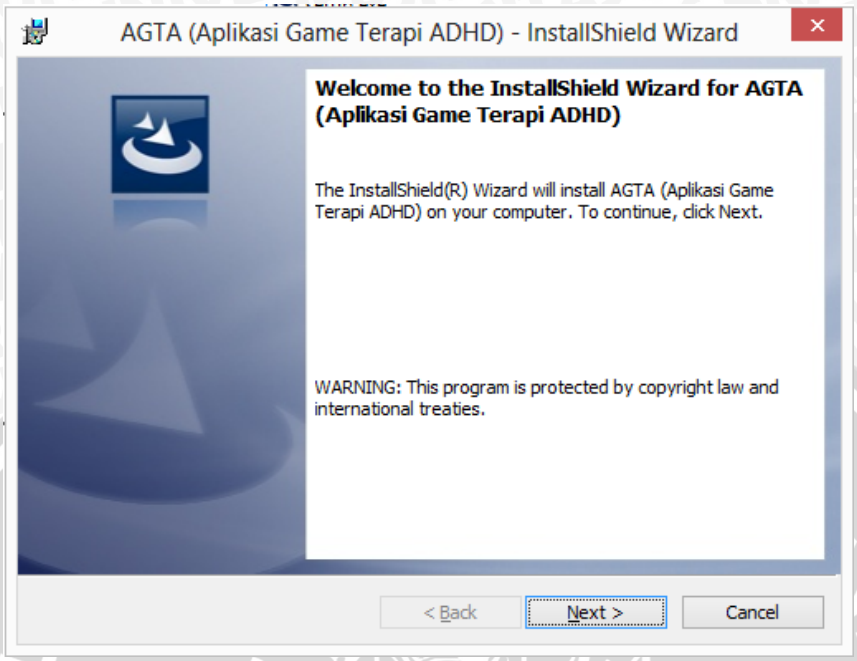

**Gambar L. 1 Tampilan lokasi *file* setup.exe pada *list* isi CD Instalasi**

3. Pilih bahasa yang digunakan dalam proses installasi seperti pada gambar L.2



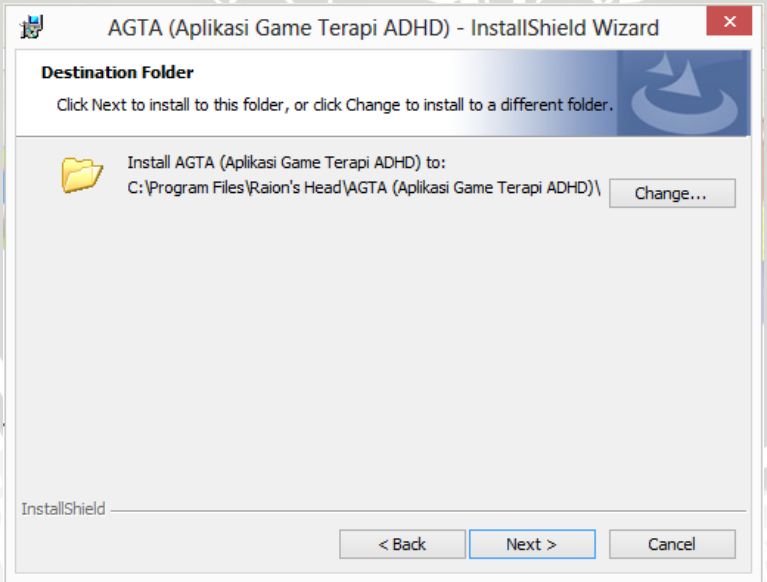

**Gambar L. 2 Tampilan *window* pilih bahasa**

4. Klik tombol '*Next'* pada *welcome screen* seperti pada gambar L.3.



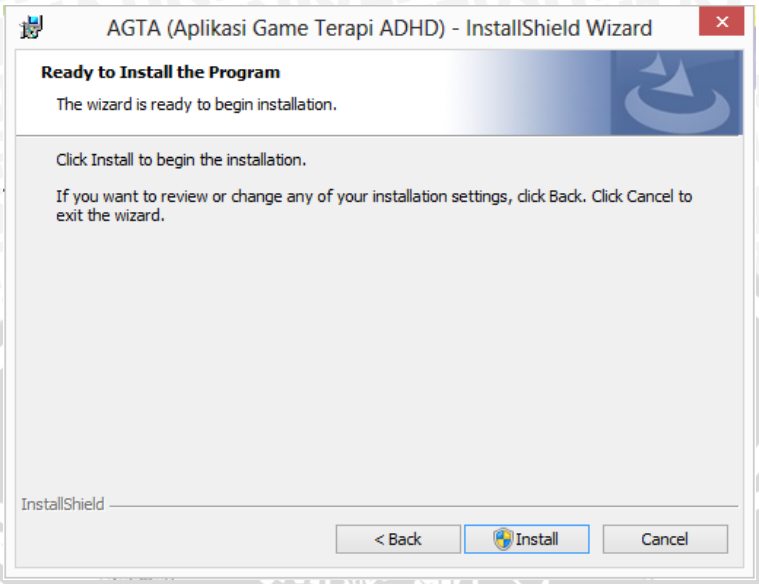

**Gambar L. 3 Tampilan *welcome screen* instalasi *game* AGTA**

5. Pilih *folder* tempat program akan diinstall, kemudian klik '*Next'* seperti pada gambar L.4.



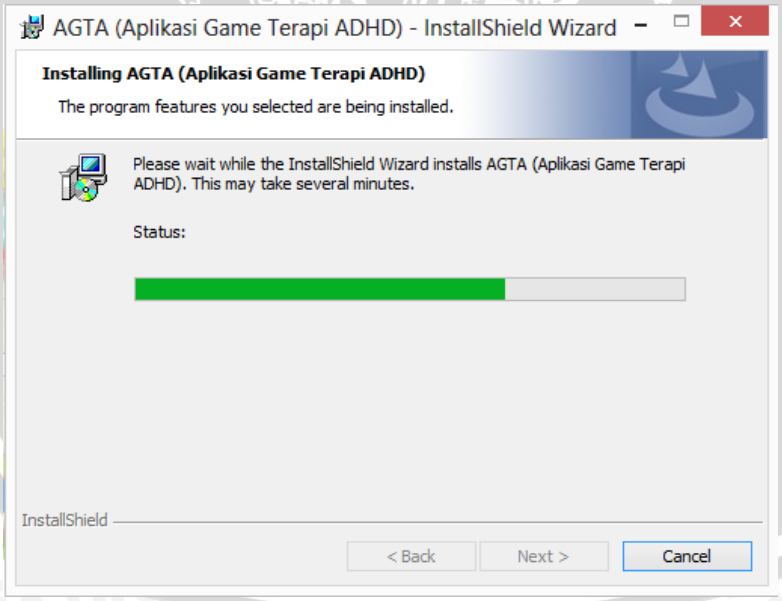

**Gambar L. 4 Tampilan *window* pemilihan *folder* instalasi**

6. Mulai proses instalasi dengan melakukan klik tombol '*Install'* seperti pada gambar L.5




**Gambar L. 5 Tampilan *window* instalasi *game* AGTA**

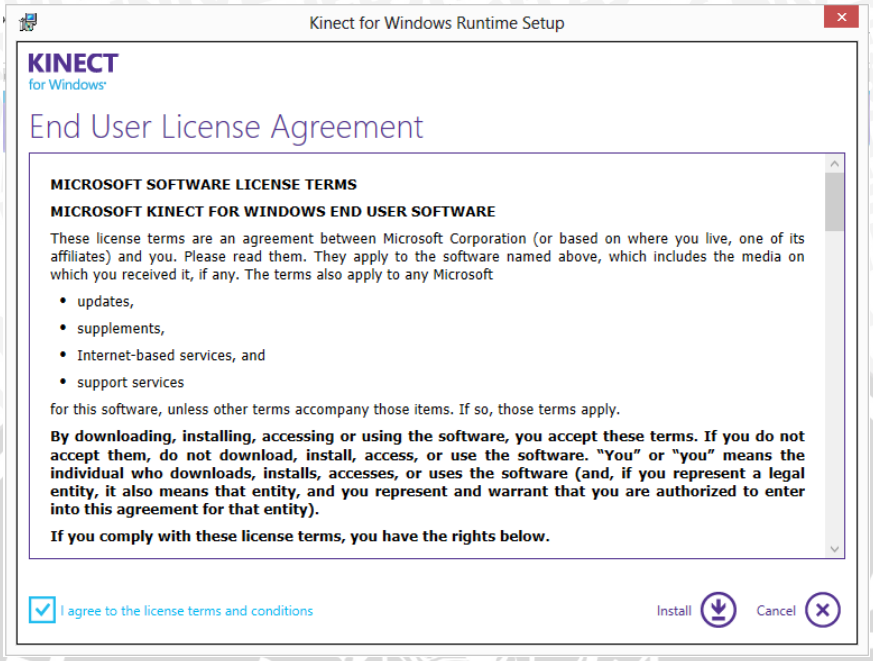
7. Proses installasi akan segera dimulai. Saat proses instalasi akan ditampilkan *window* seperti gambar L.6.




**Gambar L. 6 Tampilan *window* proses instalasi**

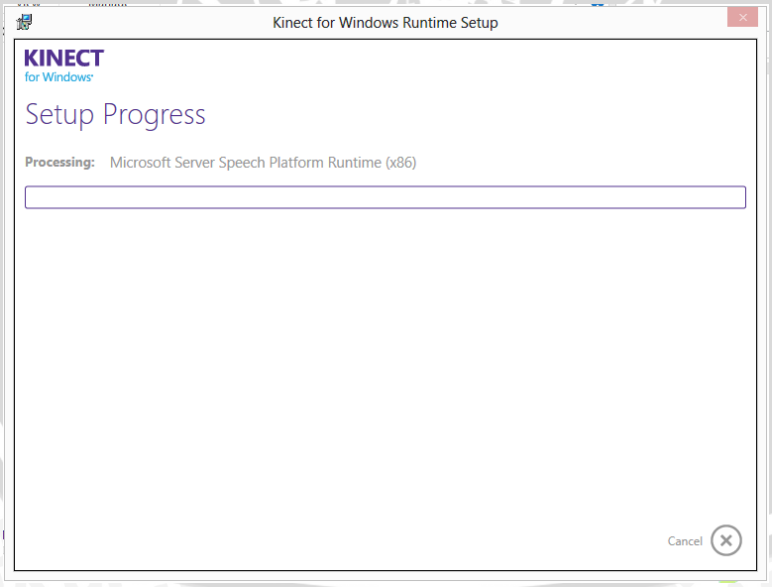
8. Pada pertengahan proses instalasi akan muncul dialog baru untuk menginstal Kinect Runtime. Jangan lewati tahap ini, karena aplikasi ini membutuhkan Kinect Runtime untuk dapat berjalan.

9. Centang bagian *user agreement*, lalu klik *install* seperti pada gambar L.7.



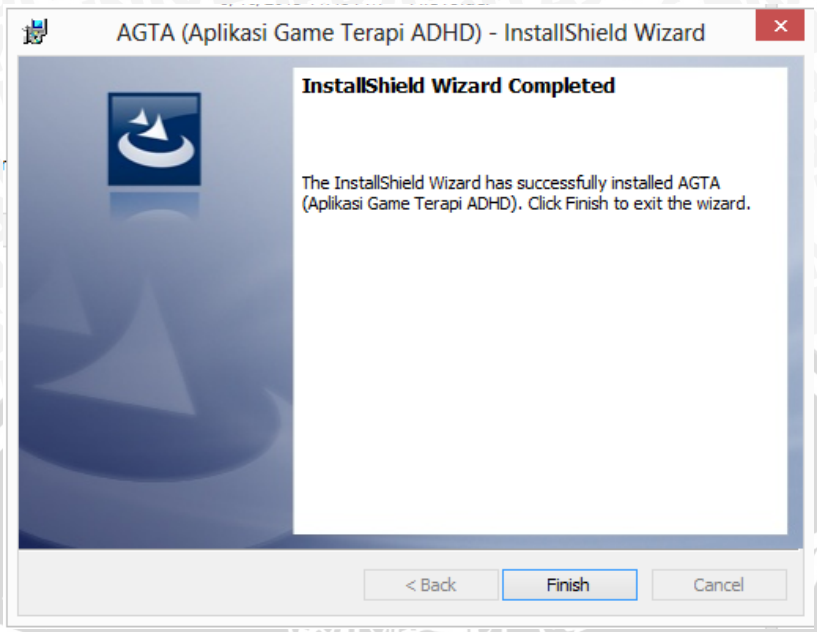

**Gambar L. 7 Tampilan *window user agreement* instalasi kinect**

10. Tunggu hingga proses instalasi selesai. Proses instalasi ditunjukkan pada gambar L.8



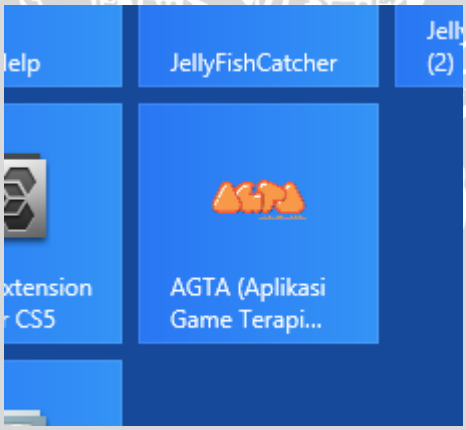

**Gambar L. 8 Tampilan proses instalasi Kinect**

11. Klik '*Finish'* saat proses instalasi telah selesai seperti pada gambar L.9.




**Gambar L. 9 Tampilan saat proses instalasi selesai**

12. *Shortcut* aplikasi akan muncul pada *desktop* dan *start menu* seperti pada gambar L.10, permainan siap dijalankan




**Gambar L. 10 *Shortcut game* AGTA pada *start menu***

### *L.1.3 Basic Tutorial*

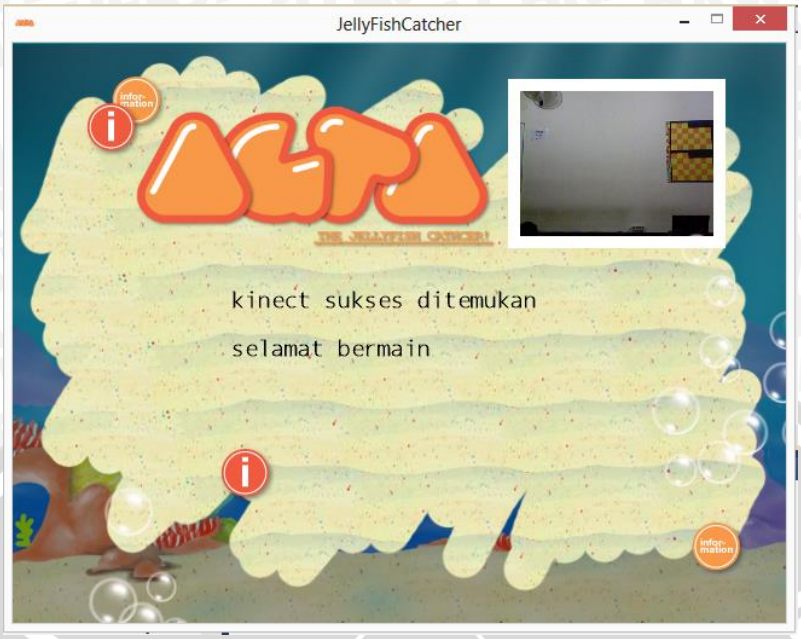
Lakukan beberapa persiapan sebagai berikut sebelum memulai aplikasi :

1. Persiapkan PC/Laptop yang telah terinstall AGTA
2. Persiapkan perangkat Kinect, dan hubungkan pada PC/Laptop anda
3. Jalankan aplikasi AGTA melalui *shortcut* atau *start menu*
4. Tunggu saat pengecekan Kinect, jika Kinect belum terpasang dengan benar maka akan muncul tampilan seperti pada gambar L.11 :


**Gambar L. 11 Tampilan Aplikasi saat Kinect belum Terpasang**

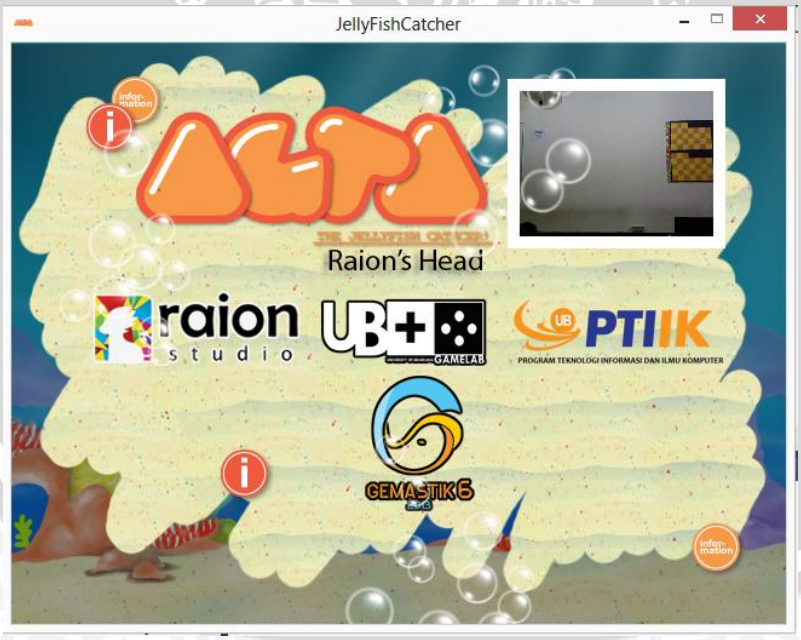
Maka coba cek dan pastikan kembali bahwa Kinect telah terhubung dengan benar ke PC/Laptop, kemudian *restart* kembali aplikasi AGTA

5. Jika Kinect sudah terpasang dengan benar maka akan muncul tampilan seperti pada gambar L.12 :




**Gambar L. 12 Tampilan Aplikasi saat Kinect Terkoneksi**

6. Tunggu beberapa saat maka *splash screen* akan muncul seperti pada gambar L.13, kemudian diikuti dengan tampilan menu utama seperti pada gambar L.14




**Gambar L. 13 Tampilan *Splash Screen* AGTA**

### L.1.4 Menu Utama


**Gambar L. 14 Tampilan Menu Utama**

Terdapat dua ikon menu permainan terapi terdapat *icon* help pada gambar L.17 dan information pada gambar L.18 :

- *Catch the Jellyfish*

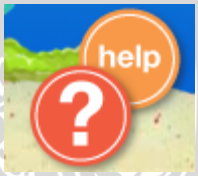
  Ikon *Catch the Jellyfish* ditunjukkan pada gambar L.15


**Gambar L. 15 *Icon Gameplay CatchTheJellyfish***

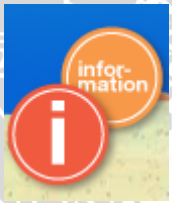
- *Falling Party*

  Ikon *game Falling Party* ditunjukkan pada gambar L.16


**Gambar L. 16 *Icon Gameplay Falling Party***




**Gambar L. 17 *Icon Help***




**Gambar L. 18 *Icon Information***

Arahkan kursor menggunakan tangan kanan untuk memilih permainan terapi yang akan digunakan, kemudian angkat tangan kiri hingga diatas kepala untuk memulai permainan.

### *L.1.5 Catch the Jellyfish*


**Gambar L. 19 Tampilan *Gameplay Catch The Jellyfish***

Dalam permainan ini pengguna/penderita akan belajar untuk berkonsentrasi dengan objek yang bergerak dan belajar mengkoordinasikan tangan kanannya. Tampilan permainan *Catch the Jellyfish* seperti pada gambar L.19

**Tujuan**

Tujuan dari permainan ini adalah untuk meningkatkan tingkat kefokusan penderita pada objek yang bergerak.

**Komponen Permainan**

- *Time* : Hitungan waktu lama permainan akan berlangsung. Tampilan HUD *Time* seperti pada gambar L.20


**Gambar L. 20 Tampilan HUD Waktu Permainan**

- *Score* : Jumlah *score* dari objek ubur-ubur yang berhasil didapatkan. Tampilan HUD *Score* seperti pada gambar L.21


**Gambar L. 21 Tampilan HUD *Score***

- Tombol *Pause* : Untuk berhenti sejenak pada permainan. Tampilan tombol *pause* seperti pada gambar L.22

**Gambar L. 22 Tampilan Tombol *Pause***

**Cara Bermain**

1. Pastikan tubuh anda berada pada tepat didepan perangkat Kinect sehingga dapat terlihat pada layar permainan
2. Jaring ubur-ubur akan muncul di layar berada pada tangan kanan anda
3. Arahkan jaring kearah ubur-ubur untuk menangkapnya
4. Score akan bertambah setiap kali ubur-ubur ditangkap
5. Untuk melakukan "*Pause*", arahkan jaring ke tombol *pause* dan tahan, maka tampilan *pause* akan muncul seperti pada gambar L.23:


**Gambar L. 23 Tampilan *Pause Menu***

6. Arahkan kursor pada tombol pilihan dan tahan :
   a. *Resume* : untuk kembali melanjutkan permainan
   b. *Restart* : mengulang permainan mulai awal
   c. *Exit* : kembali ke menu utama

### *L.1.6 Falling Party*


**Gambar L. 24 Tampilan *Gameplay Falling Party***

Dalam permainan ini pengguna/penderita akan belajar untuk mengkoordinasikan kedua tangannya secara bersamaan baik kiri maupun kanan. Tampilan permainan *Falling Party* seperti pada gambar L.24

**Tujuan**

Tujuan dari permainan ini adalah untuk meningkatkan tingkat keseimbangan dan koordinasi anggota tubuh pengguna/penderita.

**Komponen Permainan**

- *Time* : Hitungan waktu lama permainan akan berlangsung. Tampilan HUD *Time* seperti pada gambar L.25


**Gambar L. 25 Tampilan HUD Waktu Permainan**

- *Score* : Jumlah *score* dari objek ubur-ubur yang berhasil didapatkan. Tampilan HUD *Score* seperti pada gambar L.26

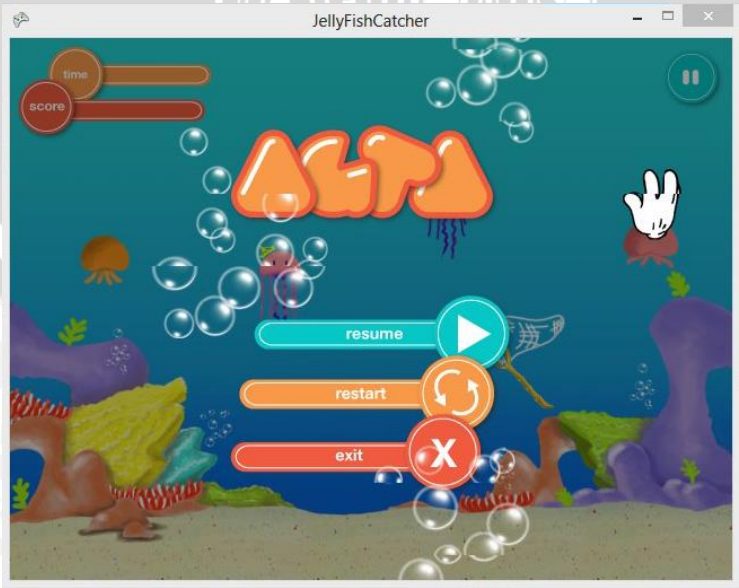

**Gambar L. 26 Tampilan HUD *Score***

- Tombol *Pause* : Untuk berhenti sejenak pada permainan. Tampilan tombol *pause* seperti pada gambar L.27

**Gambar L. 27 Tampilan Tombol *Pause***

**Cara Bermain**

1. Pastikan tubuh anda berada pada tepat didepan perangkat Kinect sehingga dapat terlihat pada layar permainan
2. Wadah untuk menangkap hewan laut yang muncul akan ada pada tangan kiri dan tangan kanan pengguna/penderita
3. Arahkan tangan kekiri dan kekanan untuk menangkap objek hewan laut yang muncul
4. Score akan bertambah setiap objek yang jatuh berhasil ditangkap
5. Untuk melakukan "*Pause*", wadah yang berada pada tangan kanan ke tombol *pause* dan tahan, maka tampilan *pause* akan muncul seperti pada gambar L.28 :



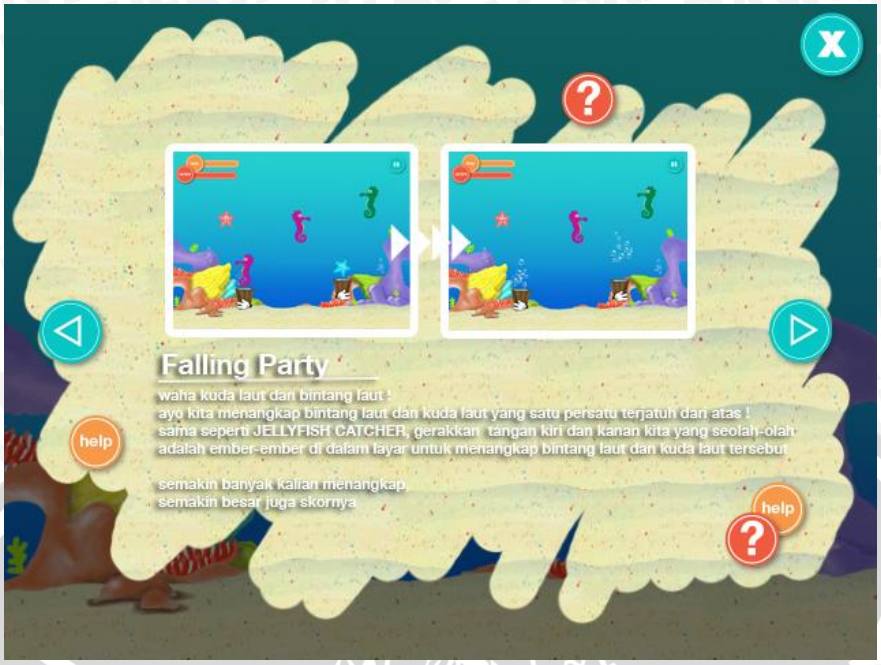

**Gambar L. 28 Tampilan *Pause Menu***

6. Arahkan kursor pada tombol pilihan dan tahan :
   a. *Resume* : untuk kembali melanjutkan permainan
   b. *Restart* : mengulang permainan mulai awal
   c. *Exit* : kembali ke menu utama

### *L.1.7 Help Screen*


**Gambar L. 29 Tampilan Halaman pertama *Help Screen***


**Gambar L. 30 Tampilan Halaman kedua *Help Screen***




**Gambar L. 31 Tampilan Halaman ketiga *Help Screen***

Pada halaman ini pengguna/penderita dapat melihat bagaimana cara memainkan *game* AGTA.

**Tujuan**

Tujuan dari *help screen* ini adalah untuk mengetahui cara menggunakan *game* AGTA.

**Komponen Halaman**

- Tombol *Next* : Tombol untuk pindah ke halaman *help* selanjutnya

**Gambar L. 32 Tampilan tombol *next***

- Tombol *Previous* : Tombol untuk pindah ke halaman *help* sebelumnya

**Gambar L. 33 Tampilan tombol *previous***

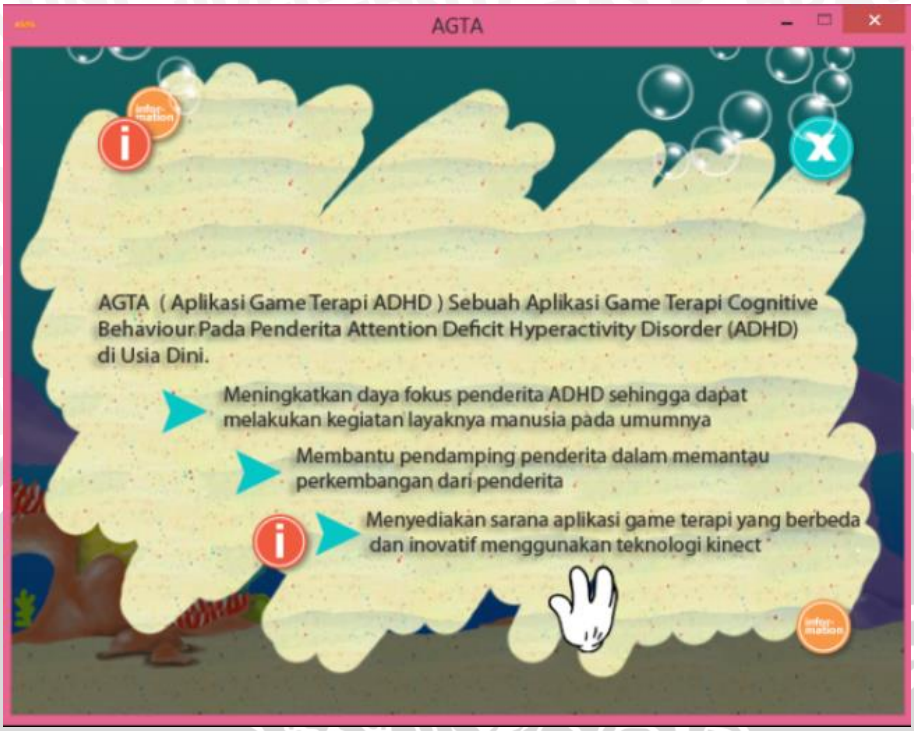
- Tombol *Exit* : Tombol untuk keluar dari halaman *help* dan menuju ke menu utama

**Gambar L. 34 Tampilan tombol *exit***

**Cara Bermain**

Arahkan kursor menggunakan tangan kanan untuk memilih tombol yang akan digunakan, kemudian angkat tangan kiri hingga diatas kepala untuk memilih.

### *L.1.8 Information Screen*




**Gambar L. 35 Tampilan Halaman *Information***

Pada halaman ini pengguna/penderita dapat melihat informasi dan manfaat *game* AGTA.

**Tujuan**

Tujuan dari *information screen* ini adalah untuk mengetahui informasi dan manfaat *game* AGTA.

**Komponen Halaman**

- Tombol *Exit* : Tombol untuk keluar dari halaman *help* dan menuju ke menu utama

**Gambar L. 36 Tampilan tombol *exit***

**Cara Bermain**

Arahkan kursor menggunakan tangan kanan untuk memilih tombol *exit*, kemudian angkat tangan kiri hingga diatas kepala untuk memilih.