RANCANG BANGUN GAME SLASH WORD MENGGUNAKAN KINECT SEBAGAI SENSOR KENDALI GAME DENGAN GERAKAN TANGAN

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
MUHAMMAD AMINUL AKBAR
0810680012

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS BRAWIJAYA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER
MALANG
2013

LEMBAR PERSETUJUAN

RANCANG BANGUN GAME SLASH WORD MENGGUNAKAN KINECT SEBAGAI SENSOR KENDALI GAME DENGAN GERAKAN TANGAN

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
MUHAMMAD AMINUL AKBAR
0810680012

Dosen Pembimbing I	Dosen Pembimbing II
Adharul Muttaqin, ST., MT.	**Eriq M. Adams J., ST., Mkom.**
NIP. 19760121 200501 1 001	**NIK. 850410 06 1 1 0027**

LEMBAR PENGESAHAN

RANCANG BANGUN GAME SLASH WORD MENGGUNAKAN KINECT SEBAGAI SENSOR KENDALI GAME DENGAN GERAKAN TANGAN

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh :

MUHAMMAD AMINUL AKBAR
NIM. 0810680012

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 25 Maret 2013

Penguji I	Penguji II
Denny Sagita R, S.Kom., M.Kom.	**Issa Arwani, S.Kom., M.Sc.**
NIP. 851124 06 1 1 0250	**NIP. 830922 06 1 1 0074**

Penguji III

Aryo Pinandito, S.T., M.MT

Mengetahui,
Ketua Program Studi Teknik Informatika

Drs. Marji, MT.
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 di dalam 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, 5 April 2013

Mahasiswa,

Muhammad Aminul Akbar

0810680012

KATA PENGANTAR

Dengan menyebut nama Allah Yang Maha Pengasih dan Penyayang. Puji syukur penulis panjatkan kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan laporan tugas akhir yang berjudul "Rancang Bangun Game Slash Word Menggunakan Kinect Sebagai Sensor Kendali Game Dengan Gerakan Tangan". Shalawat dan salam atas junjungan besar kita Nabi Muhammad S.A.W. beserta keluarga dan para sahabat sekalian. Laporan tugas akhir ini disusun untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer di Program Studi Informatika / Ilmu Komputer Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Malang.

Penyusunan laporan tugas akhir ini dapat terlaksana dengan baik karena adanya bantuan secara langsung maupun tidak langsung dari pihak tertentu, diantaranya :

1. Bapak **Adharul Muttaqin, ST., MT.** selaku dosen pembimbing I yang telah memberikan ilmu dan saran untuk laporan tugas akhir ini.
2. Bapak **Eriq M. Adams J., ST., Mkom.** selaku dosen pembimbing II yang juga memberikan ilmu dan saran untuk laporan tugas akhir ini.
3. **Orang Tua,** yang telah memberikan dukungan moral dan material.
4. Teman-teman yang telah membantu memberi kritik dan saran atas proposal ini.

Penulis menyadari bahwa laporan tugas akhir ini masih terdapat banyak kekurangan, oleh karena itu kritik dan saran yang bersifat membangun sangat diharapkan untuk menyempurnakan laporan tugas akhir ini. Saya berharap laporan tugas akhir ini dapat bermanfaat khususnya bagi diri sendiri dan bagi semua pihak.

Malang, 31 Maret 2013

Penulis

ABSTRAK

Muhammad Aminul Akbar. 2013. : Rancang Bangun Game Slash Word Menggunakan Kinect Sebagai Sensor Kendali Game Dengan Gerakan Tangan. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing: Adharul Muttaqin, ST., MT., dan Eriq M. Adams J., ST., Mkom.

Bahasa Inggris merupakan bahasa internasional yang harus dikuasai agar bisa berkomunikasi dan bersaing di era globalisasi. Kosakata bahasa sangat penting sebagai fondasi awal untuk belajar bahasa, termasuk bahasa Inggris. *Game* yang mendorong pemain untuk bergerak aktif dapat dimanfaatkan sebagai media pembelajaran yang menarik dalam mempelajari kosakata bahasa Inggris. Berdasarkan pada permasalahan tersebut, penulis membuat *game* Slash Word yang berisi pembelajaran kosakata bahasa Inggris. *Game* Slash Word menerapkan teknik interaksi alami dengan memproses setiap gerakan tangan pemain memanfaatkan sensor Kinect.

Game dirancang berdasarkan tahapan *game concept design* dan *technical design* yang kemudian diimplementasikan menggunakan Jmonkey *engine* 3. Penggabungan sensor kinect dengan Jmonkey *engine* 3 menggunakan *library* Kinect milik Glauco Márdano. Pengujian *white box* dari *game* Slash Word menghasilkan kesimpulan bahwa logika program telah berfungsi secara benar. Pengujian *black box* dari *game* Slash Word menghasilkan kesimpulan bahwa implementasi dan fungsionalitas *game* Slash Word telah memenuhi kebutuhan yang telah dijabarkan dalam daftar kebutuhan. Pengujian kinerja *game* Slash Word menghasilkan kesimpulan bahwa *game* Slash Word dapat berjalan dengan baik pada komputer yang dapat menghasilkan *frame per second* lebih besar atau sama dengan 30 FPS ketika memainkan *game* Slash Word.

Kata Kunci : *Game*, Kinect, Jmonkey, Kosakata.

ABSTRACT

Muhammad Aminul Akbar. 2013. : *Development of Slash Word Game Using Kinect Censor as Game Controller with Hand Movement. Undergraduate Thesis of Informatic Engineering Study Program, Information Technology and Computer Science Program, Brawijaya University, Malang. Advisor* : Adharul Muttaqin, ST., MT., *and* Eriq M. Adams J., ST., Mkom.

English is an international language that must be mastered in order to communicate and compete in the globalization era. Vocabulary is very important as the foundation for learning languages such as English. Games which encourage players for actively moving can be used as an interesting media for learning English vocabulary. Based on these problems, author has developed Slash Word game containing english vocabulary learning. Slash Word Game applied the natural interaction techniques to process every movements of the player's hand using Kinect sensor.

Game has been designed based on game design concept stage and technical design phase. It was implemented using Jmonkey engine 3. Implementation of Kinect sensor used Kinect library which made by Glauco Márdano. White box testing of Slash Word game shown that the program logic has been valid. Black box testing of Slash Word game proved that the implementation and functionality of Slash Word game has been appropriate with the requirement list. The result of performance testing shown that Slash Word game can be executed well on computers which be able to produce frames per second larger than or equal to 30 fps when playing it.

Keyword : Games, Kinect,Jmonkey, *Vocabulary.*

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Bahasa Inggris merupakan bahasa internasional yang harus dikuasai agar bisa berkomunikasi dan bersaing di era globalisasi seperti sekarang ini. Kosakata bahasa sangat penting sebagai fondasi awal untuk belajar bahasa, termasuk bahasa Inggris. Ada banyak cara yang dilakukan untuk meningkatkan kosakata bahasa Inggris, seperti, membaca literatur, percakapan, *listening*, membuat catatan saku, multimedia, dan permainan [HAP-11].

Dalam memberikan pelajaran bahasa Inggris, Beberapa guru lebih senang menggunakan metode ceramah. Guru lebih banyak memberikan teori daripada praktek. Dalam proses belajar mengajar siswa diminta menghafalkan beberapa kosakata bahasa Inggris dari kamus yang tebal ataupun dari buku materi yang ada. Metode tersebut membuat siswa mengalami kesulitan dalam menghafalkan beberapa kosakata bahasa Inggris. Metode pembelajaran seperti ini menjadi kurang efektif karena membuat siswa kurang tertarik dan malas untuk belajar [NOV-12].

Kegiatan belajar dan bermain sering dipersepsikan sebagai kegiatan yang kontradiktif. Belajar dianggap sebagai kegiatan yang serius dan membosankan. Sebaliknya, bermain dianggap sebagai aktivitas yang menyenangkan, tetapi membuang waktu dan tenaga. Namun, kedua aktivitas tersebut dapat dilakukan bersama – sama. Kegiatan bermain yang menyenangkan dapat dimanfaatkan sebagai media belajar dengan memberikan materi pembelajaran didalamnya. Kegiatan bermain dan belajar yang disajikan dengan multimedia serta berisi materi yang membuat pemain tertantang dan ingin bergerak akan menumbuhkan minat belajar [SUP-08].

Video game adalah salah satu media permainan. Perkembangan generasi terbaru dari *video game* telah menunjukan arah pengembangan *game* untuk memperoleh pengalaman realistis melalui teknik interaksi alami. Teknik interaksi alami menjadi teknik yang meniru interaksi dunia nyata menggunakan gerakan tubuh dan menggunakan tindakan yang mirip dengan yang digunakan untuk

melakukan kegiatan di dalam dunia nyata. Menurut McMahan dkk, dalam evaluasi teknik interaksi alami dalam *video game* menunjukan penggunaan teknik interaksi alami dianggap lebih menyenangkan dibandingkan dengan *non-natural interaction* [MCM-10:14]. Penggunaan *motion-sensing game controller* sebagai teknik interaksi alami pada konsol Wii membuat Nintendo sukses mengirim 75 juta unit Wii di seluruh dunia lebih besar dibandingkan dengan PS3 dan XBOX 360 yang hanya dijual kurang dari 50 juta unit pada Desember 2010 [SUN-11]. Hal ini menunjukan ketertarikan lebih masyarakat terhadap *game* yang lebih interaktif dan inovatif.

Perkembangan penggunaan teknik interaksi alami pada *video game* berlanjut dengan munculnya sistem Kinect pada November 2010 yang merupakan tanggapan Microsoft terhadap sistem dari Nintendo Wii [SUN-11:93]. Yang menarik dari device ini adalah Kinect memiliki konsep *controller-free gaming* [MCM-10:14]. Disebut *controller-free gaming* karena Kinect hanya berupa device kamera tambahan untuk konsol Xbox 360, dimana kendali *game* dapat digerakan secara penuh oleh gerakan tubuh pemain tanpa menggunakan remote seperti pada Nintendo Wii.

Berdasarkan pada permasalahan diatas maka dibuatlah *game* Slash Word yang mempunyai materi pendidikan yaitu pembelajaran kosa kata bahasa Inggris dan terdapat unsur gerakan didalamnya. *Game* Slash Word menerapkan teknik interaksi alami dengan memanfaatkan kamera sensor Kinect. Pembuatan *game* ini menggunakan *game engine* 3D yang merupakan perangkat lunak yang dirancang untuk menciptakan dan mengembangkan *video game* 3D. Diharapkan *game* Slash Word dapat menjadi solusi *game* edukasi bahasa Inggris yang menarik dan menyenangkan untuk semua umur.

1.2. Rumusan Masalah

Berdasarkan pada permasalahan yang telah dijelaskan pada bagian latar belakang, maka rumusan masalah dapat disusun sebagai berikut :

1. Bagaimana *gameplay* dari *game* Slash Word yang menerapkan teknik interaksi alami menggunakan sensor Kinect untuk pembelajaran kosakata bahasa Inggris ?

2. Bagaimana merancang dan membangun *game* Slash Word menggunakan *game engine* 3D ?
3. Bagaimana pengujian *game* Slash Word ?

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. Implementasi *game* Slash Word menggunakan *game engine* 3D dengan bahasa pemrograman berorientasi objek.
3. Pengujian *game* meliputi :
    a. Pengujian unit menggunakan metode *White Box.*
    b. Pengujian integrasi menggunakan metode *White Box.*
    c. Pengujian validasi menggunakan metode *Black Box*
    d. Pengujian kinerja dengan melihat FPS (*Frame Per Second*) yang dihasilkan *game*.

1.4. Tujuan

Tujuan dari penelitian ini adalah :

1. Membuat *gameplay* dari *game* Slash Word yang menggunakan teknik interaksi alami menggunakan sensor Kinect untuk pembelajaran kosakata bahasa Inggris.
2. Merancang dan membangun *game* Slash Word menggunakan *game engine* 3D.
3. Menguji *game* Slash Word.

1.5. Manfaat

Penelitian ini diharapkan dapat bermanfaat untuk berbagai pihak. Manfaat dari penelitian ini adalah sebagai berikut:

1. Bagi penulis:
    a. Mendapatkan pengetahuan tentang pembuatan *game* dengan menggunakan sensor Kinect.

2. Bagi pengguna
   a. Membantu para pengembang *game* dalam mengembangkan *game* yang lebih interaktif dengan menerapkan teknik interaksi alami menggunakan sensor Kinect.
   b. Materi *game* Slash Word digunakan sebagai media pembelajaran Bahasa Inggris yang menarik.

1.6. Sistematika Pembahasan

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*, *game engine*, sensor Kinect, OpenNI, NITE dan penjelasan mengenai proses - proses pembuatan *game*.

Bab III Metodologi Penelitian

Bab ini berisi tentang gambaran umum langkah – langkah dalam pembuatan *game* Slash Word.

Bab IV Perancangan

Membahas tentang *game concept design* dan *technical design*.

Bab V Implementasi

Membahas tentang implementasi *game*.

Bab VI Pengujian dan Analisis

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

Bab VII 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* Slash Word diantaranya tentang *game*, *game engine*, sensor Kinect, OpenNI, NITE *middleware* 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-10:2]. *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* dapat memiliki lebih dari satu genre yang disebut dengan *hybrid* [PED-08:33]. *Game* Slash Word adalah *game* yang memiliki genre gabungan dari *educational* dan *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-08:34].

Educational game adalah tipe *game* yang menekankan pada pembelajaran. *Game* ini dirancang untuk mengajarkan atau memperkuat konsep belajar. *Educational games* yang paling dasar adalah permainan seperti pengisian teks kosong, pilihan ganda atau esai [PED-08:42]. *Educational game* dapat dirubah sedemikian rupa untuk membuat *game* tersebut lebih menarik.

Dalam membuat sebuah *game* kita dapat memanfaatkan *Game Engine.* Pembuatan *game* Slash Word menggunakan *Game Engine* 3D dengan bahasa pemrograman berorientasi objek.

2.2. *Game Engine*

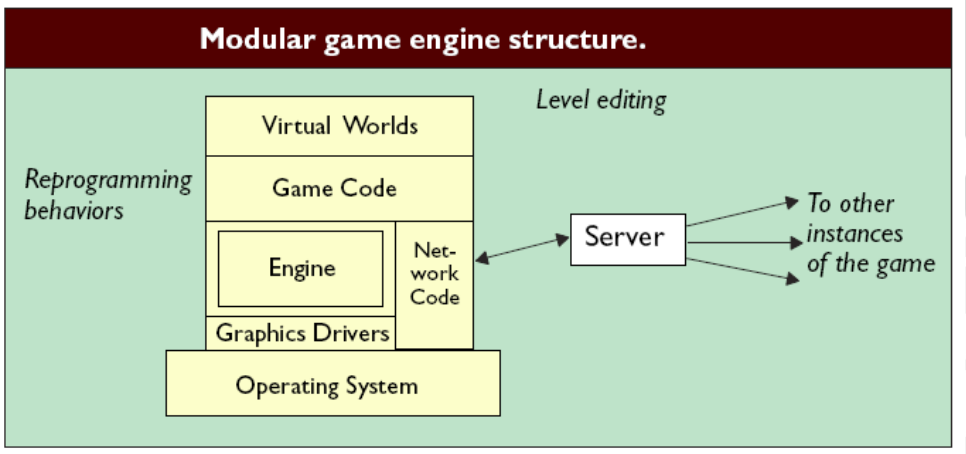
Game engine merujuk pada kumpulan modul kode simulasi yang tidak secara langsung menentukan perilaku permainan *(game logic)* atau lingkungan permainan *(level data). Game engine* mencakup modul untuk menangani input, output (3D Rendering, gambar 2D, suara) dan *generic physics* atau dinamika untuk dunia *game* [LEW-02:28]. Gambar 2.1 menunjukan struktur *game engine.*

Bagian paling atas yaitu *virtual worlds* atau skenario dimana pemain berinteraksi. Terdiri dari berbagai macam tampilan dan peraturan interaksi. Bahkan beberapa tipe dari simulasi fisika.

Game code menangani sebagian besar mekanik dasar *game* itu sendiri, seperti fisika sederhana, parameter di layar, jaringan dan animasi beberapa aksi. Perubahan pada *game code* memerlukan pengetahuan *scripting* permainan yang spesifik.

Rendering engine adalah bagian mahkota dari *game engine*. Terdiri dari kode yang rumit yang diperlukan untuk mengefisiensi identifikasi dan pembuatan lingkungan tampilan pemain dari model 3D yang kompleks.

Network code mendukung pembangunan protocol jaringan yang memungkinkan beberapa pengguna untuk berinteraksi dalam lingkungan virtual yang sama.



Gambar 2.1. Struktur *Game Engine.*

Sumber : [LEW-02:28]

Graphics Driver menerjemahkan permintaan dari *rendering engine* ke perpustakaan grafis, menggunakan API seperti DirectX, OpenGL dan API lainnya.

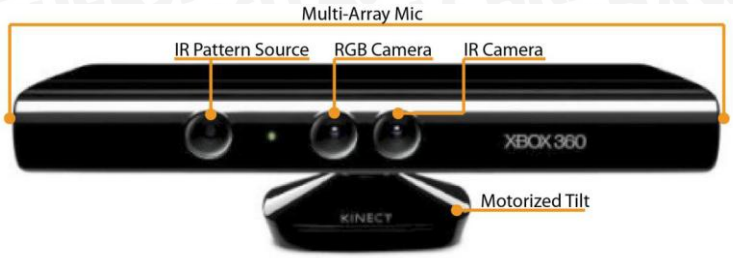
Yang terakhir adalah server menangani proses terpisah, biasanya pada mesin yang berbeda yang menangani informasi pada dunia virtual *game* tersebut. Server mengkomunikasikan informasi pada lingkungan yang digunakan bersama oleh *game client* seperti interaksi antar pemain seperti terjadinya kerusakan pada salah seorang pemain dan masalah sinkronisasi informasi antar pemain. Sebagai contoh perubahan suatu pemain maka perubahan tersebut juga harus ditampil sama pada pemain lain [LEW-02:29].

Dalam pembuatan *game* Slash Word memanfaatkan *game engine* 3D menggunakan bahasa pemrograman berorientasi objek. Dengan menggunakan *game engine* 3D dengan bahasa pemrograman berorientasi objek diharapkan *game* Slash Word dapat dibuat dengan efisien, optimal dan memenuhi *requirement* yang telah ditentukan. *Game* Slash Word menggunakan input dari *device* Kinect, sehingga membutuhkan *game engine* yang dapat menangani inputan dari *device* tersebut.

2.3. Sensor Kinect

Kinect adalah antarmuka permainan baru untuk konsol *game* Microsoft Xbox 360. Kinect secara resmi diluncurkan pada bulan Oktober 2010, adalah yang pertama untuk mencapai kontrol seluruh tubuh yang rumit dengan 3D *motion capture* . Antarmuka ini didasarkan pada teknologi yang dikembangkan oleh Rare dan PrimeSense untuk teknologi *game* dan rekontruksi gambar berbasis 3D. Kinect terdiri dari tiga perangkat yaitu kamera RGB, sensor kedalaman, dan mikrofon multi-array. Kamera ini dapat menghasilkan output video pada 30 Hz frame dengan 8-bit resolusi VGA (640 x 480 pixels) [YOO-11]. *Device* Kinect ditunjukan pada gambar 2.2.

Kinect digunakan sebagai antarmuka untuk kendali pemain pada *game* Slash Word yang dirancang hanya menggunakan gerakan tangan pemain sebagai kendali permainannya. Terdapat beberapa *library* untuk mengakses *device* Kinect diantaranya adalah *library* OpenNI.



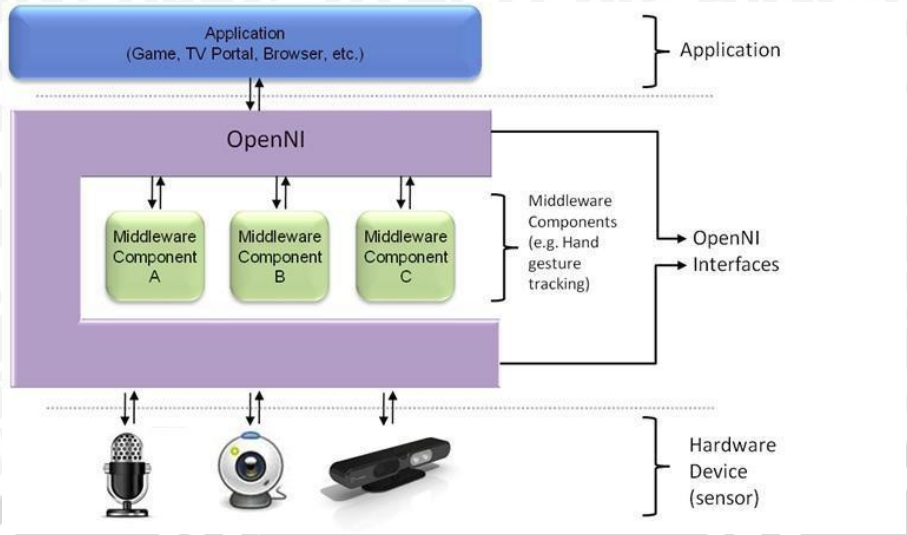
Gambar 2.2. Sensor Kinect.

Sumber : [BOU-11]

2.4. OpenNI dan NITE

OpenNI *(Open Natural Interaction)* adalah *library* yang digunakan untuk antarmuka kinect melalui driver PrimeSense. Warna dan kedalaman gambar dapat diakses melalui API OpenNI, dan dua gambar dapat secara otomatis ditransformasi dan dipetakan bersama. OpenNI juga menyediakan unit konversi dari proyektif ke koordinat dunia nyata dan sebaliknya [PIU-11]. Diagram OpenNI ditunjukan pada gambar 2.3.

NITE *(Natural Interaction Technology for End-user)* adalah *middleware* yang dikembangkan oleh Primesense yang memiliki paten di balik teknologi yang digunakan oleh Kinect. NITE memiliki algoritma untuk identifikasi pengguna, deteksi fitur, dan pengenalan gerakan, serta kerangka untuk pengelolaan penanda pengguna pada *scene* dan pengakuisisian serta pelepasan control pengguna [BOU-11]. Pada gambar 2.3 NITE adalah sebagai *middleware component*. OpenNI dikombinasikan dengan NITE memungkinkan pelacakan tubuh penuh dalam berbagai bahasa dan aplikasi [HUC-11].

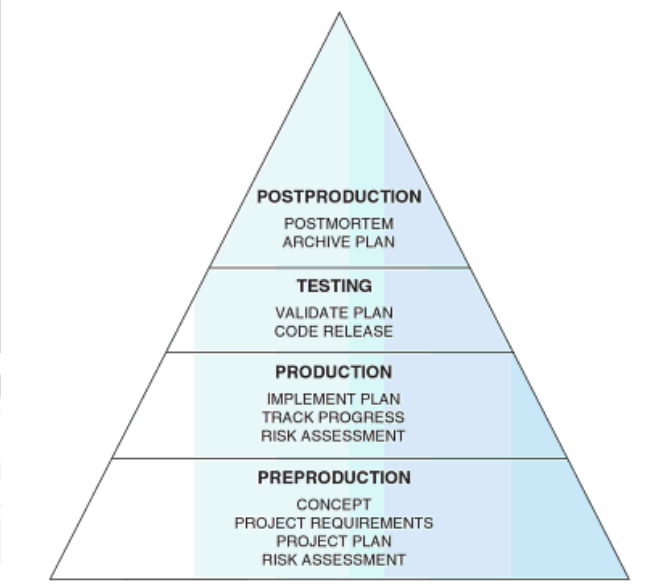


Gambar 2.3. Diagram Arsitektur OpenNI

Sumber : [HUC-11]

2.5. *Game Production*

Proses produksi dimulai dengan mendefinisikan game konsep 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 2.4 menunjukan gambaran ikhtisar dari *basic production cycle*.



Gambar 2.4. *Basic Production Cycle*

Sumber : [CHA-11]

2.5.1. *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* Slash Word, *preproduction phase* terdiri dari *game concept design* dan *technical design.*

2.5.1.1. *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-10:32]. *Brainstroming note* disusun dalam sebuah *one sheet document* agar lebih terstruktur sehingga pembaca awam mengerti ringkasan tentang *game* yang dibuat [ROG-10:60]. 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-10:62]. Langkah terkahir 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-10:75].

2.5.1.2. *Technical Design*

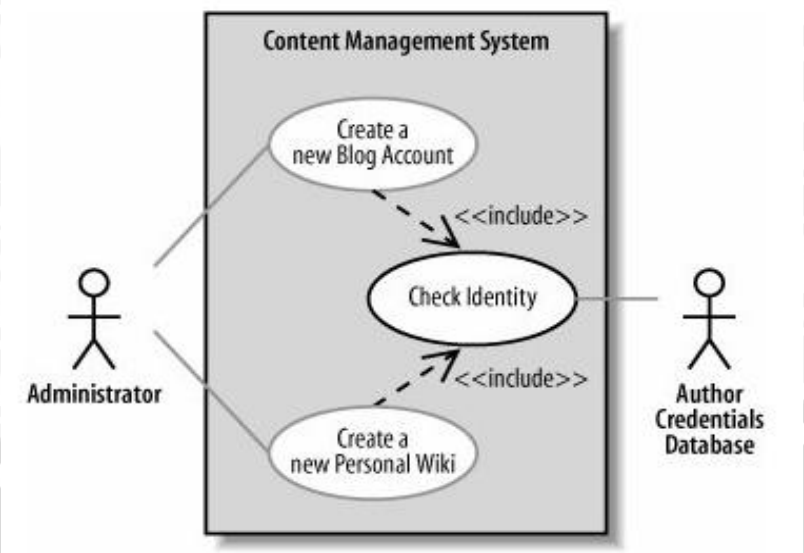
Pada tahap *technical design*, pemodelan sistem dilakukan dengan menggunakan UML (*Unified Markup Language*). UML adalah bahasa grafis untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. UML berorientasi objek, menerapkan banyak level abstraksi, tidak

bergantung proses pengembangan, tidak bergantung bahasa dan teknologi [HAR-04:259]. 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* Slash Word yaitu : (1) pembuatan diagram *use case* , (2) *pembuatan* rancangan diagram kelas, (3) pembuatan rancangan *activity diagram.*

2.5.1.2.1. *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-04:267] :

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.



Gambar 2.5. Diagram *Use Case*.

Sumber : [HAM-06]

4. Menyediakan kemampuan melacak kebutuhan fungsionalitas menjadi kelas – kelas dan operasi – operasi aktual di sistem.

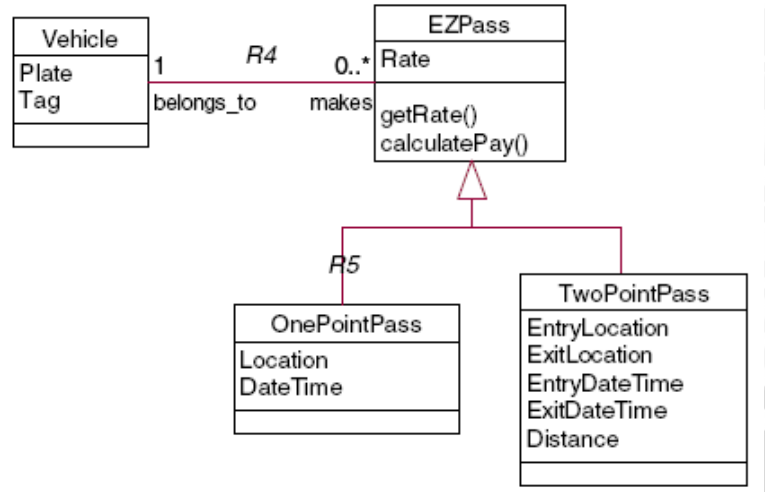
Elemen diagram *use case* adalah [HAR-04:268] :

1. Aktor merupakan pemakai sistem.
2. *Use case* adalah cara spesifik penggunaan sistem oleh aktor.
3. Hubungan ketergantungan, generalisasi dan asosiasi.

Contoh dari diagram *use case* ditunjukan pada gambar 2.5.

2.5.1.2.2. *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 sitem. Kelas di diagram kelas dapat secara langsung diimplementasikan ke dalam bahasa pemrograman berorientasi objek yang secara langsung mendukung bentukan kelas [HAR-04:277]. Contoh diagram kelas ditunjukan pada gambar 2.6.



Gambar 2.6. Diagram Kelas.

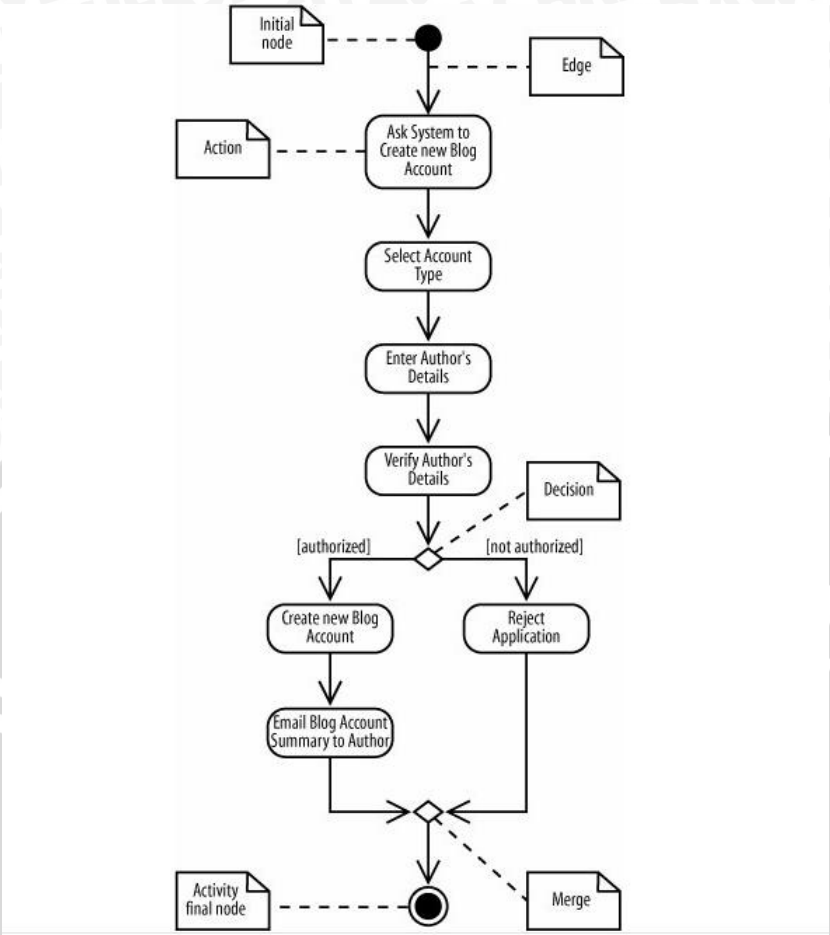
Sumber : [LIU-06:19]

Elemen – elemen esensi di diagram kelas adalah sebagai berikut :

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

2.5.1.2.3. *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* Slash Word, diagram aktivitas digunakan untuk menggambarkan aliran aktivitas atau proses dari setiap use case dalam *game* Slash Word. Gambar 2.7 menunjukan contoh dari diagram aktivitas proses pembuatan akun blog.



Gambar 2.7. Diagram Aktivitas Proses Pembuatan Akun Blog.

Sumber : [HAM-06]

2.5.2. *Production Phase*

Tahap ini adalah tahap yang dilakukan untuk merealisasikan rancangan dari *game*. Implementasi *game* Slash Word menggunakan bahasa pemrograman berorientasi objek (*Object Oriented Programming Language*).

2.5.3. *Testing*

Pengujian adalah proses pemerikasaan atau evaluasi sistem atau komponen sistem secara manual atau otomatis untuk memverifikasi apakah sistem memenuhi kebutuhan – kebutuhan yang dispesifikasikan atau mengidentifikasi perbedaan – perbedaan antara hasil yang diharapkan dengan hasil yang terjadi. Awalnya, pengujian diartikan sebagai aktivitas yang dapat atau hanya dilakukan setelah

pengkodean (Kode program selesai). Namun, pengujian seharusnya dilakukan dalam skala lebih luas. Pengujian dapat dilakukan begitu spesifikasi kebutuhan telah dapat didefinisikan [HAR-04:569].

Terdapat dua teknik pengujian berdasarkan ketersediaan logik sistem, yaitu *Black Box Testing* dan *White Box Testing* [HAR-04:577]. Pada penelitian ini menggunakan dua teknik pengujian tersebut.

2.5.3.1. *Black Box Testing*

Konsep kotak hitam digunakan untuk mempresentasikan sistem yang cara kerja di dalamnya tidak tersedia untuk diinspeksi. Di dalam kotak hitam, item – item yang diuji dianggap "gelap" karena logiknya tidak diketahui, yang diketahui hanya apa yang masuk dan apa yang keluar dari kotak hitam. Teknik pengujian *Black Box* dapat digunakan untuk pengujian berbasis skenario, dimana isi dalam sistem mungkin tidak tersedia untuk diinspeksi tapi masukan dan keluaran yang didefinisikan dengan *use case* dan informasi analisis yang lain [HAR-04:577].

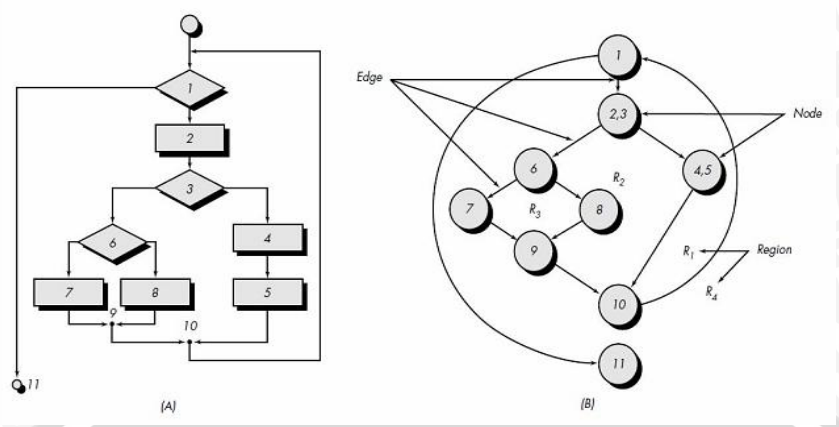
2.5.3.2. *White Box Testing*

Pengujian *White Box* mengasumsikan bahwa logik spesifik adalah penting dan harus diuji untuk menjamin sistem melakukan fungsi dengan benar. Pada pengujian *White Box*, kita mencari *bug* yang mempunyai peluang eksekusi yang rendah, atau yang diimplementasikan secara sembrono.

Pengujian kotak putih juga disebut pengujian struktur (*Struktural Testing*). Penurunan kasus – kasus pengujian berdasarkan struktur program. Teknik pengujian *White Box* antara lain [HAR-04:578] :

1. ***Basis Path Testing***

   Metode ini melakukan pencarian ukuran kompleksitas perancangan prosedur dan menggunakannya untuk mendefinisikan sekumpulan jalur eksekusi dasar. Kasus uji diturunkan dari himpunan basis yang dijamin mengeksekusi seluruh pernyataan di program setidaknya satu kali selama pengujian. Langkah – langkahnya adalah sebagai berikut :



Gambar 2.8. Transformasi *Flow Chart* ke *Flow Graph*.
Sumber : [PRE-01:447]

1) Gunakan rancangan atau kode program sebagai dasar, kemudian buat *flow graphnya*. Gambar 2.8 menunjukan tranformasi *flow chart* prosedur ke bentuk *flow graph.*
2) Tentukan kompleksitas siklomatik dari *flow graph* yang dibuat. *Cyclomatic complexity* adalah metriks perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program. Bila metriks ini digunakan dalam konteks metode pengujian *Basis Path,* maka nilai yang terhitung untuk *cyclomatic complexity* menentukan jumlah jalur independen (*independent path*) dalam *basis set* suatu program dan memberi batas atas bagi jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua *statement* telah dieksekusi sedikitnya satu kali. Jalur independen adalah jalur yang melalui program yang mengenalkan sedikitnya satu rangkaian *statement* proses baru atau suatu kondisi baru. Untuk menentukan *cyclomatic complexity* bisa dilakukan dengan beberapa cara, diantaranya [PRE-10:488]:
   a) Jumlah *region* pada *flow graph* sesuai dengan *cyclomatic complexity*.
   b) *Cyclomatic complexity* V(G), untuk grafik G adalah V(G) = E – N + 2, dimana E adalah jumlah *edge*, dan N adalah jumlah *node.*

c) V(G) = P + 1, dimana P adalah jumlah *predicate node* yaitu *node* yang merupakan kondisi (ada 2 atau lebih *edge* akan keluar *node* ini).

3) Tentukan himpunan basis dari jalur – jalur yang independen secara linear.

4) Persiapkan kasus pengujian yang akan memaksa eksekusi masing – masing jalur di himpunan basis.

Pada penelitian ini menggunakan teknik *Basis Path Testing*.

2. *Control Structure Testing*

Terdapat beragam *Control Structure Testing, yaitu* :

1) *Condition testing*
2) *Data flow testing*
3) *Loop testing*

2.5.3.3. *Metode Pengujian*

Pengujian di sistem berorientasi objek umumnya dilakukan secara *bottom-up* dalam empat level, yaitu [HAR-04:585]:

1. Pengujian level metode yang menguji metode individu di kelas.
2. Metode – metode dan atribut – atribut yang menyusun kelas. Pengujian level kelas (intra kelas) adalah pengujian terhadap interaksi – interaksi di antara komponen – komponen di suatu kelas individu.
3. Kelas – kelas yang bekerja sama menyusun tandan kelas (*class cluster*). Pengujian level tandan kelas menguji interaksi – interaksi di antara kelas – kelas.
4. Tandan – tandan kelas menyusun sistem. Pengujian level sistem berurusan dengan masukan dan keluaran yang tampak bagi pemakai sistem.

Perubahan strategi pengujian pada pendekatan berorientasi objek :

1. Pengujian unit, konsep mengenai unit diperluas di sistem berorientasi objek
2. Pengujian integrasi, integrasi berfokus pada kelas – kelas dan eksekusinya pada satu *thread* atau satu *use case*
3. Pengujian validasi, menggunakan pengujian kotak hitam tradisional.

2.5.3.4. *Frame Per Second (* FPS *)*

Menurut Claypol Mark dan Kajal (dalam Muller dkk, 2010:126) *frame per second* adalah metrik yang paling umum digunakan sebagai evaluasi kinerja dari *video games*. Sebuah aturan praktis bahwa sebagian besar *games* sangat menyenangkan untuk dimainkan dengan *frame rate* diatas 30 FPS. Namun *games* saat ini mencoba mencapai *frame rate* 60 FPS agar dapat melakukan sinkronisasi dengan layar yang ada saat ini. Berapa banyak *frame* yang dapat diproduksi dalam tiap detik adalah akibat langsung dari kinerja keseluruhan sistem dan pengaturan kualitas *game* [MUL-10:126].

Dalam *game panel*, sebuah *frame* sesuai dengan satu kali loop melewati proses *update-render-sleep* dalam prosedur run. Oleh karena itu jika diinginkan 100 FPS berarti bahwa setiap iterasi harus mengambil 1000 / 100 ==10 ms. Waktu iterasi ini disimpan dalam variable waktu dalam *game panel* [DAV-05:23]. *Pseudo code* dari prosedur run termasuk *timing code* dan *sleep calculation* ditunjukan pada tabel 2.1.

Tabel 2.1. *Pseudo Code* Prosedur Run.

NO	PSEUDOCODE
1	**METHODE** Run
2	
3	**DECLARATION :**
4	TYPE beforeTime IS long
5	timeDiff IS long
6	sleepTime IS long
7	
8	**DESCRIPTION :**
9	running ← true
10	**WHILE** (running)**DO**
11	**CALL** gameUpdate()
12	**CALL** gameRender()
13	**CALL** paintScreen()
14	
15	timeDiff ← **CALL** System.currentTimeMillis()- beforeTime
16	sleepTime ← period – timeDiff
17	
18	**IF** (sleepTime <= 0)
19	**THEN** sleepTime ← 5
20	**ENDIF**
21	
22	**TRY**
23	**CALL** Thread.sleep(sleepTime)
24	**ENDTRY**
25	
26	**CATCH** (InterruptedException ex)
27	**ENDCATCH**
28	
29	beforeTime ← **CALL** System.currentTimeMillis()
30	
31	**ENDWHILE**
32	
33	**CALL** System.exit(0)
34	
35	**END** Run

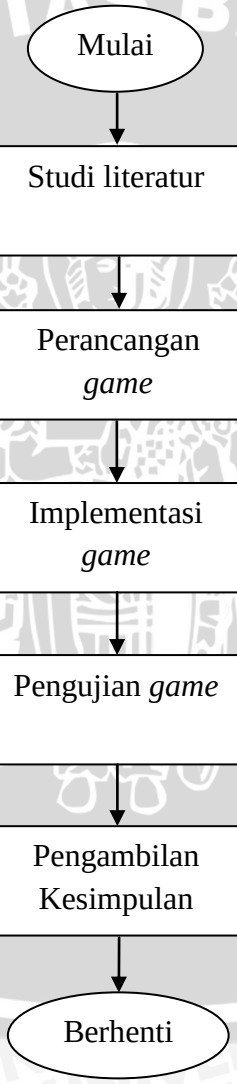
Sumber : [DAV-05:23]

2.5.4. *Postproduction Phase*

Pada proses *postproduction* yang dilakukan adalah membuat *archive* dari game Slash Word yang berisi dokumen konsep, aset dan kode dari game Slash Word. Tahap ini tidak dilaporkan dalam penulisan laporan skripsi ini.

BAB III
METODOLOGI PENELITIAN

Langkah – langkah yang dilakukan dalam penelitian ini yaitu : studi literatur, perancangan *game*, implementasi *game*, pengujian *game* dan pengambilan kesimpulan. Gambar 3.1 menunjukan diagram alir metode penelitian.



Gambar 3.1. Diagram Alir Metode Penelitian.

Penjelasan dari masing – masing proses pada diagram alir Metode Penelitian adalah sebagai berikut :

3.1. Studi Literatur

Studi literatur dilakukan untuk mempelajari teori yang digunakan dalam pembuatan *game* Slash Word. Teori – teori yang dipelajari yaitu tentang :

1. *Game Engine.*
2. *Natural Interaction* menggunakan device kinect.
3. Openni dan Nite API.
4. Proses produksi *game.*

3.2. Perancangan *Game*

Perancangan *game* merupakan bagian dari tahap *preproduction* yang terdiri dari dua tahap. Tahap pertama yaitu *game concept design* dan *technical design.*

3.2.1. *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-10:32]. *Brainstroming note* disusun dalam sebuah *one sheet document* agar lebih terstruktur sehingga pembaca awam mengerti ringkasan tentang *game* yang dibuat [ROG-10:60]. 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-10:62]. Langkah terkahir 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-10:75].

3.2.1.1. *One Sheet Document*

One sheet document game Slash Word ditunjukan pada tabel 3.1.

Tabel 3.1. *One Sheet Document Game* Slash Word

NO	ELEMEN	KETERANGAN
1	***Game Title***	Slash Word
2	***Game Outline***	• Permainan menebak gambar dalam bahasa Inggris • Pemain diharuskan mengumpulkan huruf – huruf secara urut untuk membentuk nama dari gambar benda soal yang muncul ke dalam bahasa inggris. • Huruf yang tidak terpakai dalam pembentukan jawaban soal dihilangkan dengan menggerakkan kursor pemain hingga mengenai huruf tersebut.
3	***Target User***	Semua Umur
4	***Target hardware***	PC Windows
5	***Unique Selling Point***	• ***Natural Interaction techniques*** **:** kursor game digerakkan dengan gerakan tangan menggunakan sensor Kinect. • ***Voice Command*** **:** Penggunaan perintah suara untuk navigasi menu game. • ***Fun and Active Gameplay*** **:** menghilangkan huruf – huruf yang salah dengan kursor yang digerakan menggunakan gerakan tangan.

		• ***Educational Content*** **:** Materi game berisi pembelajaran bahasa Inggris untuk menebak nama – nama dari gambar yang muncul di layar.

3.2.1.2. *Ten Page Design Document*

Ten page design menjelaskan secara garis besar tentang game yang dibuat. Dengan demikian diharapkan pembaca mengetahui *game* yang dibuat seperti apa tanpa harus mengetahui produk akhirnya.

1. *Title Page*

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

Game Title **:** Slash Word

Gambar 3.2. Logo *game* Slash Word

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

Target User **:** semua umur

Logo : logo *game* Slash Word ditunjukan pada gambar 3.2.

2. *Gameplay*

Slash Word merupakan permainan *single player*. Pemain harus menjawab soal gambar yang muncul dalam bahasa Inggris. Jawaban soal gambar diperoleh dari huruf – huruf yang turun di game area, Huruf yang tidak terpakai dalam pembentukan jawaban soal dihilangkan dengan menggerakkan kursor pemain

dengan gerakan telapak tangan hingga mengenai huruf tersebut. Permainan untuk mendapatkan atau memecahkan nilai tertinggi.

3. *Game Controller*

Kendali pada permainan Slash Word yang utama yaitu menggunakan gerakan tangan memanfaatkan sensor kamera kinect. Kendali permainan yang kedua yaitu menggunakan mouse, kendali mouse merupakan alternatif kendali kinect yang dapat dipilih pada menu pemilihan kendali permainan. Pada game Slash Word terdapat fitur *voice command* untuk mengakses menu – menu *game.*

4. ***Main Gameplay Concepts and Platform Specific Features***

- Genre : *Action – education.*
- *Game Level,* Terdapat 3 level dalam Permainan Slash Word yang dapat dipilih yaitu *easy, medium* dan *hard*
- *Platform* : PC Windows
- Features
  - ***Natural Interaction techniques*** : kursor game digerakkan dengan gerakan tangan menggunakan sensor Kinect.
  - ***Voice Command* :** Penggunaan perintah suara untuk navigasi menu game.
  - ***Fun and Active Gameplay*** : menghilangkan huruf – huruf yang salah dengan kursor yang digerakan menggunakan gerakan tangan.
  - ***Educational Content*** : Materi game berisi pembelajaran bahasa Inggris untuk menebak nama – nama dari gambar yang muncul di layar.

5. ***Game World***

Arena permainan Slash Word bertema ruangan - ruangan atau tempat yang ada di sekolah seperti ruangan kelas dan perpusatakaan.

6. ***Interface***

Pada *game* Slash Word dirancang 10 *screen* yang terdiri dari menu utama, menu *high score, credit screen,* menu *level,* menu *device mode, loading screen, how to play screen, in game screen,* menu *pause* dan *end score screen*. Gambar latar belakang pada permainan Slash Word merupakan gambar dua dimensi dengan kamera yang tetap. Sistem HUD pada permainan Slash Word memberikan informasi waktu, *player life,* skor pemain, gambar soal serta susunan huruf yang telah dibentuk oleh pemain.

3.2.2. *Technical Design*

Pada tahap *technical design,* pemodelan sistem dilakukan dengan menggunakan UML (*Unified Markup Language*). *Technical design* berupa :

1. Pembuatan diagram use case untuk memetakan interaksi antara aktor dengan fitur pada sistem atau perangkat lunak. Sebelum membuat diagram use case, ada dua langkah yang harus dilakukan yaitu :
   a. **Identifikasi aktor**
      Tahapan identifikasi aktor dilakukan untuk mengetahui aktor - aktor yang terlibat dalam *game*.
   b. **Identifikasi kebutuhan**
      Identifikasi kebutuhan terdiri dari identifikasi 2 kebutuhan yaitu :
      a. Kebutuhan fungsional
      b. Kebutuhan non fungsional

   Setelah melakukan dua tahap di atas, maka langkah terakhir adalah untuk memetakan interaksi antara aktor dengan kebutuhan pada sistem.
2. Pembuatan diagram kelas untuk memodelkan kelas-kelas dan interface-interface yang akan dibutuhkan dalam pembuatan *game*.
3. Pembuatan diagram aktifitas untuk menggambarkan urutan aktivitas dari proses pada setiap *use case* yang ada.

3.3. Implementasi *Game*

Setelah melakukan perancangan, maka *game* Slash Word akan dibuat dengan mengacu pada perancangan *game* Slash Word tersebut. Tahap ini

merupakan bagian dari tahap selanjutnya dalam siklus produksi *game* yaitu *production*, yang telah dijelaskan pada subbab 2.5. Pembuatan *game* menggunakan *game engine* 3D dengan bahasa pemrograman berorientasi objek.

3.4. Pengujian *Game*

Pengujian *game* dilakukan untuk memastikan bahwa *game* Slash Word telah sesuai dengan spesifikasi dari kebutuhan yang telah ditentukan sebelumnya. Pengujian *game* meliputi :

1) Pengujian unit

   Pengujian unit berfokus pada usaha verifikasi pada inti terkecil dari desain perangkat lunak. Pengujian unit menggunakan *White Box Testing* yang dilakukan pada beberapa metode (operasi) saja. Metode *White Box Testing* yang digunakan adalah *Basis Path Testing.* Metode ini melakukan pencarian ukuran kompleksitas perancangan prosedur dan menggunakannya untuk mendefinisikan sekumpulan jalur eksekusi dasar. Kasus uji diturunkan dari himpunan basis yang dijamin mengeksekusi seluruh pernyataan di program setidaknya satu kali selama pengujian. Pengukuran yang dilakukan pada pengujian *White Box* menggunakan pengukuran *cyclomatic complexity* yang ditentukan dari *flow graph* kode program. Rumus pengukuran *cyclomatic complexity* ditunjukkan pada Gambar 3.3.

2) Pengujian integrasi

   Pengujian integrasi menggunakan *White Box Testing* dengan metode *Basis Path Testing* yang diterapkan pada proses yang mengintegrasikan fungsionalitas dari beberapa klas untuk melakukan sebuah operasi tertentu.

$$V(G) = E - N + 2$$

Keterangam: E adalah jumlah *edge* dan N adalah jumlah *node*

Gambar 3.3. Rumus Pengukuran *Cyclomatic Complexity*

Sumber : [PRE-10]

3) Pengujian validasi

   Pengujian validasi menggunakan teknik *Black Box Testing* karena tidak berkosentrasi pada alur jalannya algoritma program. Pengujian dilakukan untuk mengetahui apakah *game* yang dibangun telah sesuai dengan daftar kebutuhan.

4) Pengujian kinerja dengan melihat FPS (*Frame Per Second*) yang dihasilkan *game*. Fasilitas untuk melihat FPS telah disediakan oleh *game engine*.

3.5. Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian *game* telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap *game* yang telah dibuat. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan - kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi *game* selanjutnya.

BAB IV
PERANCANGAN

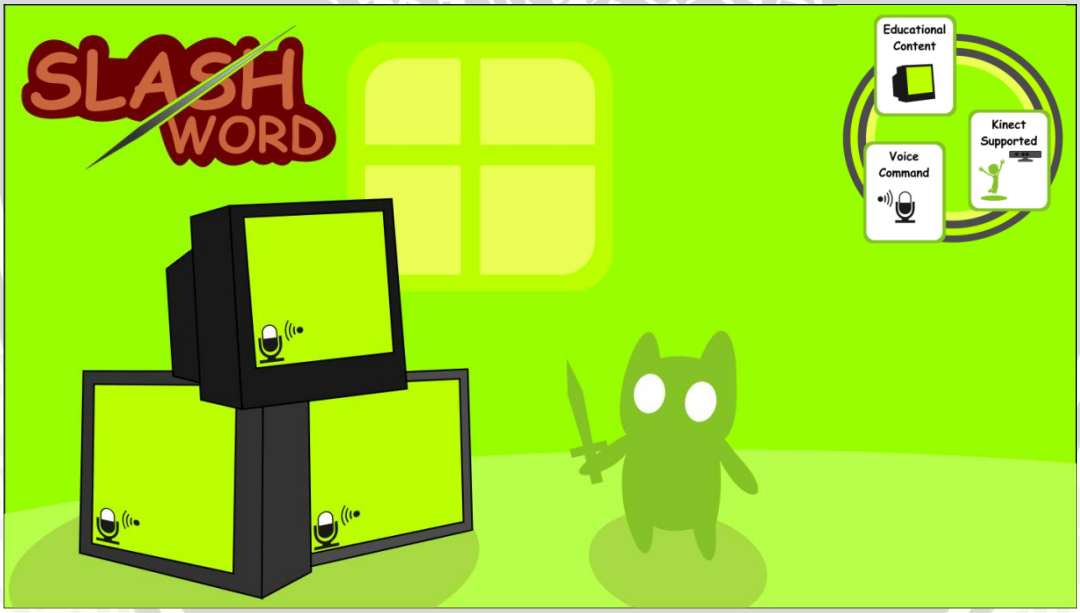
Bab ini membahas mengenai perancangan *game*. Perancangan *game* merupakan bagian dari tahap *preproduction* yang terdiri dari dua tahap. Tahap pertama yaitu *game concept design* dan tahap kedua yaitu *technical design.*

4.1. *Game Concept Design*

Bagian ini akan membahas *game concept design* Slash Word yang merupakan penjabaran lebih detail dari *ten page design* yang telah dibuat pada bab metodologi penelitian.

4.1.1 *Cover*

Gambar 4.1 menunjukan gambar *cover* dari *game* Slash Word.



Gambar 4.1. *Cover Game* Slash Word

4.1.2 *Gameplay Descriptions*

Slash Word merupakan permainan *single player*. *Game* Slash Word bermateri edukasi pembelajaran kosakata bahasa Inggris, dimana pemain diharuskan mengumpulkan huruf – huruf secara urut untuk membentuk nama dari

gambar benda yang harus ditebak dalam bahasa Inggris. Dalam mengumpulkan huruf, pemain harus menghilangkan huruf yang tidak terpakai dalam pembentukan jawaban soal dengan menggerakkan kursor pemain hingga mengenai huruf tersebut. Kursor pemain akan mengikuti arah gerakan telapak tangan pemain. Pemain harus berhati – hati terhadap bom yang muncul diantara huruf tersebut. Bom yang terkena kursor pemain akan mengurangi *life* pemain. Kumpulkan poin sebanyak mungkin dengan menjawab semua gambar soal dengan benar dan cepat. Setiap satu permainan pemain diberi waktu 10 menit dan 3 *life*. Permainan berakhir ketika waktu atau *life* pemain habis. Pemain dengan skor tertinggi akan masuk dalam *Top 5 High Scores*. Pemain dapat bergantian memainkan permainan Slash Word untuk saling memecahkan skor tertinggi.

Perancangan penerapan teknik interaksi alami dalam *gameplay* yaitu dimana pemain secara bebas menggerakkan salah satu telapak tangannya untuk menggerakkan kursor berbentuk telapak tangan di area permainan. Sebelum dapat menggerakkan kursor berbentuk telapak tangan tersebut, salah satu telapak tangan pemain yang ingin digunakan untuk menggerakkan kursor harus melakukan gerakan melambai agar telapak tangan pemain tersebut dapat dikenali sistem. Kursor di area permainan akan mengikuti posisi salah satu telapak tangan pemain yang telah dikenali oleh sistem dalam koordinat sumbu X dan sumbu Y. Kursor akan melakukan animasi melambai jika telapak tangan pemain yang telah dikenali oleh sistem melakukan gerakan melambai. Kursor akan melakukan animasi mendorong jika telapak tangan pemain yang telah dikenali oleh sistem melakukan gerakan mendorong. Kursor telapak tangan digunakan untuk menghilangkan huruf yang tidak terpakai dalam pembentukan jawaban soal.

4.1.3 *Starting a new game*

Untuk memulai game baru, pemain harus memilih level permainan terlebih dahulu setelah itu pemain harus memilih *device* kendali yang digunakan. Sebelum permainan dimulai akan muncul layar penjelasan cara bermain. Pemain diberi 3 *life* setiap bermain.

4.1.4 *Ending the game*

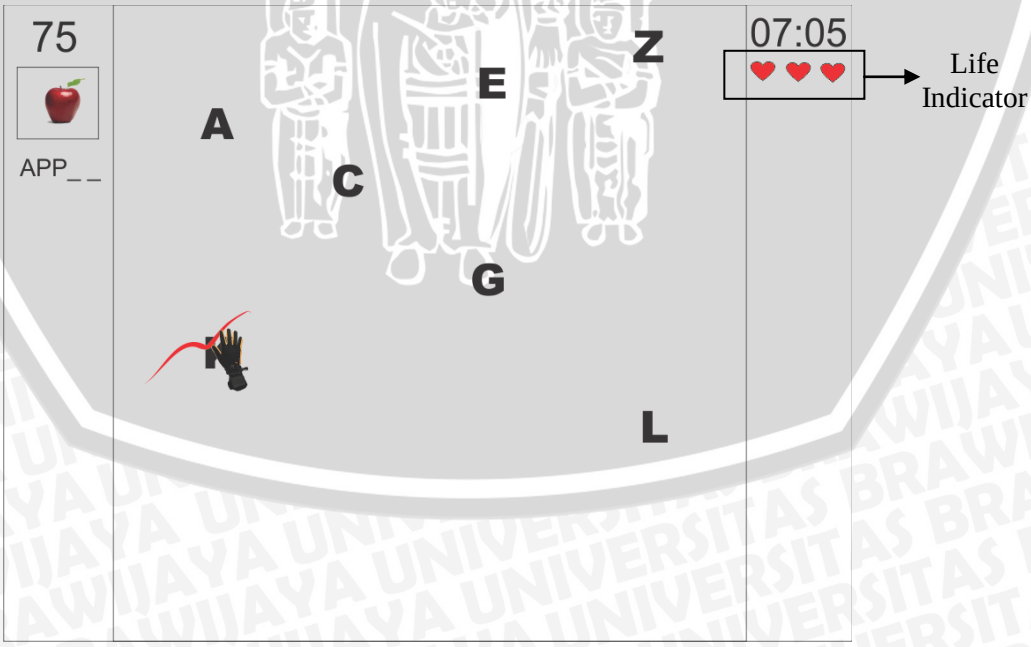
Permainan berakhir ketika waktu atau *life* pemain habis. Pada akhir permainan akan ditampilkan skor yang telah didapatkan oleh pemain. Pemain dengan skor tertinggi akan masuk dalam *Top 5 High Scores*. Pemain dapat bergantian memainkan permainan Slash Word untuk saling memecahkan skor tertinggi.

4.1.5 *Time limit*

Waktu yang diberikan dalam tiap satu permainan adalah 10 menit.

4.1.6 *Life*

Pemain diberi 3 *life* setiap kali bermain. Jika pengurangan poin terjadi ketika skor pemain 0, maka pemain akan mendapat pengurangan 1 *life*. Jika kursor pemain menyentuh item bom, maka pemain akan mendapat pengurangan 1 *life*. Permainan berakhir ketika waktu atau *life* pemain habis. Pemain dengan skor tertinggi akan masuk dalam *Top 5 High Scores*. Rancangan indikator life pemain dalam rancangan HUD *game* Slash Word ditunjukan pada gambar 4.2.



Gambar 4.2. Rancangan HUD Slash Word

4.1.7 *Victory and Loss Conditions*

Permainan Slash Word merupakan permainan untuk saling memecahkan nilai tertinggi yang pernah didapatkan oleh pemain dengan waktu tiap permainan adalah 10 menit. Hanya lima nilai tertinggi yang akan masuk dalam *top 5 high scores*.

4.1.8 *Scoring*

Skor awal pemain adalah 0. Pemain harus menghilangkan box huruf yang tidak sesuai dalam pembentukan jawaban soal sebelum box jatuh di area batas bawah permainan dengan menggunakan kursor pemain. Jika box huruf yang jatuh di area batas bawah permainan adalah huruf yang tepat dalam pembentukan jawaban, maka pemain akan mendapat tambahan 10 poin. Jika box huruf yang jatuh di area batas bawah permainan adalah huruf yang tidak tepat dalam pembentukan jawaban, maka pemain akan mendapat pengurangan 10 poin. Pemain dengan skor tertinggi akan masuk dalam *Top 5 High Scores*.

4.1.9 *Levels*

Permainan terdiri dari tiga level yaitu *easy, medium,* dan *hard*. level *easy* dapat digunakan oleh pemain untuk penyesuaian diri terhadap permainan sebelum memainlan level yang lebih sukar. Game level pada permainan Slash Word adalah:

1) ***Easy***, tingkat permainan lebih mudah dengan kecepatan pergerakan item yang rendah, sedikit bom yang muncul.
2) ***Medium***, tingkat permainan yang lebih sukar dibandingkan dengan level easy, dengan kecepatan pergerakan item yang lebih cepat dibandingkan dengan level easy, lebih banyak bom yang muncul dibandingkan dengan level easy.
3) ***Hard***, tingkat permainan yang paling sukar dibandingkan dengan level yang lain, dengan kecepatan pergerakan item paling cepat dibandingkan dengan level lain, lebih banyak bom yang muncul dibandingkan dengan level lain.

4.1.10 *Game Controller*

Kendali pemain pada permainan Slash Word adalah sebagai berikut :

1) Sensor Kinect

Sensor Kinect merupakan *device* utama dalam permainan Slash Word. Dengan menggunakan sensor kinect, permainan Slash Word dapat digerakan sepenuhnya menggunakan gerakan tangan dan perintah suara, kecuali pada pengisian nama pemain.

2) *Voice Command*

Daftar perintah suara pada permainan Slash Word :

- ***High Score***, perintah ini digunakan untuk masuk ke menu high score.
- ***Let's Play***
  - Perintah ini digunakan untuk masuk ke menu pemilihan level jika diucapkan pada menu utama,
  - Perintah ini digunakan untuk masuk ke *state* permainan jika diucapkan setelah melewati loading *screen*.
- ***Easy***, perintah ini digunakan untuk memilih level permainan paling mudah pada menu pemilihan level.
- ***Medium***, perintah ini digunakan untuk memilih level permainan pada tingkat tengah dalam menu pemilihan level.
- ***Hard***, perintah ini digunakan untuk memilih level permainan paling sukar pada menu pemilihan level.
- ***Use Kinect***, memilih mode kendali kinect pada menu pemilihan *device*.
- ***Use Mouse***, memilih mode kendali mouse pada menu pemilihan *device*.
- ***Previous***, perintah ini digunakan untuk kembali ke menu sebelumnya.
- ***Game Pause***, perintah ini digunakan untuk menghentikan sementara permainan yang sedang berjalan dan masuk menu pause.
- ***Resume***, perintah ini digunakan untuk menjalankan kembali permainan dari menu pause.
- ***Menu***, perintah ini gunakan untuk kembali ke menu utama dari menu pause.

- ***Submit***, perintah ini digunakan pada menu skor setelah pemain mengisi nama pemain untuk memasukan nama pemain ke dalam daftar top 5 high score jika skor pemain masuk ke dalam top 5 high score dan setelah itu sistem kembali ke menu utama.
- ***Exit,*** pada menu utama perintah ini digunakan untuk keluar dari *game* Slash Word
- ***Credit,*** perintah ini digunakan untuk ke *credit screen* menu utama.

3) Keyboard

Fungsi *keyboard* yang utama adalah untuk memasukan nama pemain pada menu skor. Selain itu fungsi tombol *keyboard* yang lain adalah :

1) **Esc**, digunakan untuk menghentikan sementara permainan yang sedang berjalan dan masuk menu pause.

2) **Space**, digunakan untuk menjalankan kembali permainan dari menu pause.

Fungsi lain *keyboard* selain fungsi utama semua telah terwakili oleh fitur perintah suara.

4) Mouse

Mouse bukan device utama dalam permainan Slash Word. Fungsi mouse dapat digunakan untuk memilih menu pada menu utama dan menggerakkan kursor pada permainan pada mode kendali mouse**.**

4.1.11 *Game Screens*

Rancangan tampilan antar muka *game* Slash Word terdiri dari :

- *Menu Game*.
  - *Main Menu*.

    Menu utama yang pertama kali muncul pada game Slash Word. Prototipe *Main Menu* ditunjukan pada gambar 4.3.
  - *Game Level Setting Menu.*

    Menu untuk memilih level permainan. Prototipe *Game Level Setting Menu* ditunjukan pada gambar 4.4.

- *High Score Menu*

  Menu untuk melihat 5 nilai tertinggi yang telah tersimpan dalam permainan Slash Word. Prototipe *High Score Menu* ditunjukan pada gambar 4.5.

- *Pause Menu*

  Menu yang muncul ketika permainan dalam kondisi *pause*. Prototipe *Pause Menu* ditunjukan pada gambar 4.6.

- *Device Setting Menu*

  Menu untuk memilih *device* yang digunakan dalam permainan. Prototipe *Device Setting* ditunjukan pada gambar 4.7.

- *Loading screen*

  Tampilan ketika *game* sedang dalam proses *loading.* Prototipe loading screen ditunjukan pada gambar 4.8.

- *How to play screen*

  Tampilan informasi mengenai cara bermain. Prototipe How to play screen ditunjukan pada gambar 4.9.

- HUD (*Heads-Up Display*) *game.*

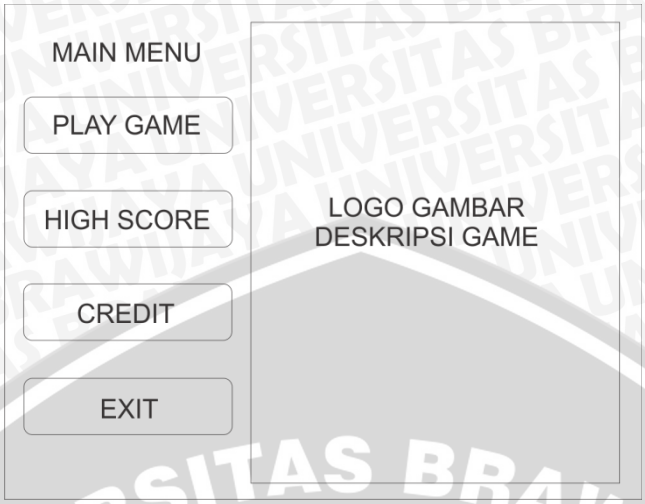
  Tampilan yang berisi informasi mengenai waktu, *life*, soal dan jawaban pemain. Prototipe HUD ditunjukan pada gambar 4.10.

- *In game state.*

  Tampilan dalam permainan. Prototipe *In game state* ditunjukan pada gambar 4.11.

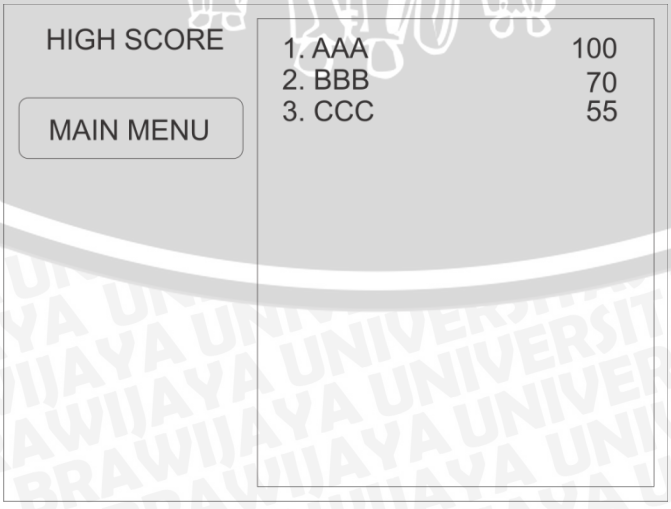
- *End game score screen.*

  Tampilan ketika permainan berakhir. Prototipe *end game score screen* ditunjukan pada gambar 4.12.

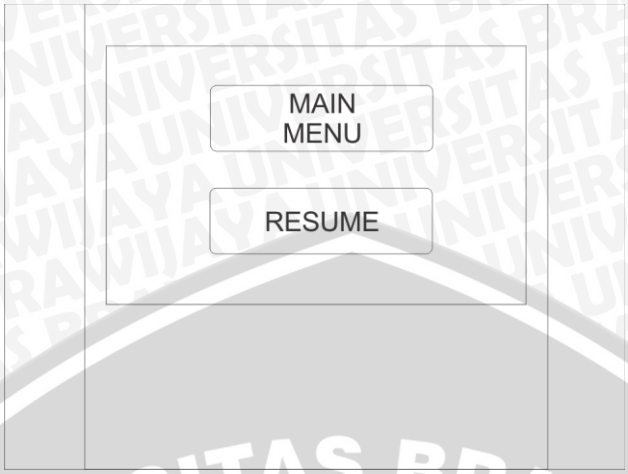


Gambar 4.3. *Prototipe Main Menu*

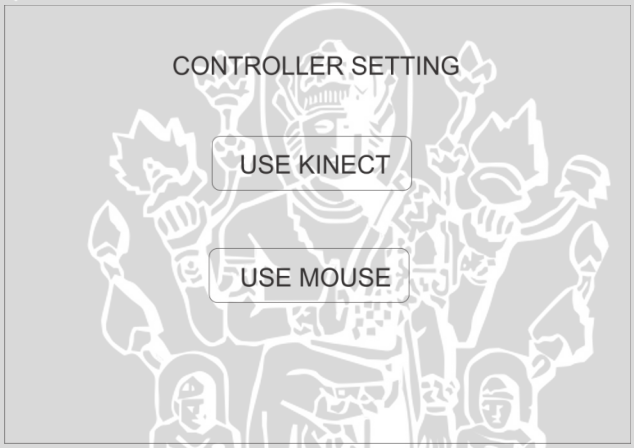
Gambar 4.4. *Prototipe Game Level Setting Menu.*



Gambar 4.5. *Prototipe High Score Menu.*



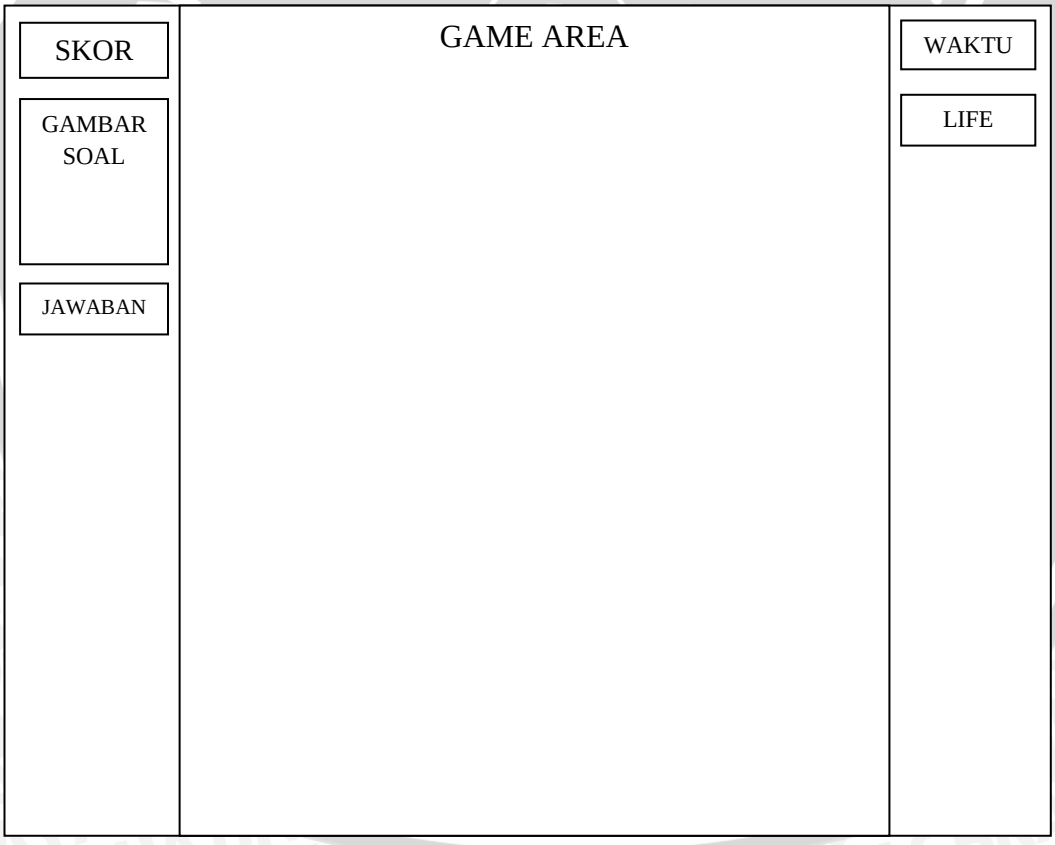
Gambar 4.6. *Prototipe Pause Menu.*



Gambar 4.7. *Prototipe Device Setting Menu.*

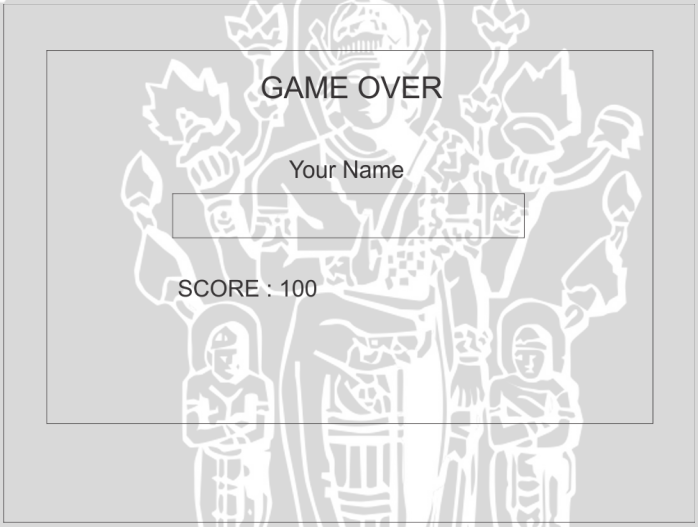
Gambar 4.8. *Prototipe Loading Screen.*

Gambar 4.9. *Prototipe How To Play Screen.*



Gambar 4.10. *Prototipe* HUD (*Heads-Up Display*).

Gambar 4.11. *Prototipe In Game State.*

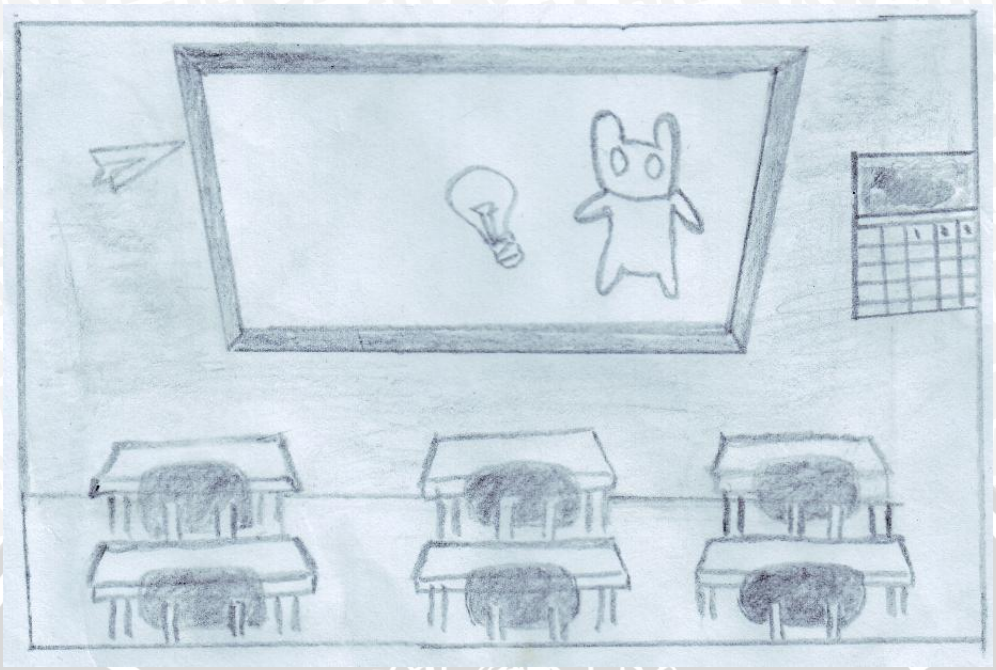


Gambar 4.12. *Prototipe End Game Score Screen.*

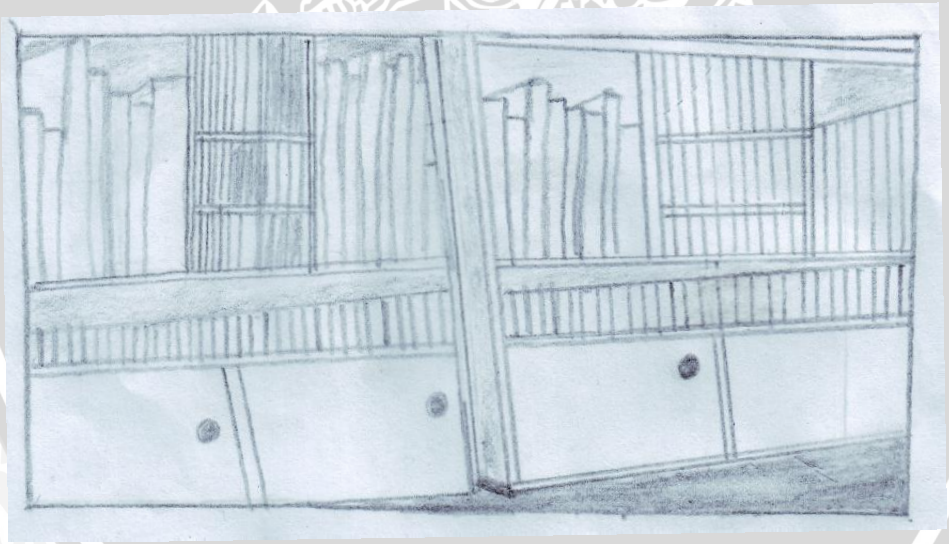
4.1.12 *Environment*

Arena permainan Slash Word bertema ruangan - ruangan yang ada di sekolah. Berikut 2 prototipe arena permainan slash word :

1) Arena ruang kelas ditunjukan oleh gambar 4.13.
2) Arena perpustakaan ditunjukan oleh gambar 4.14.



Gambar 4.13. *Prototipe* Ruang Kelas.

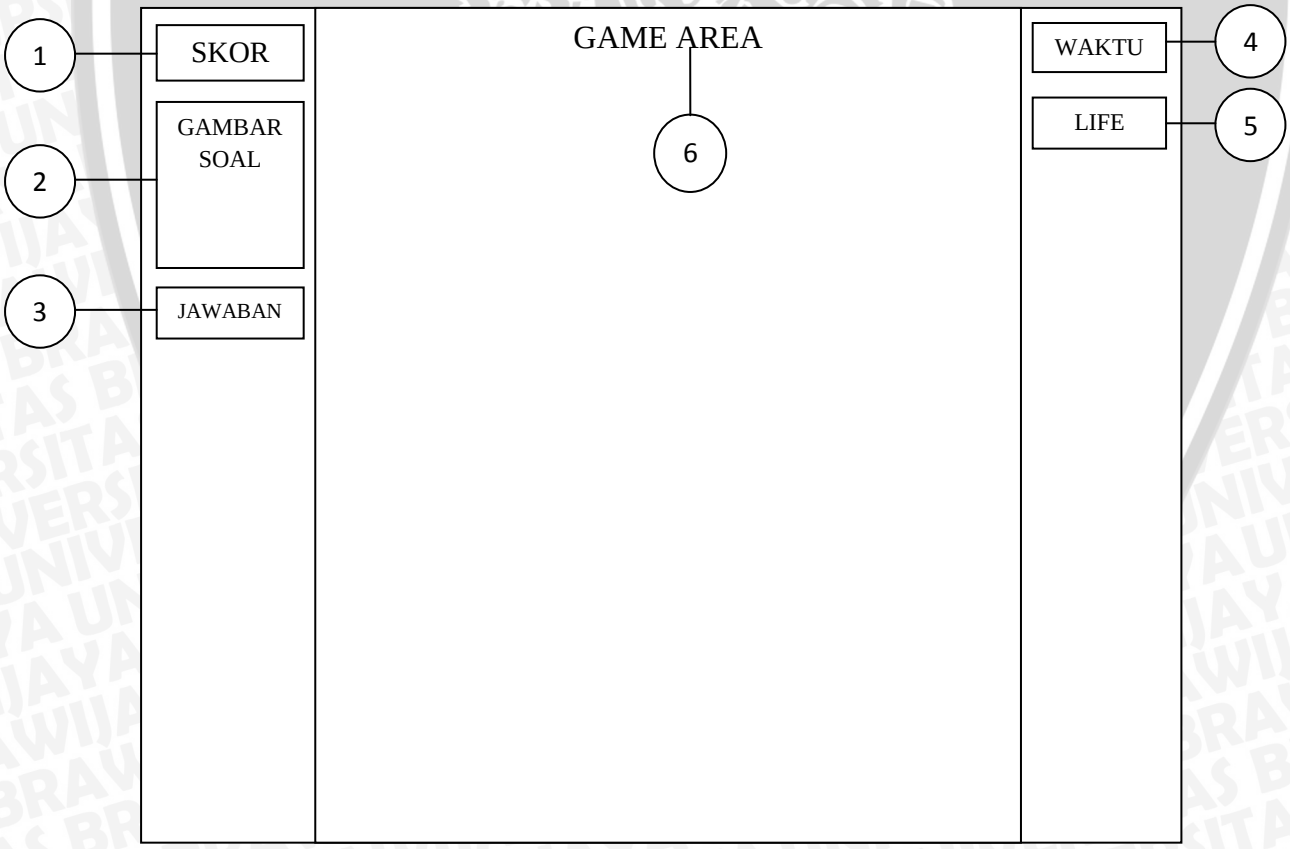


Gambar 4.14. *Prototipe* Ruang Perpustakaan.

4.1.13 HUD *System*

Prototipe HUD *(Heads-Up Display)* pada *game* Slash Word ditunjukan pada gambar 4.15. Berikut penjelasan keterangan nomor pada gambar 4.15 :

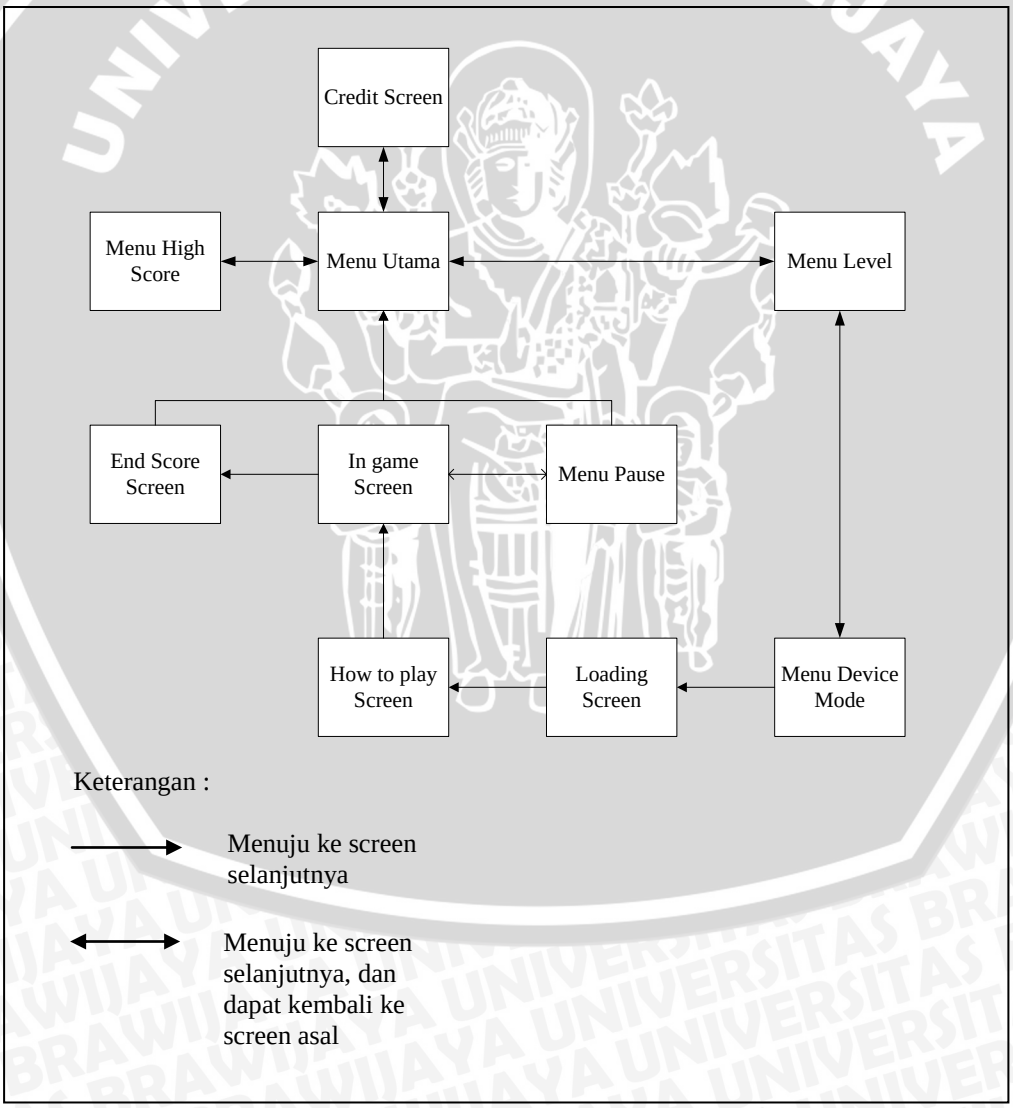
1) **Skor,** panel skor berisi skor pemain.
2) **Gambar Soal,** panel gambar soal berisi gambar soal yang muncul secara random yang harus ditebak dalam bahasa inggris.
3) **Jawaban,** panel jawaban berisi susunan huruf jawaban yang telah dibentuk oleh pemain.
4) **Waktu,** panel waktu berisi sisa waktu permainan.
5) **Life,** panel life berisi sisa life dari pemain
6) ***Game area,*** pada game area ini merupakan tempat munculnya item huruf dan bom. Kursor telapak tangan hanya dapat digerakkan sebatas *game area* tersebut



Gambar 4.15. *Prototipe* HUD (*Heads-Up Display*).

4.1.14 *Sitemap*

Pada permainan Slash Word dirancang 10 *screen* yang terdiri dari menu utama, menu *high score*, *credit screen*, menu *level*, menu *device mode*, *loading screen*, *how to play screen*, *ingame screen*, menu *pause* dan *end score screen*. Menu pertama yang tampil pada saat awal menjalankan permainan Slash Word adalah menu utama, Di menu utama terdapat beberapa pilihan menu yaitu menu *play game* untuk masuk ke menu pemilihan level permainan yang akan dimainkan, *high score* untuk melihat nilai tertinggi yang telah tersimpan dalam permainan dan menu *credit* dari permainan Slash Word. Pada gambar 4.16 menunjukan *sitemap* dari *game* Slash Word.



Gambar 4.16. *Sitemap Screen Game* Slash Word.

4.1.15 *Game elements*

Prototipe elemen game slash word terdiri dari :

1) Gambar soal

Dirancang 3 prototipe untuk gambar soal yaitu gambar buah apel, gambar hewan kucing dan gambar pesawat terbang. Untuk implementasi game Slash Word akan dibuat 25 gambar soal untuk dijawab oleh pemain. Prototipe gambar soal buah apel ditunjukan pada tabel 4.1, prototipe gambar soal hewan ayam ditunjukan pada tabel 4.2 dan prototipe gambar soal pesawat terbang ditunjukan pada tabel 4.3.

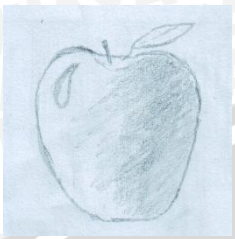
2) *Hand Cursor*

Prototipe *Hand Cursor* pemain ditunjukan oleh tabel 4.4. *Hand Cursor* ini yang akan digerakan oleh gerakan tangan pemain untuk menghilangkan huruf - huruf yang tidak tepat digunakan dalam pembentukan jawaban soal.

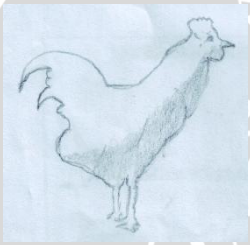
3) *Letter Box*

Prototipe *Letter Box* atau kotak huruf ditunjukan pada tabel 4.5. Terdapat 26 kotak huruf yaitu kotak huruf A sampai dengan kotak huruf Z. Kotak huruf inilah yang akan membentuk jawaban soal. Kotak huruf yang tidak tepat digunakan dalam pembentukan jawaban soal harus dihilangkan menggunakan kursor tangan pemain. Kotak huruf yang terkena kursor tangan pemain akan memunculkan animasi sprite sayatan sesaat sebelum kotak tersebut menghilang untuk menghasilkan kesan kotak huruf tersebut terpotong.

Tabel 4.1. *Prototipe* Gambar Soal Buah Apel.

Game Object	**Profil**
	**Nama** : *Apple*
	Keterangan : Gambar soal buah apel. Jawaban soal adalah *apple*.

Tabel 4.2. *Prototipe* Gambar Soal Hewan Ayam.

Game Object	**Profil**
	**Nama** : *Chicken*
	Keterangan : Gambar soal hewan ayam. Jawaban soal adalah *chicken*.

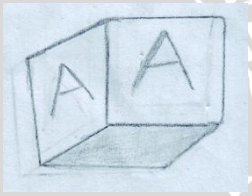
Tabel 4.3. *Prototipe* Gambar Soal Transportasi Pesawat Terbang.

Game Object	**Profil**
	**Nama** : *Plane*
	Keterangan : Gambar soal transportasi pesawat terbang. Jawaban soal adalah *plane*.

Tabel 4.4. *Prototipe* Hand Cursor.

Game Object	**Profil**
	**Nama** **:** *Hand Cursor*
	Keterangan : *Hand Cursor* ini yang akan digerakan oleh gerakan tangan pemain untuk menghilangkan huruf - huruf yang tidak tepat digunakan dalam pembentukan jawaban soal

Tabel 4.5. *Prototipe* Kotak Huruf.

Game Object	**Profil**
	**Nama** **:** kotak huruf A
	Keterangan : Kotak huruf inilah yang akan membentuk jawaban soal. Kotak huruf yang tidak tepat digunakan dalam pembentukan jawaban soal harus dihilangkan menggunakan kursor tangan pemain. Terdapat 26 kotak huruf yaitu kotak huruf A sampai dengan kotak huruf Z

4.1.16 *Music and SFX*

Terdapat 3 musik latar pada permainan Slash Word yaitu :

1) Musik utama yang dimainkan pada menu utama, menu high score, menu level, dan menu device mode.
2) Musik ingame yang dimainkan pada ingame screen.
3) Musik endgame yang dimainkan pada end score screen.

Sound effect permainan Slash Word terdiri dari :

1) Bom ketika meledak.
2) Sumbu bom ketika bom muncul.
3) Huruf yang terkena kursor.
4) Huruf benar yang jatuh di area bawah permainan.
5) Huruf salah yang jatuh di area bawah permainan.
6) Sound effect ketika susunan huruf yang membentuk jawaban telah lengkap.

4.2. *Technical Design*

Pada tahap *technical design*, pemodelan sistem dilakukan dengan menggunakan UML (*Unified Markup Language*).

4.2.1. Perancangan Diagram *Use Case*

Diagram *use case* dalam perancangan *game* Slash Word digunakan untuk memodelkan fungsionalitas dari *game*. Diagram *use case* ini melibatkan *user* sebagai aktor dan beberapa *use case*. Sebelum merancang diagram *use case*, yang harus dilakukan adalah menidentifikasi aktor dan indentifikasi kebutuhan.

4.2.1.1. Identifikasi Aktor

Aktor adalah pemakai sistem, dapat berupa orang atau sistem terotomastisasi lain [HAR-04:268]. Aktor dari permainan Slash Word adalah pemain *game* Slash Word.

4.2.1.2. Identifikasi Kebutuhan

Keseluruhan kebutuhan fungsional dan non fungsional *game* Slash Word dimasukan ke dalam tabel daftar kebutuhan. Daftar kebutuhan fungsional dan non fungsional keseluruhan sistem ditunjukan oleh tabel 4.6.

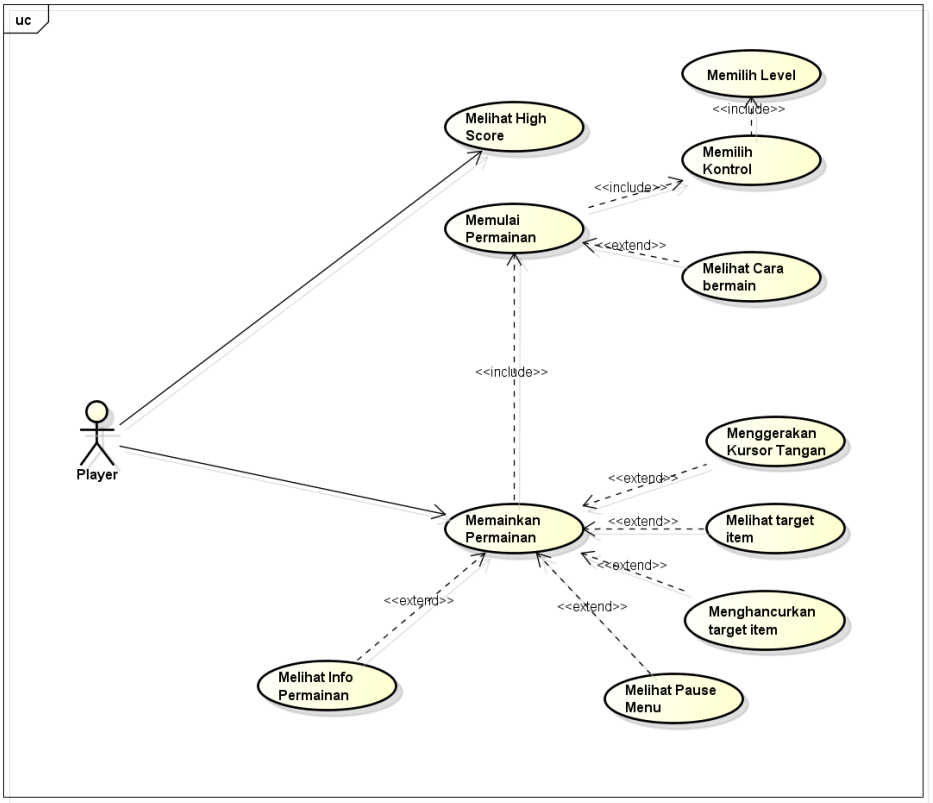
Tabel 4.6. Daftar Kebutuhan Fungsional dan Non Fungsional.

ID	***Requirements***	**Aktor**	**Nama *Use Case***
F01	Perangkat lunak harus menyediakan antarmuka untuk menampilkan berbagai menu utama permainan	Pemain.	Melihat menu utama.
F02	Perangkat lunak harus menyediakan antarmuka untuk melihat *High Score* pemain.	Pemain.	Melihat *High Score*.
F03	Perangkat lunak harus menyediakan antarmuka untuk proses memulai permainan.	Pemain.	Memulai permainan.
F04	Perangkat lunak harus menyediakan antarmuka untuk memilih *level* permainan.	Pemain.	Memilih level.
F05	Perangkat lunak harus menyediakan antarmuka menu *device mode* untuk memilih kendali permainan.	Pemain.	Memilih kendali permainan.
F06	Perangkat lunak harus menyediakan tampilan loading *screen* ketika sistem melakukan proses memuat asset – asset permainan.	Pemain.	Melihat *loading screen.*
F07	Perangkat lunak harus menyediakan tampilan cara bermain sebelum pemain memulai permainan.	Pemain.	Melihat cara bermain.
F08	Perangkat lunak harus menyediakan antarmuka untuk masuk ke *state* bermain Slash Word	Pemain	Memainkan permainan
F09	Perangkat lunak harus menyediakan proses untuk menghentikan sementara permainan yang berlangsung dan menampilkan menu	Pemain.	Melihat *pause menu*.

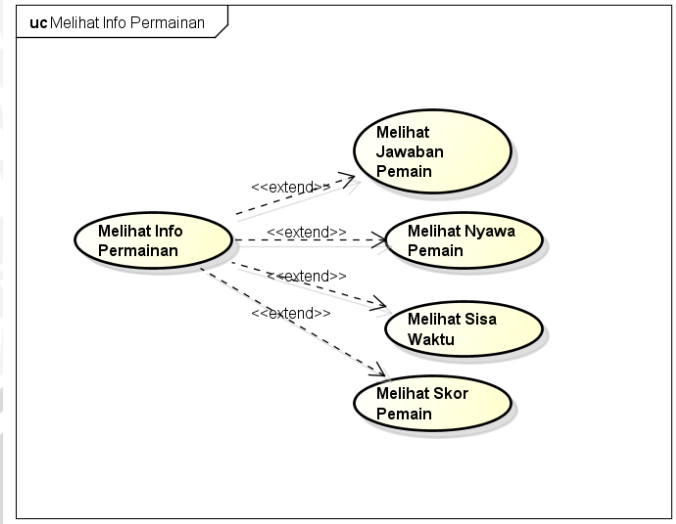
	pause yang memberikan pilihan untuk kembali ke permainan atau keluar dari game.		
F10	Perangkat lunak harus menyediakan proses untuk mendeteksi gerakan tangan pemain untuk menggerakkan kursor tangan.	Pemain.	Menggerakkan kursor tangan.
F11	Perangkat lunak harus menyediakan komponen untuk menampilkan informasi sisa nyawa pemain.	Pemain.	Melihat nyawa pemain.
F12	Perangkat lunak harus menyediakan komponen untuk menampilkan informasi skor pemain.	Pemain.	Melihat skor pemain.
F13	Perangkat lunak harus menyediakan komponen untuk menampilkan sisa waktu permainan.	Pemain.	Melihat sisa waktu.
F14	Perangkat lunak harus menyediakan komponen untuk menampilkan gambar soal dan proses untuk merandom gambar soal yang muncul.	Pemain	Melihat gambar soal.
F15	Perangkat lunak harus menyediakan komponen untuk menampilkan jawaban pemain.	Pemain.	Melihat jawaban pemain.
F16	Perangkat lunak harus menyediakan proses untuk memproduksi dan menampilkan item target secara random.	Pemain.	Melihat target item.
F17	Perangkat lunak harus menyediakan proses untuk menghilangkan *target item* jika mengenai kursor tangan	Pemain	Menghancurkan target item

	player.		
N01	Perangkat lunak harus dapat menghasilkan rata – rata nilai *frame per second* lebih dari sama dengan 30 FPS	------	------

Daftar kebutuhan fungsional yang didapat dari proses analisis kebutuhan akan dimodelkan dalam *use case diagram.* Gambar 4.17 dan 4.18 menunjukan diagram *use case* dari *game* Slash Word.



Gambar 4.17. *Use Case Diagram Game* Slash Word.



Gambar 4.18. *Sub Use Case* Melihat Info Permainan.

4.2.2. Perancangan *Class Diagram*

Perancangan diagram kelas untuk memodelkan kelas - kelas dan *interface* – *interface* yang dibutuhkan dalam pembuatan *game* Slash Word. Perancangan diagram kelas pada *game* Slash Word ditunjukkan pada gambar 4.19 yang ada pada lampiran. Keterangan kelas ditunjukan pada tabel 4.7.

Tabel 4.7. Deskripsi Diagram Kelas *Game* Slash Word

No	Kelas	Deskripsi
1	InGame	Kelas Utama yang berisi algoritma utama permainan Slash Word.
2	Player	Kelas yang menangani status atribut pemain.
3	SoalGenerator	Kelas yang menangani bagian soal permainan.
4	HighScoreGenerator	Kelas yang menangani bagian nilai tertinggi permainan.
5	GeneratorAllItem	Kelas yang menangani bagian produksi target item dalam permainan.
6	RandomerCharNBom	Kelas yang melakukan pengacakan target item.
7	mStartScreen	Kelas kontrol untuk *start screen.*
8	mMenuScreen	Kelas kontrol untuk *menu screen.*

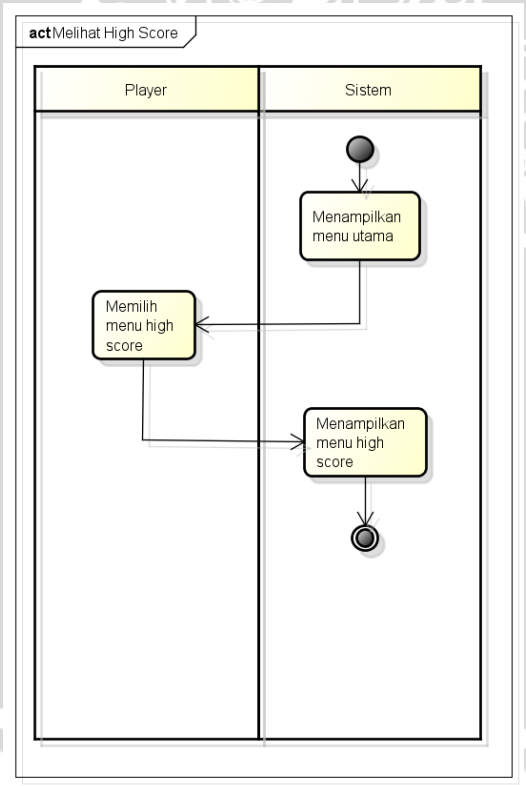
9	mHighScoreScreen	Kelas kontrol untuk *High Score screen.*
10	mLevelScreen	Kelas kontrol untuk *level screen.*
11	mDeviceScreen	Kelas kontrol untuk *device screen.*
12	mLoadingScreen	Kelas kontrol untuk *loading screen.*
13	mHowToPlayScreen	Kelas kontrol untuk *how to play screen.*
14	mHudScreen	Kelas kontrol untuk *Hud screen.*
15	mPauseScreen	Kelas kontrol untuk *pause screen.*
16	mUnameScreen	Kelas kontrol untuk *end game screen.*
17	mCreditScreen	Kelas kontrol untuk *credit screen.*
18	VoiceListener	Kelas yang menangani pengenalan perintah suara. Kelas ini sebagai library tambahan dalam *game enggine*. Operasi `OnListen()` dalam kelas ini yang akan digunakan secara overriding dalam kelas InGame.
19	KinectListenerImpl	Kelas yang menangani input yang diterima sensor kinect salah satunya pengenalan gerakan telapak tangan. Kelas ini sebagai library dalam *game enggine*. Operasi `OnPointUpdate()` dalam kelas ini yang akan digunakan secara overriding dalam kelas InGame.
20	AppStateAbs	Abstrak kelas yang berisi operasi abstrak untuk menangani state aplikasi
21	KinectListener	Interface untuk sensor kinect. Interface ini sebagai library dalam *game enggine*. berisi operasi – operasi abstrak untuk menangani input dari sensor kinect.

4.2.3. Perancangan *Activity Diagram*

Pembuatan *activity diagram* ini bertujuan untuk menggambarkan urutan aktivitas dari proses pada setiap *use case* yang ada. Berikut merupakan gambar dari masing-masing *activity diagram.*

4.2.3.1. *Activity Diagram* Melihat *High Score*

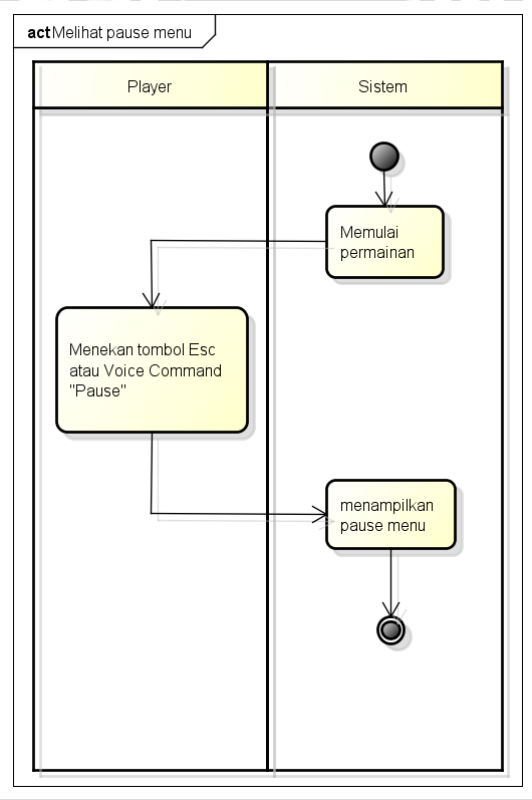
Pada gambar 4.20. menunjukan *activity diagram* melihat *high score,* yang menunjukan urutan proses dalam melihat *high score*. Penjelasan gambar 4.20 dalam proses melihat *high score* adalah pemain memilih menu *high score* dengan perintah suara *"high score"* atau dengan gesture menekan pada posisi kursor tepat berada di button *high score menu* pada menu utama. Sistem akan menampilkan halaman nilai tertinggi yang berisi lima nilai tertinggi dari pemain.



Gambar 4.20. *Activity Diagram* Melihat *High Score.*

4.2.3.2. *Activity Diagram* Melihat *Pause Menu*

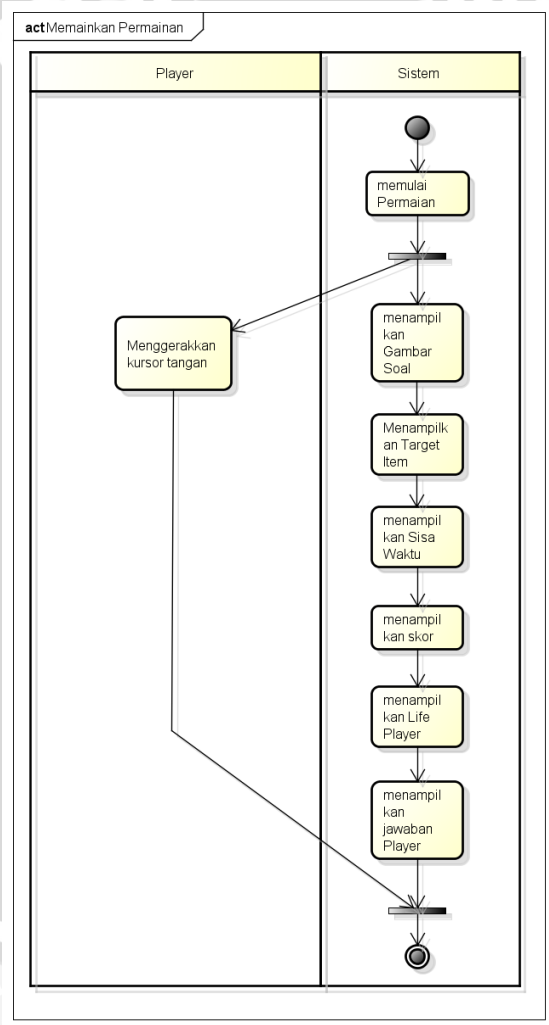
Pada gambar 4.21. menunjukan *activity diagram* melihat *pause menu*, yang menunjukan urutan proses dalam melihat *pause* menu. Penjelasan gambar 4.21 dalam proses menampilkan *pause* menu adalah pemain menekan tombol "Esc" atau dengan perintah suara pada saat permainan telah dimulai. Sistem akan menampilkan halaman *pause menu* yang akan menghentikan sementara permainan. Halaman *pause menu* memberikan pilihan untuk kembali ke permainan atau ke menu utama.



Gambar 4.21. *Activity Diagram* Melihat *Pause Menu.*

4.2.3.3. *Activity Diagram* Memainkan Permainan

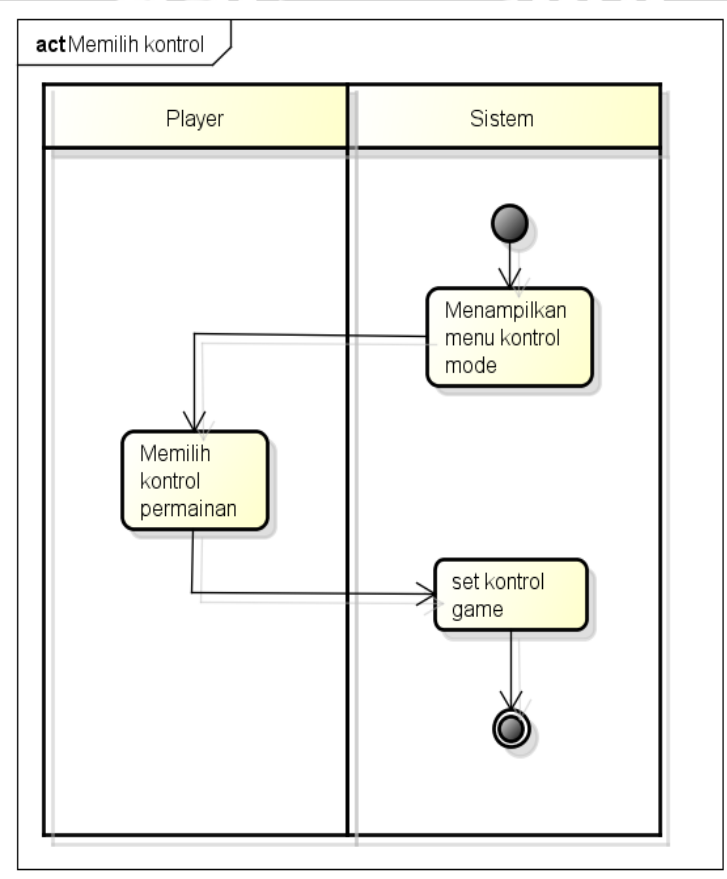
Pada gambar 4.22. menunjukan *activity diagram* memainkan permainan, yang menunjukan urutan proses dalam memainkan permainan. Penjelasan gambar 4.22 dalam proses memainkan permainan adalah Pemain dapat menggerakan kursor tangan dengan kendali permainan yang telah dipilih dan pemain dapat melihat informasi permainan yang disediakan sistem selama permainan berlangsung setelah proses memualai permainan telah selesai dilakukan oleh sistem.



Gambar 4.22. *Activity Diagram* Memainkan Permainan.

4.2.3.4. *Activity Diagram* Memilih Kendali Permainan

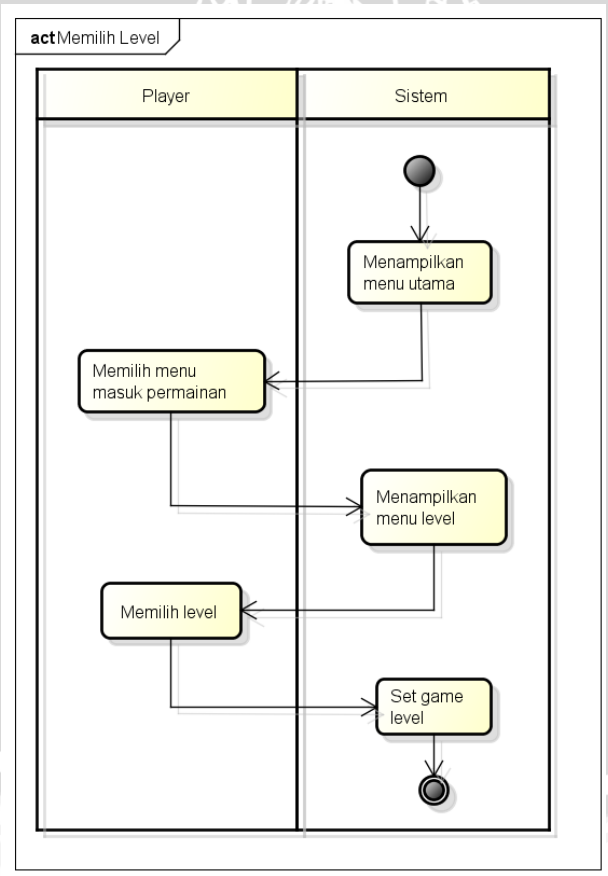
Pada gambar 4.23. menunjukan *activity diagram* memilih kendali permainan, yang menunjukan urutan proses dalam memilih kendali permainan. Penjelasan gambar 4.23 dalam proses memilih kendali permainan adalah pemain memilih kendali permainan pada saat sistem menampilkan halaman memilih kendali permainan. Pemain dapat memilih kendali menggunakan mouse atau kinect. Sistem akan membuat kursor tangan dapat dikendalikan oleh kendali yang telah dipilih oleh pemain.



Gambar 4.23. *Activity Diagram* Memilih Kendali Permainan.

4.2.3.5. *Activity Diagram* Memilih Level

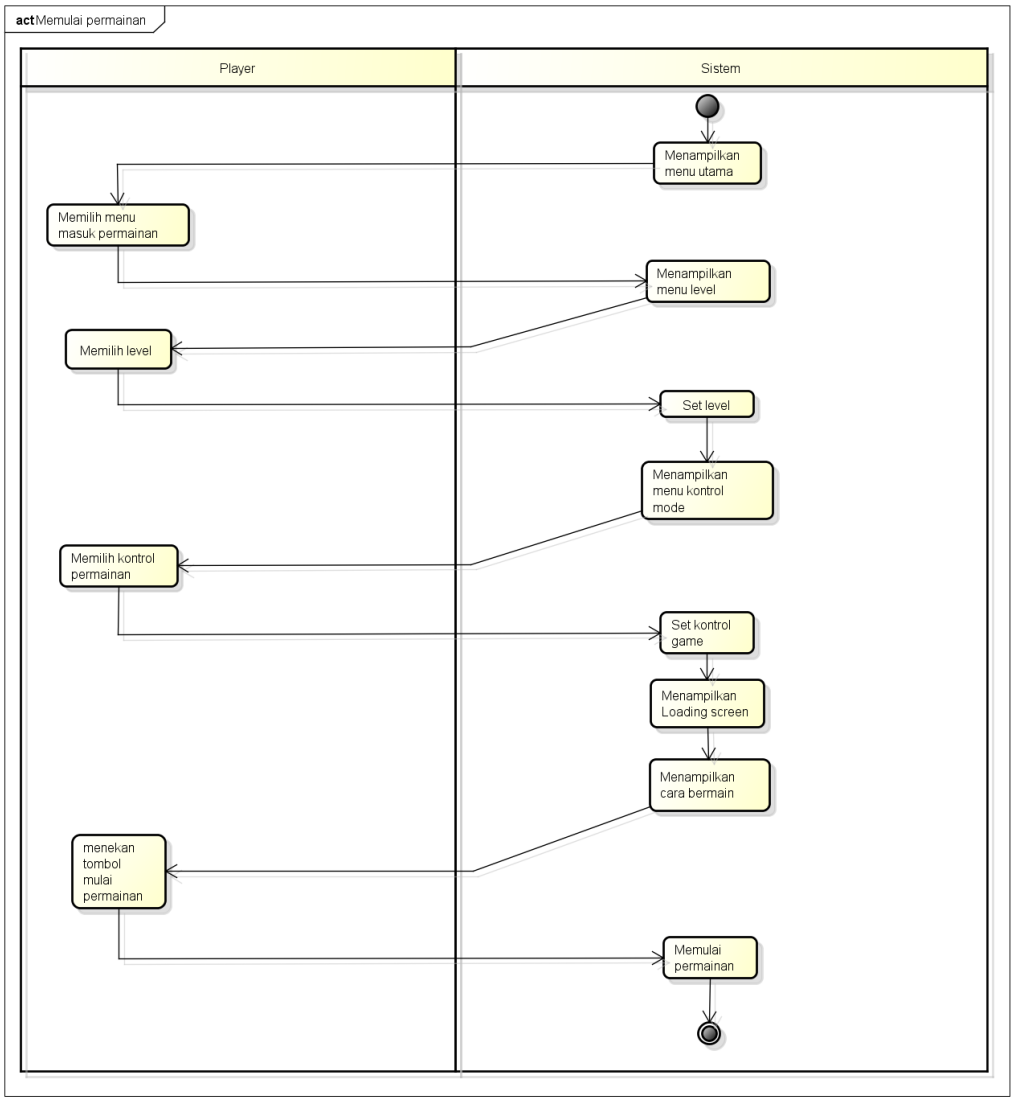
Pada gambar 4.24. menunjukan *activity diagram* memilih level, yang menunjukan urutan proses dalam memilih level. Penjelasan gambar 4.24 dalam proses memilih level permainan adalah pemain memilih menu memilih level melalui button yang telah disediakan di menu utama dengan perintah suara "*let's play*" atau dengan gesture menekan pada posisi kursor tepat berada di button tersebut. Sistem akan menampilkan halaman pilihan level dimana level permainan dapat dipilih oleh pemain. Pemain memilih level permainan pada halaman pilihan level dengan perintah suara *"easy"* , *"medium"* atau *"hard"*. Pemain juga dapat menggunakan gesture menekan pada posisi kursor tepat berada di button level yang ingin dipilih.



Gambar 4.24. *Activity Diagram* Memilih level.

4.2.3.6. *Activity Diagram* Memulai Permainan

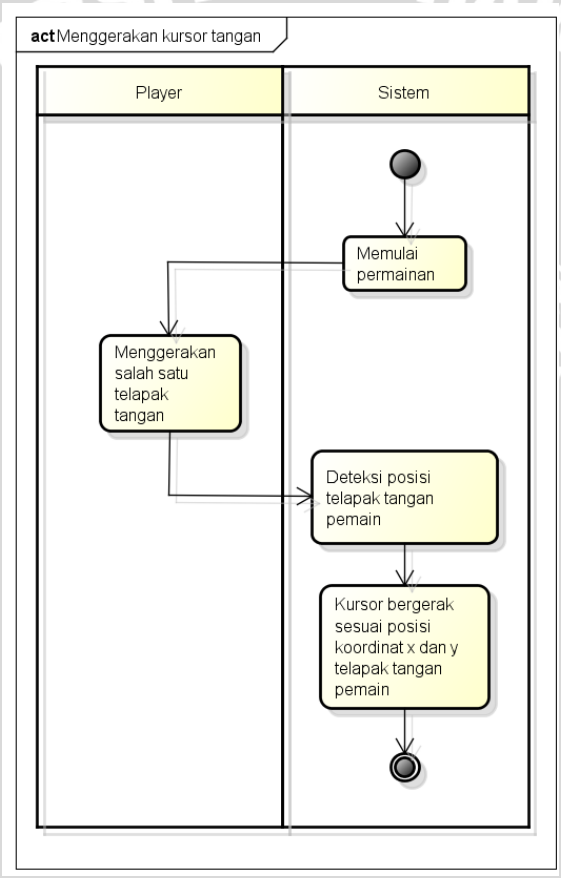
Pada gambar 4.25. menunjukan *activity diagram* memulai permainan, yang menunjukan urutan proses dalam memulai permainan. Penjelasan gambar 4.25 dalam proses memulai permainan adalah untuk memulai permainan, pemain harus memilih level permainan terlebih dahulu, setelah itu pemain harus memilih kendali permainan. Sistem akan menampilkan cara bermain sebelum permainan dimulai setelah pemain memilih level dan kendali permainan.



Gambar 4.25. *Activity Diagram* Memulai Permainan.

4.2.3.7. *Activity Diagram* Menggerakkan Kursor Tangan

Pada gambar 4.26. menunjukan *activity diagram* menggerakkan kursor tangan, yang menunjukan urutan proses dalam menggerakkan kursor tangan. Penjelasan gambar 4.26 dalam proses menggerakan kursor tangan adalah ketika sistem telah memulai permainan, pemain dapat menggerakan salah satu 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 4.26. *Activity Diagram* Menggerakkan Kursor Tangan.

BAB V
IMPLEMENTASI

Pada bab ini dibahas mengenai implementasi *game* berdasarkan hasil yang telah didapatkan dari perancangan *game*. Tahap ini merupakan bagian dari tahap selanjutnya dalam siklus produksi game yaitu *production*, yang telah dijelaskan pada subbab 2.5. Pembahasan terdiri dari penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi, implementasi algoritma, dan implementasi antarmuka aplikasi.

5.1. Spesifikasi Sistem

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

5.1.1. Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang dipakai dalam proses pengembangan dijelaskan pada tabel 5.1.

Tabel 5.1. Spesifikasi Perangkat Keras Komputer

Notebook ASUS A43TA	
Processor	AMD ® APU A6-3400M
Memory (RAM)	4 GB
Harddisk	750 GB
Motherboard	AMD M780G Chipset
Graphic Card	AMD Radeon ® HD 6720G2 with 1GB DDR3 VRAM Built-in A6-3400M
Monitor	14.0" 16:9 HD (1366x768) LED Backlight
Motion Sensor	Microsoft Kinect

Sumber: Implementasi

5.1.2. Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang dipakai dalam proses pengembangan perangkat lunak dijelaskan pada tabel 5.2.

Tabel 5.2. Spesifikasi Perangkat Lunak Komputer.

Notebook ASUS A43TA	
Operating System	Microsoft Windows 7 (6.1) Ultimate Edition (Build 7600)
DirectX Version	DirectX 11
Programming Language	Java
Software Development Kit	Java Development Kit 7 Update 7, jMonkey Engine SDK 3 RC 2, openni-win64-1.5.2.23-dev, nite-win64-1.5.2.21-dev, SensorKinect091-Bin Win64-v5.1.0.25
Programming Environment	Java Runtime Environment 7
Editor	jMonkey Engine 3 Editor

Sumber: Implementasi

5.2. Batasan Implementasi

Batasan dalam mengimplementasikan *game* ini adalah sebagai berikut

1. *Game* Slash Word ini berjalan pada komputer dengan sistem operasi windows 7 atau versi diatasnya.
2. Pembuatan *game* Slash Word menggunakan jMonkey Engine 3 RC2.
3. Penggabungan sensor kendali *game* kinect dengan jMonkey Engine menggunakan *library* Kinect untuk jMonkey Engine 3 yang dibuat oleh Glauco Márdano dengan basis API Openni 1.5.2.23 dan NITE 1.5.2.21.
4. Implementasi *voice command* menggunakan library CloudGarden's JSAPI versi 1.7
5. *Driver* kinect menggunakan SensorKinect091-Bin v5.1.0.25.
6. *Game* Slash Word merupakan *game single player*.

5.3. Implementasi Kelas

Setiap klas yang telah dirancang pada proses perancangan direalisasikan pada sebuah *file* program *.java. Dari 21 kelas di perancangan, hanya 17 kelas yang diimplementasikan. Kelas yang tidak diimplementasikan ada 4 kelas yaitu VoiceListener, KinectListener, KinectListenerImpl dan AppStateAbs. Fungsi kelas VoiceListener telah digantikan oleh kelas yang ada pada *library* CloudGarden's JSAPI. Fungsi interface KinectListener dan KinectListenerImpl telah digantikan oleh kelas yang ada pada *library* Kinect. Kelas AppStateAbs telah disediakan oleh *game engine* jMonkey. Tabel 5.3 menjelaskan mengenai pasangan antara klas dengan *file* program yang digunakan untuk mengimplentasikannya.

Tabel 5.3. Implementasi Kelas Pada Kode Program *.java

No	Nama Paket	Nama Kelas	Nama *File* Program
1	MyGame	InGame	InGameKin.java
2	Entity	Player	Player.java
3	Generator	SoalGenerator	SoalGenerator.java
4		HighScoreGenerator	HighScoreGen.java
5		GeneratorAllItem	GeneratorAllItem.java
6		RandomerCharNBom	RandomerCharNBom.java
7	Nifty.Controller	mStartScreen	mStartScreen.java
8		mMenuScreen	mMenuScreen.java
9		mHighScoreScreen	mHighScoreScreen.java
10		mLevelScreen	mLevelScreen.java
11		mDeviceScreen	mDeviceScreen.java
12		mLoadingScreen	mLoadingScreen.java
13		mHowToPlayScreen	mHowToPlayScreen.java
14		mHudScreen	mHudScreen.java
15		mPauseScreen	mPauseScreen.java
16		mUnameScreen	mUnameScreen.java
17		mCreditScreen	mCreditScreen.java

5.4. Implementasi Algoritma

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

5.4.1. Algoritma *Generate Random Item*

Generate random item merupakan proses yang akan merandom item yang akan muncul dalam game. Item yang muncul dapat berupa huruf acak konsonan atau vocal, huruf jawaban soal, atau item bom. Implementasi algoritma *generate random item* ditunjukkan pada tabel 5.4.

Tabel 5.4. Implementasi Algoritma *Generate Random Item*

Pseudocode *Generate random item*

```
Methode: getRandomItem PARAMETER assetManager, posX, posY,curKar IS
Spatial

DECLARATION:
     TYPE assetManager IS AssetManager
     posX IS float
     posY IS float
     curKar IS char
     chooseTipe IS ItemTipe
     chooseKar IS char
     item is Spatial

DESCRIPTION:
  1    chooseTipe <- CALL randomItemTipe()
  2    IF(chooseTipe == konsonanItem) THEN
  3      chooseKar <- CALL getRandomKonsonanKar()
  4      Item <- CALL buildItemKar(assetManager,posX,
  5      posY,chooseKar)
  6      RETURN item
  7    ELSE IF (chooseTipe == vocalItem) THEN
  8      chooseKar <- CALL getRandomVocalKar()
  9      Item <- CALL buildItemKar(assetManager,posX,
 10      posY,chooseKar)
```

```
11      RETURN item
12    ELSE IF (chooseTipe == currentAnswerItem) THEN
13      chooseKar <- curKar
14      Item <- CALL buildItemKar(assetManager,posX,
15      posY,chooseKar)
16      RETURN item
17    ELSE
18       Item <- CALL loadModel("bomModel")
19       item.name <- "Bom"
20       item.posX <- posX
21       item.posY <- posY
22       RETURN item
23    END IF
24    END getRandomItem
```

Sumber: Implementasi

Penjelasan dari algoritma *generate random item* untuk getRandomItem() pada tabel 5.4., yaitu

1. Baris 1 menjelaskan pemberian nilai kepada variable chooseTipe yang didapat dari fungsi randomItemTipe(). Fungsi variable ini adalah untuk menampung pilihan item yang harus diproduksi yang didapat secara random yang akan menentukan item yang akan diproduksi.
2. Baris 2 - 6 menjelaskan kondisi ketika item yang harus diproduksi adalah item konsonan, maka akan diproduksi item huruf konsonan secara acak dan memberikan kembalian item bertipe spatial yang telah terinisialisi huruf konsonan.
3. Baris 7 - 11 menjelaskan kondisi ketika item yang harus diproduksi adalah item vocal, maka akan diproduksi item huruf vocal secara acak dan memberikan kembalian item bertipe spatial yang telah terinisialisi huruf vocal.
4. Baris 12 - 16 menjelaskan kondisi ketika item yang harus diproduksi adalah item huruf yang dibutuhkan untuk pembentukan jawaban, maka akan diproduksi item huruf yang dibutuhkan untuk pembentukan jawaban

dan memberikan kembalian item bertipe spatial yang telah terinisialisi huruf yang dibutuhkan untuk pembentukan jawaban.

5. Baris 17 - 22 menjelaskan kondisi ketika item yang harus diproduksi adalah item bom, maka akan diproduksi item bom dan memberikan kembalian item bertipe spatial yang telah terinisialisi model bom.
6. Baris 23 merupakan akhir dari pengecekan kondisi

5.4.2. Algoritma Membuat Item Karakter

Implementasi algoritma membuat item karakter ditunjukkan pada tabel 5.5.

Tabel 5.5. Membuat Item Karakter

Pseudocode Membuat Item Karakter

```
Methode : buildItemKar PARAMETER assetManager, posX, posY, Kar IS
Spatial

DECLARATION :
     TYPE assetManager IS AssetManager
     posX IS float
     posY IS float
     Kar IS String
     BoxKar IS Box
     Item IS Spatial
     Mat IS Material
     Tex IS Texture

DESCRIPTION:
  1  BoxKar <- CREATE OBJEK Box()
  2  Item <- CREATE OBJEK Geometry(Kar,BoxKar)
  3  Mat <- CREATE OBJEK Material()
  4  Tex <- CALL loadTexture(Kar)
  5  CALL Mat.setTexture(Tex)
  6  CALL Item.setMaterial(Mat)
  7  CALL Item.setLocalTranslation(posX,posy,0)
  8  RETURN Item
  9  END buildItemKar
```

Sumber : Implementasi

Penjelasan dari algoritma membuat item karakter untuk buildItemKar pada tabel 5.5., yaitu

1. Baris 1 menjelaskan instansiasi objek BoxKar.
2. Baris 2 menjelaskan instansiasi objek Item.
3. Baris 3 menjelaskan instansiasi objek Mat.
4. Baris 4 menjelaskan loading texture dimana texture diambil dari file gambar dengan nama yang sesuai dengan karakter yang akan dibuat.
5. Baris 5 menjelaskan pemberian tekstur kepada objek material.
6. Baris 6 menjelaskan pemberian material kepada objek item.
7. Baris 7 menjelaskan pengaturan translasi objek item.
8. Baris 8 menjelaskan pengembalian nilai operasi buildItemKar berupa objek item yang telah diinisialisasi.

5.4.3. Algoritma Inisialisi Soal

Inisialisasi soal merupakan proses inisialisasi awal soal yang terdiri dari proses pengambilan file soal dan pemilihan soal awal yang muncul dalam *game*. Implementasi algoritma inisialisasi soal ditunjukkan pada tabel 5.6.

Tabel 5.6. Implementasi Algoritma Inisialisasi Soal

Pseudocode Inisialisasi Soal

```
METHODE: initSoal PARAMETER pathSoal IS boolean

DECLARATION:
    TYPE pathSoal IS String
    curSoal IS String
DESCRIPTION:
 1   curSoal <- ""
 2   IF(CALL setSoal(pathSoal) > 0) THEN
 3     curSoal <- CALL  getRandomSoal()
 4     RETURN true
 5   END IF
 6   RETURN false
 7   END initSoal
```

Sumber: Implementasi

Penjelasan dari algoritma inisialisasi soal untuk initSoal() pada tabel 5.6., yaitu

1. Baris 1 menjelaskan inisialisasi nilai awal untuk curSoal berisi String kosong.
2. Baris 2 – 4 menjelaskan kondisi ketika fungsi setSoal bernilai true, yang berarti file soal ditemukan dan file tersebut berisi list soal dengan jumlah lebih dari 0. Dalam kondisi tersebut curSoal akan diinisialisasi soat pertama secara random setelah itu method initSoal() akan mengembalikan nilai true
3. Baris 5 – 6 mengembalikan nilai false ketika soal tidak berhasil diinisialisasi

5.4.4. Algoritma *Setting* Soal

Algoritma *setting* soal merupakan algoritma untuk mengambil soal dari file. Implementasi algoritma *setting* soal ditunjukkan pada tabel 5.7.

Tabel 5.7. Implementasi Algoritma *Setting* Soal

Pseudocode *Setting* Soal

```
Methode : setSoal PARAMETER path IS int

listSoal IS StringList  //atribut kelas

DECLARATION:
    TYPE path IS String
    fileSoal IS File
    inputStream IS BufferedReader
    line IS String

DESCRIPTION:
  1  fileSoal <- CREATE OBJEK File (path)
  2  listSoal <- CREATE OBJEK StringList
  3  TRY
  4    inputStream <- CREATE OBJEK BufferedReader (fileSoal)
  5    WHILE ((line <- CALL inputStream.readLine)!=null) DO
  6     CALL listSoal.add(line)
  7    END WHILE
```

```
 8     CALL inputStream.close()
 9     RETURN listSoal.size
10   CATCH (IOException e)
11     CALL PrintError()
12     RETURN 0
13   END TRY
14   END setSoal
```

Sumber: Implementasi

Penjelasan dari algoritma *setting* soal untuk setSoal pada tabel 5.7., yaitu

1. Baris 1 – 2 menjelaskan instansiasi objek variabel fileSoal dan listSoal.
2. Baris 3 – 9 menjelaskan kondisi jika tidak ada kesalahan atau *exception* dalam pembacaan file. Proses yang dilakukan adalah menyimpan soal - soal kedalam objek listSoal. Proses diakhiri dengan mengembalikan nilai yang berisi jumlah soal.
3. Baris 10 – 12 merupakan kondisi jika ada kesalahan atau *exception* dalam pembacaan file. Proses yang dilakukan adalah menampilkan pesan kesalahan dan mengembalikan nilai 0.

5.4.5. Algoritma *Random Soal*

Algoritma random soal merupakan algoritma untuk memilih soal dari list soal secara random, lalu menghapusnya dari list soal agar soal tersebut tidak muncul kembali. Implementasi algoritma random soal ditunjukkan pada tabel 5.8.

Tabel 5.8. Implementasi Algoritma Random Soal

Pseudocode Random Soal

```
Methode : getRandomSoal IS String

//atribut kelas
listSoal IS StringList
curSoal IS String

DECLARATION:
    randomINum IS int

DESCRIPTION:
```

```
1  IF(listSoal.size<1)THEN
2    curSoal=”END”
3    RETURN curSoal
4  ELSE
5    randomNum <- CALL random(listSoal.size)
6    curSoal <- CALL listSoal.get(randomNumber)
7    CALL listSoal.remove(curSoal)
8    RETURN curSoal
9  END IF
10 END getRandomSoal
```

Sumber: Implementasi

Penjelasan dari algoritma random soal untuk getRandomSoal pada tabel 5.8., yaitu

1. Baris 1 – 3 menjelaskan kondisi jika objek listSoal sudah tidak berisi soal maka kembalian dari operasi getRandomSoal adalah string yang berisi kalimat “END”.
2. Baris 4 – 8 menjelaskan kondisi jika listSoal berisi minimal satu soal maka kembalian dari operasi getRandomSoal adalah string soal yang telah diacak dari listSoal.
3. Baris 9 – 10 merupakan akhir dari operasi getRandomSoal.

5.4.6. Algoritma *Input High Score*

Input high score merupakan proses memasukan nilai yang didapat *player* ke dalam data lima nilai tertinggi. Nilai *player* akan diperiksa terlebih dahulu untuk masuk ke dalam lima nilai tertinggi. Nilai player yang masuk nilai tertinggi akan menggeser nilai dibawahnya dan menghilangkan nilai terendah di dalam data lima nilai tertinggi. Implementasi algoritma *input high score* ditunjukkan pada tabel 5.9.

Tabel 5.9. Algoritma *Input High Score*

Pseudocode *Input High Score*

```
METHODE: inputHighScore PARAMETER nameNscore, highScorePath IS boolean

DECLARATION:
     TYPE nameNscore IS String
     highScorePath IS String
     listFiveScore IS StringList
     nameNscoreOld IS String

DESCRIPTION:
 1   listFiveScore <- CALL bacaHighScore()
 2   IF(listFiveScore.size <=0) THEN
 3     CALL listFiveScore.add(nameNscore)
 4     CALL tulisFileHighScore(highSCorePath,listFiveScore)
 5     RETURN true
 6   ELSE IF ( listScore.size < 5 AND listScore.size >0 ) THEN
 7     CALL listFiveScore.add(nameNscore)
 8     CALL sortingScore(listFiveScore)
 9     CALL tulisFileHighScore(highSCorePath,listFiveScore)
10     RETURN true
11   ELSE
12     IF( CALL isTopFive(listFiveScore, nameNscore)) THEN
13        tulisFileHighScore(highSCorePath,listFiveScore)
14        RETURN true
15     ELSE
16        RETURN false
17     END IF
18   END IF
19   END inputHighScore
```

Sumber: Implementasi

Penjelasan dari algoritma *input high score* untuk inputHighScore() pada tabel 5.9., yaitu

1. Baris 1 menjelaskan inisialisasi variabel listFiveScore dengan nilai yang didapat dari fungsi bacaHighScore. listFiveScore menampumg data nilai tertinggi dalam *game* Slash Word.

2. Baris 2 – 5 merupakan kondisi jika listFiveScore tidak berisi data lima nilai tertinggi atau daftar lima nilai tertinggi masih kosong, maka yang dilakukan adalah memasukan skor pemain ke dalam daftar lima nilai tertinggi.
3. Baris 6 – 10 merupakan kondisi jika listFiveScore telah berisi data lima nilai tertinggi tetapi kurang dari lima orang, maka yang dilakukan adalah memasukan skor pemain ke dalam daftar lima nilai tertinggi yang sebelumnya dilakukan sorting data terlebih dahulu agar data urut. Operasi ini menghasilkan nilai kembalian true.
4. Baris 11 – 17 merupakan kondisi jika listFiveScore telah berisi data lima nilai tertinggi dan telah berisi lima data, maka yang dilakukan adalah memeriksa data skor pemain yang baru untuk dimasukan ke dalam daftar lima nilai tertinggi. Baris 12 – 14 merupakan kondisi jika nilai pemain masuk dalam daftar lima nilai tertinggi, proses yang dilakukan adalah memasukan skor pemain terbaru kedalam daftar lima nilai tertinggi, menggeser nilai yang ada dibawahnya dan mengembalikan nilai true. Baris 15 – 16 merupakan kondisi jika nilai pemain tidak masuk dalam daftar lima nilai tertinggi, proses yang dilakukan adalah mengembalikan nilai false.

5.4.7. Algoritma Baca *High Score*

Algoritma baca *high score* merupakan algoritma untuk mengambil data *high score* dari file. Implementasi algoritma baca *high score* ditunjukkan pada tabel 5.10.

Tabel 5.10. Implementasi Algoritma Baca *High Score*

Pseudocode Baca *High Score*

`Methode : bacaHighScore PARAMETER path IS StringList`
 	
`listFiveScore IS StringList`
 	
`DECLARATION :`
`  TYPE path IS String`
`  highFiveFile IS File`

```
    inputStream IS BufferedReader
    line IS String

DESCRIPTION:
  1  highFiveFile <- CREATE OBJEK File (path)
  2  listFiveScore <- CREATE OBJEK StringList
  3  TRY
  4    inputStream <- CREATE OBJEK BufferedReader (highFiveFile)
  5    WHILE ((line <- CALL inputStream.readLine)!=null) DO
  6       CALL listFiveScore.add(line)
  7    END WHILE
  8    CALL inputStream.close()
  9  CATCH (IOException e)
 10    CALL PrintError()
 11  END TRY
 12  RETURN listFiveScore
 13  END bacaHighScore
```

Sumber: Implementasi

Penjelasan dari algoritma baca *high score* untuk bacaHighScore pada tabel 5.10., yaitu

1. Baris 1 – 2 menjelaskan instansiasi objek variabel highFiveFile dan listFiveScore.
2. Baris 3 – 8 menjelaskan kondisi jika tidak ada kesalahan atau *exception* dalam pembacaan file. Proses yang dilakukan adalah menyimpan data lima nilai tertinggi kedalam objek listFiveScore.
3. Baris 9 – 10 merupakan kondisi jika ada kesalahan atau *exception* dalam pembacaan file. Proses yang dilakukan adalah menampilkan pesan kesalahan.
4. Baris 11 – 13 merupakan proses mengembalikan objek listFiveScore.

5.4.8. Algoritma Tulis *High Score*

Algoritma tulis *high score* merupakan algoritma untuk menulis *high score* ke dalam file. Implementasi algoritma tulis *high score* ditunjukkan pada tabel 5.11.

Tabel 5.11. Implementasi Algoritma Tulis *High Score*

Pseudocode Tulis *High Score*

```
Methode : tulisFileHighScore PARAMETER path, listFiveScore IS boolean

DECLARATION :
    TYPE path IS String
    listFiveScore IS StringList
    highFiveFile IS File
    writer IS BufferedWriter
    Status IS boolean

DESCRIPTION:
 1  highFiveFile <- CREATE OBJEK File (path)
 2  Status <- false
 3  TRY
 4    writer <- CREATE OBJEK BufferedWriter (highFiveFile)
 5    CALL writer.write(listFiveScore.get(0))
 6    FOR (iterasi = 1 TO < listFiveScore.size STEP 1)
 7      CALL writer.newLine
 8      CALL writer.write(listFiveScore.get(iterasi))
 9    END FOR
10    Status <- true
11    CALL writer.close()
12  CATCH (IOException e)
13    CALL PrintError()
14    Status <- false
15  END TRY
16  RETURN Status
17  END tulisFileHighScore
```

Sumber: Implementasi

Penjelasan dari algoritma tulis *high score* untuk tulisFileHighScore pada tabel 5.11., yaitu

1. Baris 1 – 2 menjelaskan instansiasi objek variabel highFiveFile dan pemberian nilai false pada variabel Status.
2. Baris 3 – 11 menjelaskan kondisi jika tidak ada kesalahan atau *exception* dalam penulisan file. Proses yang dilakukan adalah menulis data dari objek listFiveScore ke dalam file dan memberikan nilai true pada Status.

3. Baris 12 – 14 merupakan kondisi jika ada kesalahan atau *exception* dalam penulisan file. Proses yang dilakukan adalah menampilkan pesan kesalahan dan memberikan nilai false pada status.
4. Baris 15 – 17 merupakan proses mengembalikan nilai Status dari operasi tulisFileHighScore.

5.4.9. Algoritma *Sorting Score*

Algoritma *sorting score* merupakan algoritma untuk mengurutkan data skor. Implementasi algoritma *sorting score* ditunjukkan pada tabel 5.12.

Tabel 5.12. Implementasi Algoritma *Sorting Score*

Pseudocode *Sorting Score*

```
Methode : sortingScore PARAMETER listFiveScore IS StringList
DECLARATION :
     TYPE listFiveScore IS StringList
     Nilai1 IS int
     Nilai2 IS int
     Temp IS String

DESCRIPTION:
  1  FOR (iterasi1 = 0 TO < listFiveScore.size-1 STEP 1)
  2    FOR (iterasi2 = iterasi1+1 TO < listFiveScore.size STEP 1)
  3      Nilai1 <- CALL toInteger(listFiveScore.get(iterasi2).
  4      score)
  5      Nilai2 <- CALL toInteger(listFiveScore.get(iterasi1).
  6      score)
  7      IF(Nilai1>Nilai2) THEN
  8        Temp <- CALL listFiveScore.get(iterasi1)
  9        CALL listFiveScore.set(iterasi1,listFiveScore.
 10        get(iterasi2))
 11        CALL listFiveScore.set(iterasi2, Temp)
 12      END IF
 13    END FOR
 14  END FOR
 15  RETURN listFiveScore
 16  END sortingScore
```

Sumber: Implementasi

Penjelasan dari algoritma *sorting score* untuk sortingScore pada tabel 5.12., yaitu

1. Baris 1 – 14 menjelaskan proses *sorting* nilai di dalam objek listFiveScore. Pada baris 7 – 11 menjelaskan kondisi jika ada nilai yang belum urut maka akan dilakukan tukar posisi.
2. Baris 15 merupakan proses pengembalian objek listFiveScore.

5.4.10. Algoritma *Score Filter*

Algoritma *score filter* merupakan algoritma untuk menyaring skor pemain untuk masuk nilai tertinggi. Implementasi algoritma *score filter* ditunjukkan pada tabel 5.13.

Tabel 5.13. Implementasi Algoritma *Score Filter*

Pseudocode *Score Filter*

```
Methode : isTopFive PARAMETER listFiveScore, nameNscore IS boolean

DECLARATION :
     TYPE listFiveScore IS StringList
     nameNscore IS String
     nameNscoreOld IS String

DESCRIPTION:
  1  FOR (iterasi = 0 TO < 5 STEP 1)
  2    IF(CALL toInteger(listFiveScore.get(iterasi).score)
  3    < CALL toInteger(nameNscore.score)) THEN
  4      nameNscoreOld <- CALL listFiveScore.get(iterasi)
  5      CALL listFiveScore.set(iterasi, NameNscore)
  6      CALL geser(listFiveScore,nameNscoreOld, iterasi)
  7      RETURN true
  8     END IF
  9  END FOR
 10  RETURN false
 11  END isTopFive
```

Sumber: Implementasi

Penjelasan dari algoritma *score filter* untuk isTopFive pada tabel 5.13., yaitu

1. Baris 1 – 9 menjelaskan proses filter skor pemain terhadap nilai yang terdapat pada listFiveScore. Pada baris 2 – 7 menjelaskan kondisi jika nilai pemain masuk dalam daftar lima nilai tertinggi, maka proses yang dilakukan adalah memasukan skor pemain terbaru kedalam objek listFiveScore, menggeser nilai yang ada dibawahnya dan mengembalikan nilai true.
2. Baris 10 merupakan proses pengembalian nilai false yang berarti skor pemain tidak masuk lima nilai tertinggi.

5.4.11. Algoritma Geser Skor

Algoritma geser skor merupakan algoritma untuk menggeser nilai yang lebih rendah dari nilai *input* yang baru di dalam daftar lima nilai tertinggi dan menghilangkan nilai terbawah yang paling rendah sehingga tetap hanya ada lima nilai tertinggi. Implementasi algoritma geser skor ditunjukkan pada tabel 5.14.

Tabel 5.14. Implementasi Algoritma Geser Score

Pseudocode Geser Skor

```
Methode : geser PARAMETER listFiveScore,nameNscore, pos
DECLARATION :
    TYPE listFiveScore IS StringList
    pos IS int
    curPos IS int
    nameNscore IS String
    curNameNscore IS String

DESCRIPTION:
  1  IF ( pos < 4) THEN
  2    curPos <- pos+1
  3    curNameNscore <- CALL listFiveScore.get(curPos)
  4    CALL listFiveScore.set (curPos,nameNscore)
  5    CALL geser(listFiveScore,curNameNscore,curPos)
  6  END IF
  7  END geser
```

Sumber: Implementasi

Penjelasan dari algoritma geser skor untuk geser pada tabel 5.14., yaitu

1. Baris 1 – 5 menjelaskan kondisi jika posisi nilai yang akan digeser adalah kurang dari 4 maka akan dilakukan proses pergeseran nilai dengan nilai baru menggeser nilai lama yang lebih kecil. Terdapat lima nilai dengan posisi awal nilai adalah 0 dan posisi akhir adalah 4. Jika posisi nilai baru ada di posisi 4 maka tidak akan terjadi pergeseran nilai, nilai baru akan langsung menghilangkan nilai yang lama.
2. Baris 7 merupakan akhir dari operasi geser

5.4.12. Algoritma *On Hand Move*

On hand move merupakan proses yang akan terus dijalankan pada *in game state* ketika posisi telapak tangan pemain berubah. Implementasi algoritma *on hand move* pada operasi `onPointUpdateEvent()` merupakan proses override dari operasi di kelas KinectListenerImpl. Kelas KinectListenerImpl merupakan kelas yang berfungsi sebagai *listener* input dari sensor Kinect. Overriding operasi ini dilakukan pada kelas MyKinectListener yang merupakan kelas *inner* di dalam kelas InGame. Kelas MyKinectListener selanjutnya didaftarkan sebagai input listener di dalam objek input manager. Objek Input manager pada *game engine* jMonkey berfungsi untuk mengatur dan memetakan input – input pada *game*. Implementasi algoritma *on hand move* ditunjukkan pada tabel 5.15.

Tabel 5.15. Implementasi Algoritma *On Hand Move*

Pseudocode *On Hand Move*
METHODE: onPointUpdateEvent PARAMETER name, evt, fps DECLARATION: TYPE evt IS HandEvent name IS String fps IS float posX IS float posY IS float DESCRIPTION:

```
1   posX <- CALL evt.getHand().getPosition().getX()
2   posy <- CALL evt.getHand().getPosition().getY()
3   IF (posX > batasPositiveX) THEN
4      PosX <- batasPositiveX
5      IF (posY > batasPositiveY) THEN
6        PosY <- batasPositiveY
7      ELSE IF (posY < batasNegativeY)
8        PosY <- batasNegativeY
9      END IF
10  ELSE IF (posX < batasNegativeX)
11     PosX <- batasNegativeX
12     IF (posY > batasPositiveY) THEN
13       PosY <- batasPositiveY
14     ELSE IF (posY < batasNegativeY)
15       PosY <- batasNegativeY
16     END IF
17  ELSE
18     IF (posY > batasPositiveY) THEN
19       PosY <- batasPositiveY
20     ELSE IF (posY < batasNegativeY)
21       PosY <- batasNegativeY
22     END IF
23  END IF
24  CALL moveKursor(posX,posy,0)
25  END onPointUpdateEvent
```

Sumber: Implementasi

Penjelasan dari algoritma *on hand move* untuk onPointUpdateEvent pada tabel 5.15., yaitu

1. Baris 1 – 2 menjelaskan inisialisasi variabel posX dan posY dengan nilai yang didapat dari posisi koordinat x dan y tangan.
2. Baris 3 – 23 merupakan proses untuk memfilter posisi x dan y yang dihasilkan dari pergerakan tangan agar kursor yang digerakkan tidak melebihi batas layar
3. Baris 24 merupakan proses untuk menggerakkan kursor tangan sesuai arah pergerakan telapak tangan dalam koordinat x dan y yang telah difilter.

5.4.13. Algoritma Gerakan Kursor Pemain

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 5.16.

Tabel 5.16. Implementasi Algoritma Gerakan Kursor Pemain

Pseudocode Gerakan Kursor Pemain

```
METHODE: moveKursor PARAMETER posX, posy

kursorItem IS Spatial

DECLARATION:
     TYPE posX IS float
     posY IS float
DESCRIPTION:
 1   CALL kursorItem.setLocalTranslation(posX,posY,0)
 2   END moveKursor
```

Sumber: Implementasi

Penjelasan dari algoritma gerakan kursor pemain untuk moveKursor pada tabel 5.16., yaitu

1. Baris 1 menjelaskan pengaturan translasi kursor berdasarkan posisi koordinat x dan y yang diberikan pada parameter masukan.

5.5. Implementasi *Game Elements*

Karakter atau elemen-elemen yang dibutuhkan pada *game* Slash Word yang telah dijelaskan di *game design document* pada BAB Perancangan. Implementasi dari elemen permainan ditunjukkan pada tabel 5.17.

Tabel 5.17. Implementasi Karakter Game Slash Word

Game Object	**Profil**
	**Nama :** Gambar soal **Keterangan :** Beberapa gambar soal yang terdiri dari gambar buah – buahan , binatang dan alat transportasi yang harus dijawab menggunakan bahasa inggris
	**Nama :** Kotak huruf **Keterangan :** Kotak huruf inilah yang akan membentuk jawaban soal. Kotak huruf yang tidak tepat digunakan dalam pembentukan jawaban soal harus dihilangkan menggunakan kursor tangan pemain. Terdapat 26 kotak huruf yaitu kotak huruf A sampai dengan kotak huruf Z
	**Nama :** *Hand Cursor* **Keterangan :** *Hand Cursor* ini yang akan digerakan oleh gerakan tangan pemain untuk menghilangkan huruf - huruf yang tidak tepat digunakan dalam pembentukan jawaban soal

Sumber: Implementasi

5.6. Implementasi Antarmuka dan HUD

Implementasi antarmuka *game* Slash Word ini terdiri dari *main menu, game level setting menu, high score menu, device setting menu, loading screen,*

how to play screen, pause menu, end game screen, game arena kelas, game arena perpustakaan dan HUD.

5.6.1. *Main Menu*

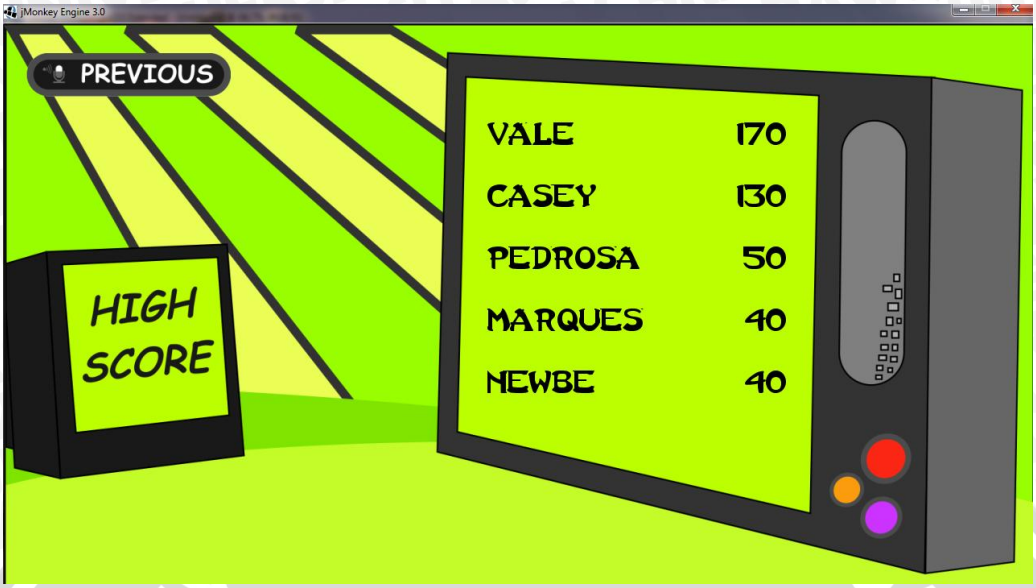
Main menu merupakan halaman menu yang akan tampil pertama kali ketika game dijalankan. Di halaman main menu, *button* atau petunjuk *voice command* yang bisa dipilih untuk menuju menu selanjutnya yaitu :

- ***High Score***, perintah suara ini digunakan untuk masuk ke menu high score. Terdapat juga button *high score* pada menu utama.
- ***Let's Play,*** perintah suara ini digunakan untuk masuk ke menu pemilihan level jika diucapkan pada menu utama. Terdapat juga button *let's play* pada menu utama.
- ***Exit,*** perintah suara ini digunakan untuk keluar dari *game* Slash Word. Terdapat juga button *let's play* pada menu utama.
- **Credit,** perintah suara ini digunakan untuk masuk ke *credit screen*. Terdapat juga button *credit* pada menu utama yaitu pada gambar logo *game studio* di pojok kanan bawah.

Implementasi *main menu* ditunjukkan oleh gambar 5.1.

Gambar 5.1. Implementasi Halaman *Main Menu*

Sumber : Implementasi



Gambar 5.2. Implementasi Halaman *High Score Menu*

Sumber: Implementasi

5.6.2. *High Score Menu*

Halaman *high score menu* merupakan menu yang menampilkan daftar lima pemain dengan skor tertinggi. Perintah suara "*previous*" digunakan untuk kembali ke menu utama. Terdapat juga *button previous* untuk kembali ke menu utama. Tampilan halaman *high score menu* ditunjukkan pada gambar 5.2.

Gambar 5.3. Implementasi Halaman *Game Level Setting Menu*

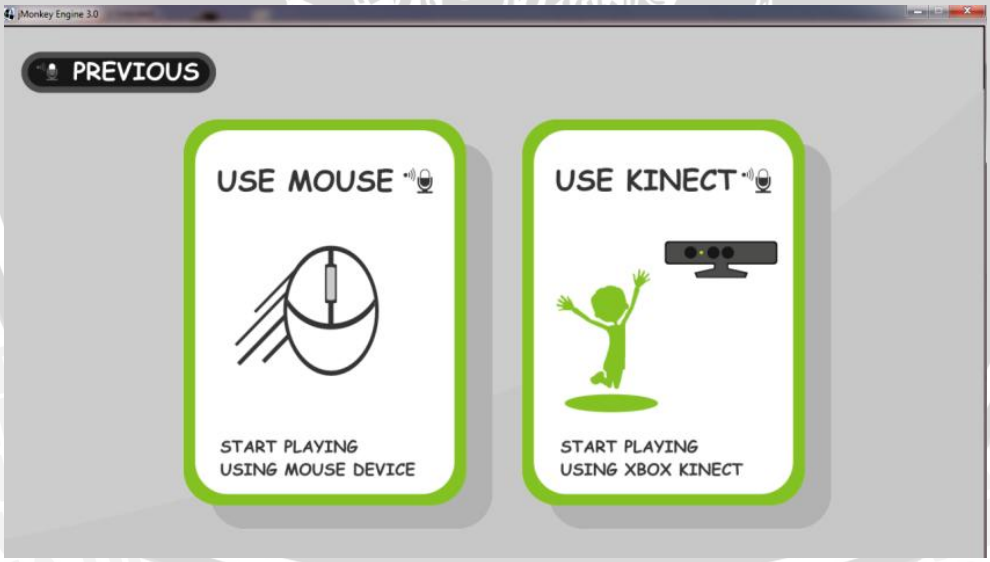
Sumber: Implementasi

5.6.3. Halaman *Game Level Setting Menu*

Halaman *game level setting menu* adalah halaman untuk memilih *game level.* Perintah suara "*easy*", "*medium*" dan "*hard*" dapat digunakan untuk memilih level permainan. Perintah suara "*previous"* digunakan untuk kembali ke halaman sebelumnya. Terdapat juga *button* untuk memilih level permainan dan untuk kembali ke halaman sebelumnya. Tampilan halaman *game level setting* ditunjukkan pada gambar 5.3.

5.6.4. Halaman *Device Setting Menu*

Halaman *device setting menu* adalah halaman untuk memilih kendali permainan. Perintah suara "*use mouse*" dan "*use kinect*" dapat digunakan untuk memilih kendali permainan. Perintah suara "*previous*" digunakan untuk kembali ke halaman sebelumnya. Terdapat juga *button* untuk memilih kendali permainan dan untuk kembali ke halaman sebelumnya. Tampilan halaman *device setting* ditunjukkan pada gambar 5.4.

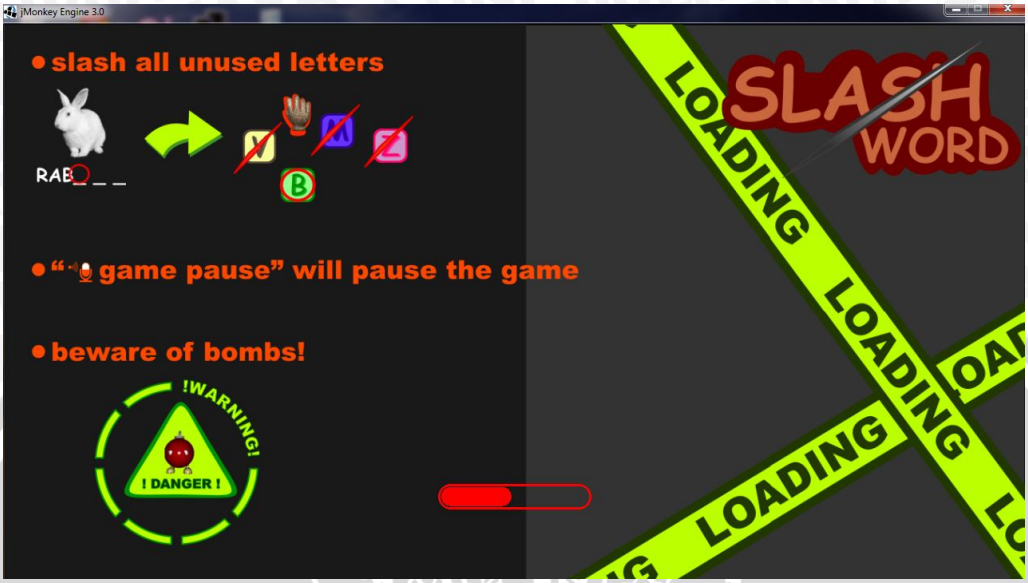


Gambar 5.4. Implementasi Halaman *Device Setting Menu*

Sumber: Implementasi

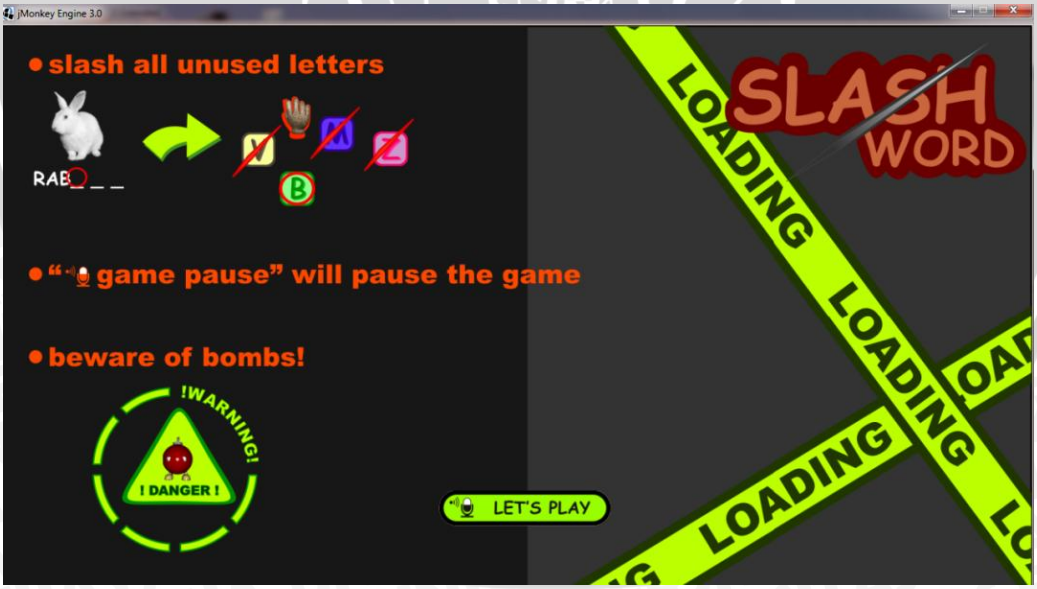
5.6.5. Halaman *Loading Screen*

Tampilan halaman *loading screen* ditunjukkan pada Gambar 5.5.



Gambar 5.5. Tampilan Halaman *Loading Screen*

Sumber : Implementasi



Gambar 5.6. Tampilan Halaman *How To Play*

Sumber : Implementasi

5.6.6. Halaman *How To Play*

Tampilan halaman *How To Play* yang memberikan informasi cara bermain ditunjukkan pada gambar 5.6. Perintah suara "*let's play*" digunakan untuk mulai ke permainan.

5.6.7. Halaman *Pause Menu*

Halaman *pause menu* merupakan halaman yang ditampilkan ketika *game* pada *state pause*. Pada *pause menu* untuk kembali melanjutkan permainan dapat dengan menggunakan perintah suara "*resume*" atau dengan menekan *button resume*. Perintah suara "*menu*" dan *button menu* digunakan untuk kembali ke menu utama. Tampilan halaman *pause menu* ditunjukkan pada gambar 5.7.

Gambar 5.7. Tampilan Halaman *Pause Menu*

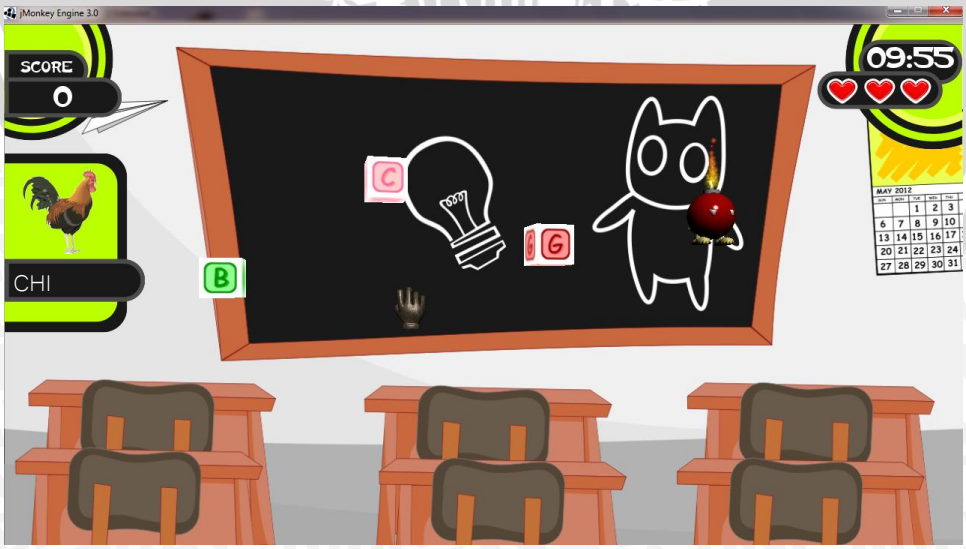
Sumber : Implementasi

5.6.8. Halaman *End Game Screen*

Halaman *end game screen* merupakan halaman yang ditampilkan ketika permainan berakhir. Pada *end game screen* terdapat keterangan skor permainan dan *text field* untuk mengisi nama pemain. Perintah suara "*submit*" dan *button submit* digunakan untuk kembali ke menu utama. Tampilan halaman *end game screen* ditunjukkan pada gambar 5.8.

Gambar 5.8. Tampilan Halaman *End Game Screen*

Sumber : Implementasi



Gambar 5.9. Tampilan HUD *screen*

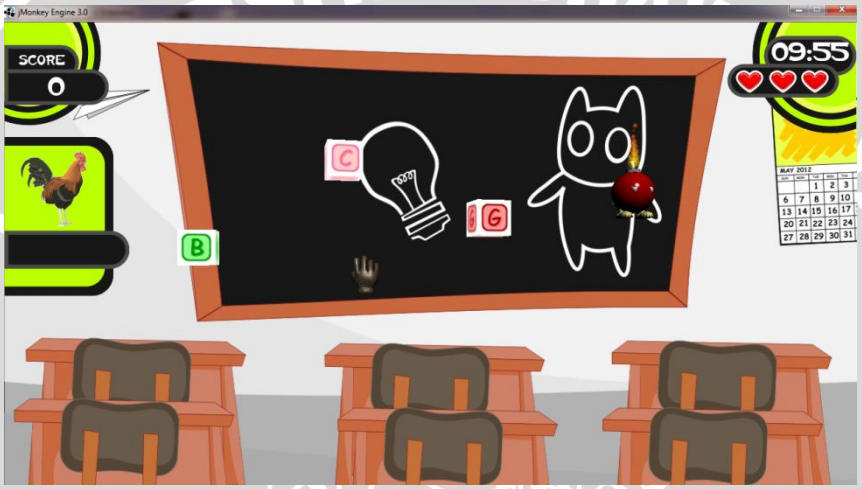
Sumber: Implementasi

5.6.9. Implementasi HUD *(Heads-Up Display)*

HUD berisi informasi mengenai waktu, *life*, soal dan jawaban pemain. implementasi HUD ditunjukan pada gambar 5.9.

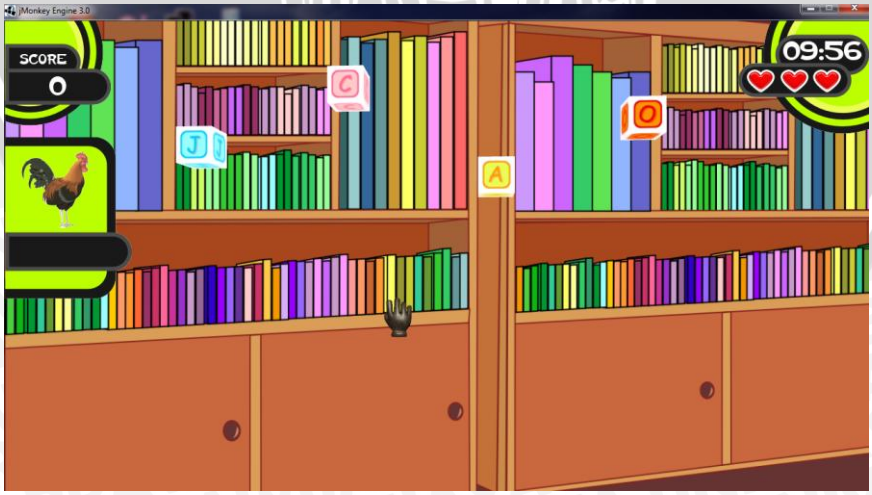
5.6.10. Game Arena

Arena permainan Slash Word terdiri dari ruang kelas dan ruang perpustakaan. *Game* arena ruangan kelas ditunjukan oleh gambar 5.10. *Game* arena ruangan perpustakaan ditunjukan oleh gambar 5.11.



Gambar 5.10. *Game Arena* Ruangan Kelas

Sumber: Implementasi



Gambar 5.11. *Game Arena* Ruangan Perpustakaan

Sumber: Implementasi

BAB VI
PENGUJIAN DAN ANALISIS

Bab ini membahas mengenai tahapan pengujian dan analisis *game* Slash Word yang telah dibangun. Proses pengujian dilakukan melalui empat tahapan yaitu pengujian unit, pengujian integrasi, pengujian validasi dan pengujian kinerja. Pada pengujian unit dan integrasi, akan digunakan teknik pengujian *White Box* (*White Box Testing*). Pada pengujian validasi akan digunakan teknik pengujian *Black Box* (*Black Box Testing*). Pada pengujian kinerja akan digunakan analisis FPS (*frame per second)* untuk mengetahui kinerja *game* Slash Word.

6.1. Pengujian

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

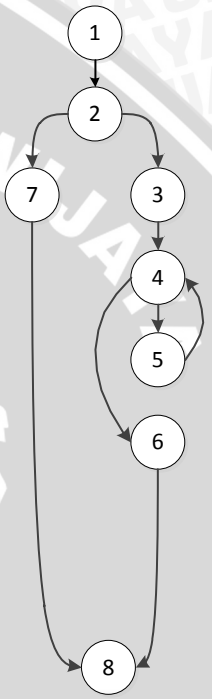
6.1.1. Pengujian Unit

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 *flow graph*, menentukan *cyclometic complexcity* dan melakukan uji kasus untuk setiap path yang ada.

6.1.1.1. Pengujian Unit Operasi `setSoal()`

Operasi `setSoal()` merupakan implementasi algoritma *setting* soal untuk mengambil soal dari file. Operasi ini terdapat dalam kelas SoalGenerator. Pemodelan operasi `setSoal()` dalam bentuk *flow graph* ditunjukkan pada tabel 6.1.

Tabel 6.1. Pemodelan *Flow Graph* Algoritma `setSoal()`

Pseudocode	*Flow Graph*
Methode : setSoal PARAMETER path IS int listSoal IS StringList //atribut kelas DECLARATION: TYPE path IS String fileSoal IS File inputStream IS BufferedReader line IS String DESCRIPTION : 1 fileSoal <- CREATE OBJEK File (path) 2 listSoal <- CREATE OBJEK StringList 3 TRY 4 inputStream <- CREATE OBJEK BufferedReader 5 (fileSoal) 6 WHILE ((line <- CALL 7 inputStream.readLine)!=null) DO 8 CALL listSoal.add(line) 9 END WHILE 10 CALL inputStream.close() 11 RETURN listSoal.size 12 CATCH (IOException e) 13 CALL PrintError() 14 RETURN 0 15 END TRY 16 END setSoal	Node (N)= 8 Edge (E)= 9 

Sumber: Pengujian dan Analisis

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

V(G) = E – N + 2

= 9 – 8 + 2 = 3

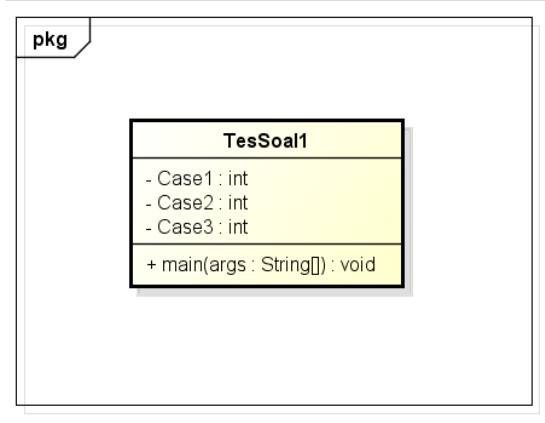
Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan tiga buah basis set dari jalur independent yaitu:

Jalur 1: 1 – 2 – 7 – 8

Jalur 2: 1 – 2 – 3 – 4 – 6 – 8

Jalur 3: 1 – 2 – 3 – 4 – 5 – 4 …..

Proses pengujian operasi `setSoal()` di kelas SoalGenerator menggunakan sebuah klas *dummy* yaitu kelas TesSoal1. Atribut dan operasi yang ada dalam klas TesSoal seperti ditunjukkan dalam gambar 6.1.



Gambar 6.1. Diagram Kelas TesSoal1

Sumber : Pengujian dan Analisis

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing – masing kasus uji dijelaskan pada tabel 6.2.

Tabel 6.2. *Test Case* Pengujian Unit Operasi `setSoal()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Parameter lokasi file yang diberikan salah sehingga file tidak ditemukan atau terdapat kesalahan pembacaan file.	Menampilkan laporan kesalahan, listSoal *size* bernilai 0 dan mengembalikan nilai 0.	Menampilkan laporan kesalahan, listSoal *size* bernilai 0 dan mengembalikan nilai 0.
2	Parameter lokasi file	listSoal *size* bernilai 0	listSoal *size* bernilai 0

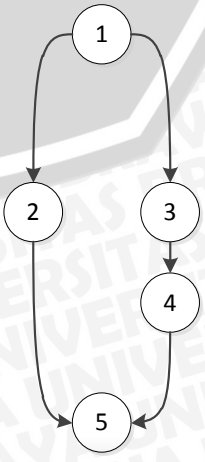
	yang diberikan benar tetapi isi file kosong.	dan mengembalikan nilai listSoal *size*.	dan mengembalikan nilai listSoal *size*.
3	Parameter lokasi file yang diberikan benar dan berisi list soal.	listSoal berisi data soal, *size* listSoal bernilai sesuai jumlah soal didalam listSoal dan mengembalikan nilai sesuai jumlah soal.	listSoal berisi data soal, *size* listSoal bernilai sesuai jumlah soal didalam listSoal dan mengembalikan nilai sesuai jumlah soal.

Sumber: Pengujian dan Analisis

6.1.1.2. Pengujian Unit Operasi `getRandomSoal()`

Operasi `getRandomSoal()` merupakan implementasi algoritma random soal. Operasi ini untuk memilih soal dari list soal secara random, lalu menghapusnya dari list soal agar soal tersebut tidak muncul kembali. Operasi ini terdapat dalam kelas SoalGenerator. Pemodelan operasi `getRandomSoal()` dalam bentuk *flow graph* ditunjukkan pada tabel 6.3.

Tabel 6.3. Pemodelan *Flow Graph* Algoritma `getRandomSoal()`

Pseudocode	*Flow Graph*
Methode : getRandomSoal IS String //atribut kelas listSoal IS StringList curSoal IS String DECLARATION: randomINum IS int DESCRIPTION : 1 IF(listSoal.size<1)THEN 2 curSoal=”END” 3 RETURN curSoal 4 ELSE	Node (N)= 5 Edge (E)= 5 

```
4   5    randomNum <- CALL random(listSoal.size)
    6    curSoal <- CALL listSoal.get(randomNumber)
    7    CALL listSoal.remove(curSoal)
    8    RETURN curSoal
    9  END IF
   10  END getRandomSoal
                          5
```

Sumber : Pengujian dan Analisis

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

V(G) = E – N + 2

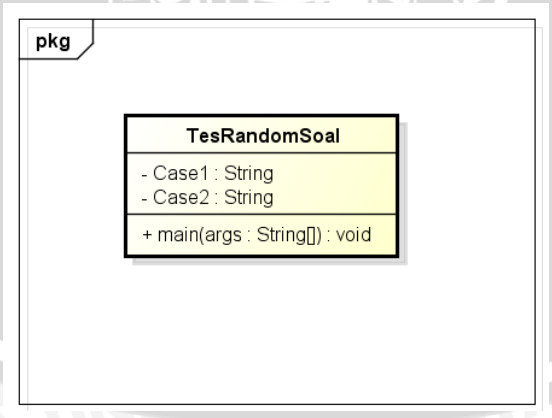
= 5 – 5 + 2 = 2

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

Jalur 1: 1 – 2 – 5

Jalur 2: 1 – 3 – 4 – 5

Proses pengujian operasi `getRandomSoal()` di kelas SoalGenerator menggunakan sebuah klas *dummy* yaitu kelas TesRandomSoal. Atribut dan operasi yang ada dalam kelas TesSoal seperti ditunjukkan dalam gambar 6.2.



Gambar 6.2. Diagram Kelas TesRandomSoal

Sumber : Pengujian dan Analisis

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing – masing kasus uji dijelaskan pada tabel 6.4.

Tabel 6.4. *Test Case* Pengujian Unit Operasi `getRandomSoal()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Objek listSoal berisi daftar soal.	Mengembalikan nilai string berupa soal acak dan soal yang terpilih langsung dihapus dari objek listSoal agar tidak muncul kembali dalam satu permainan.	Mengembalikan nilai string berupa soal acak dan soal yang terpilih langsung dihapus dari objek listSoal agar tidak muncul kembali dalam satu permainan.
2	Objek listSoal tidak berisi daftar soal.	Mengembalikan nilai string yaitu kalimat "END" yang menandakan soal telah habis atau kosong.	Mengembalikan nilai string yaitu kalimat "END" yang menandakan soal telah habis atau kosong.

Sumber: Pengujian dan Analisis

6.1.1.3. Pengujian Unit Operasi `buildItemKar()`

Operasi `buildItemKar()` merupakan implementasi algoritma membuat item karakter. Operasi ini terdapat dalam kelas RandomerCharNBom. Pemodelan operasi `buildItemKar()` dalam bentuk *flow graph* ditunjukkan pada tabel 6.5.

Tabel 6.5. Pemodelan *Flow Graph* Algoritma `buildItemKar()`

Pseudocode	*Flow Graph*
Methode : buildItemKar PARAMETER assetManager, posX, posY, Kar IS Spatial DECLARATION : TYPE assetManager IS AssetManager posX IS float posY IS float Kar IS String BoxKar IS Box Item IS Spatial Mat IS Material Tex IS Texture DESCRIPTION: 1 BoxKar <- CREATE OBJEK Box() 2 Item <- CREATE OBJEK Geometry(Kar,BoxKar) 3 Mat <- CREATE OBJEK Material() 4 Tex <- CALL loadTexture(Kar) 5 CALL Mat.setTexture(Tex) 6 CALL Item.setMaterial(Mat) 7 CALL Item.setLocalTranslation(posX,posy,0) 8 RETURN Item 9 END buildItemKar	Node (N)= 1 Edge (E)= 0 1 1

Sumber: Pengujian dan Analisis

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

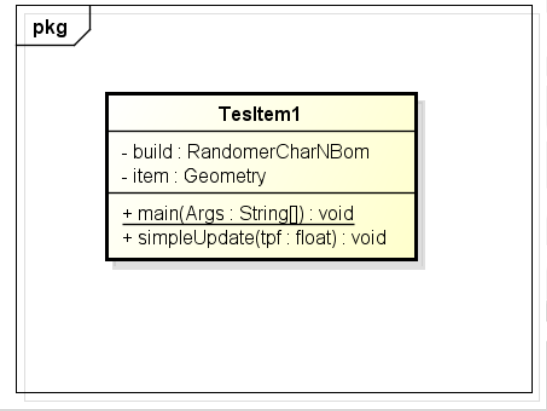
V(G) = E – N + 2

= 0 – 1 + 2 = 1

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

Jalur 1: 1

Proses pengujian operasi `buildItemKar()` di kelas RandomerCharNBom menggunakan sebuah klas *dummy* yaitu kelas TesItem1. Atribut dan operasi yang ada dalam kelas TesItem1 seperti ditunjukkan dalam gambar 6.3.



Gambar 6.3. Diagram Kelas TesItem1
Sumber : Pengujian dan Analisis

Penentuan kasus uji dan hasil eksekusi kasus uji dijelaskan pada tabel 6.6.

Tabel 6.6. *Test Case* Pengujian Unit Operasi `buildItemKar()`

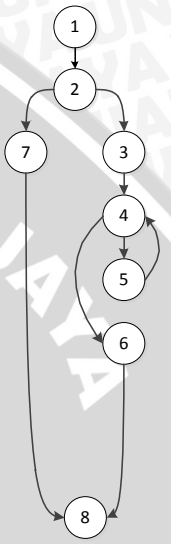
Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Membuat item karakter huruf tertentu yang ditranslasikan pada lokasi x dan y tertentu.	Mengembalikan objek spatial berupa item karakter huruf yang sesuai dengan parameter masukan dengan lokasi translasi x dan y sesuai dengan parameter masukan.	Mengembalikan objek spatial berupa item karakter huruf yang sesuai dengan parameter masukan dengan lokasi translasi x dan y sesuai dengan parameter masukan yang sesuai.

Sumber: Pengujian dan Analisis

6.1.1.4. Pengujian Unit Operasi `bacaHighScore()`

Operasi `bacaHighScore()` merupakan implementasi algoritma baca *high score*. Operasi ini terdapat dalam kelas HighScoreGenerator. Pemodelan operasi `bacaHighScore()` dalam bentuk *flow graph* ditunjukkan pada tabel 6.7.

Tabel 6.7. Pemodelan *Flow Graph* Algoritma `bacaHighScore()`

Pseudocode	*Flow Graph*
Methode : bacaHighScore PARAMETER path IS StringList listFiveScore IS StringList DECLARATION : TYPE path IS String highFiveFile IS File inputStream IS BufferedReader line IS String DESCRIPTION: 1 highFiveFile <- CREATE OBJEK File (path) 2 listFiveScore <- CREATE OBJEK StringList 3 TRY 4 inputStream <- CREATE OBJEK BufferedReader 5 (highFiveFile) 6 WHILE ((line <- CALL inputStream.readLine)!=null) DO 7 CALL listFiveScore.add(line) 8 END WHILE 9 CALL inputStream.close() 10 CATCH (IOException e) 11 CALL PrintError() 12 END TRY 13 RETURN listFiveScore 14 END bacaHighScore	Node (N)= 8 Edge (E)= 9 

Sumber : Pengujian dan Analisis

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

V(G) = E – N + 2

= 9 – 8 + 2 = 3

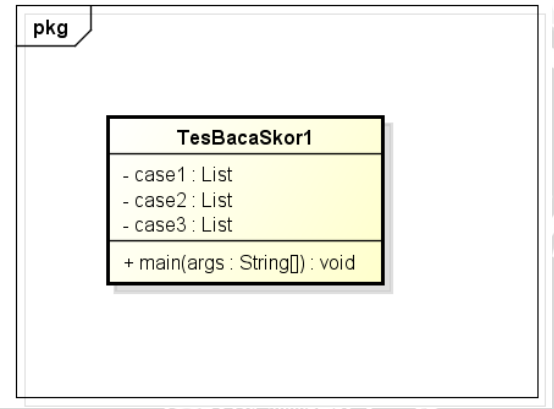
Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan tiga buah basis set dari jalur independent yaitu:

Jalur 1: 1 – 2 – 7 – 8

Jalur 2: 1 – 2 – 3 – 4 – 6 – 8

Jalur 3: 1 – 2 – 3 – 4 – 5 – 4 …..

Proses pengujian operasi `bacaHighScore()` di kelas HighScoreGenerator menggunakan sebuah klas *dummy* yaitu kelas TesBacaSkor1. Atribut dan operasi yang ada dalam kelas TesBacaSkor1 seperti ditunjukkan dalam gambar 6.4.



Gambar 6.4. Diagram Kelas TesBacaSkor1

Sumber : Pengujian dan Analisis

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing – masing kasus uji dijelaskan pada tabel 6.8.

Tabel 6.8. *Test Case* Pengujian Unit Operasi `bacaHighScore()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Parameter lokasi file yang diberikan salah sehingga file tidak ditemukan atau terdapat kesalahan pembacaan file.	Menampilkan laporan kesalahan, mengembalikan objek listFiveScore dengan *size* 0.	Menampilkan laporan kesalahan, mengembalikan objek listFiveScore dengan *size* 0.
2	Parameter lokasi file yang diberikan benar tetapi isi file kosong.	Mengembalikan objek listFiveScore dengan *size* 0.	Mengembalikan objek listFiveScore dengan *size* 0.

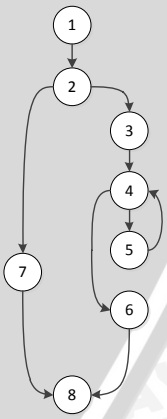
3	Parameter lokasi file yang diberikan benar dan berisi list nilai tertinggi.	Mengembalikan objek listFiveScore dengan *size* sesuai jumlah data di dalam file.	Mengembalikan objek listFiveScore dengan *size* sesuai jumlah data di dalam file.

Sumber: Pengujian dan Analisis

6.1.1.5. Pengujian Unit Operasi `tulisHighScore()`

Operasi `tulisHighScore()` merupakan implementasi algoritma tulis *high score*. Operasi ini terdapat dalam kelas HighScoreGenerator. Pemodelan operasi `tulisHighScore()` dalam bentuk *flow graph* ditunjukkan pada tabel 6.9.

Tabel 6.9. Pemodelan *Flow Graph* Algoritma `tulisHighScore()`

Pseudocode	*Flow Graph*
Methode : tulisFileHighScore PARAMETER path, listFiveScore IS boolean DECLARATION : TYPE path IS String listFiveScore IS StringList highFiveFile IS File writer IS BufferedWriter Status IS boolean DESCRIPTION: 1 highFiveFile <- CREATE OBJEK File (path) 2 Status <- false 3 TRY 4 writer <- CREATE OBJEK BufferedWriter 5 (highFiveFile) 6 CALL writer.write(listFiveScore.get(0)) 7 FOR (iterasi = 1 TO < listFiveScore.size STEP 1) 8 CALL writer.newLine 9 CALL writer.write(listFiveScore.get(iterasi)) 10 END FOR 11 Status <- true 12 CALL writer.close()	Node (N)= 8 Edge (E)= 9 

```
13  CATCH (IOException e)
14    CALL PrintError()
15    Status <- false
16  END TRY
17  RETURN Status
18  END tulisFileHighScore
```

7

8

Sumber : Pengujian dan Analisis

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

V(G) = E – N + 2

= 9 – 8 + 2 = 3

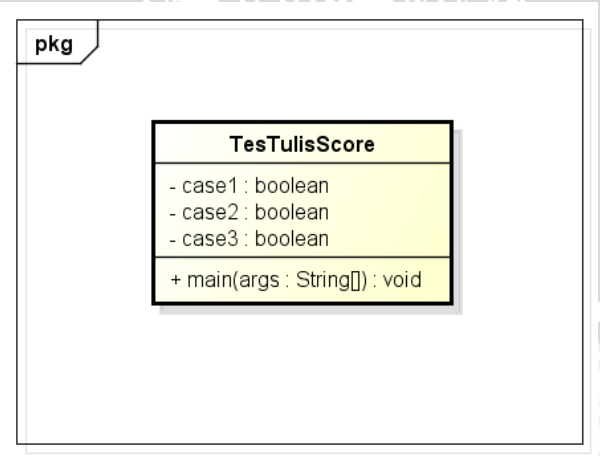
Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan tiga buah basis set dari jalur independent yaitu:

Jalur 1: 1 – 2 – 7 – 8

Jalur 2: 1 – 2 – 3 – 4 – 6 – 8

Jalur 3: 1 – 2 – 3 – 4 – 5 – 4 ….

Proses pengujian operasi `tulisHighScore()` di kelas HighScoreGenerator menggunakan sebuah klas *dummy* yaitu kelas TesTulisSkor. Atribut dan operasi yang ada dalam kelas TesTulisSkor seperti ditunjukkan dalam gambar 6.5.



Gambar 6.5. Diagram Kelas TesTulisScore

Sumber : Pengujian dan Analisis

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing – masing kasus uji dijelaskan pada tabel 6.10.

Tabel 6.10. *Test Case* Pengujian Unit Operasi `tulisHighScore()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Terdapat kesalahan pada proses penulisan file.	Menampilkan laporan kesalahan, status bernilai false.	Menampilkan laporan kesalahan, status bernilai false.
2	Tidak ada kesalahan dalam penulisan file dan isi listFiveScore ada 1.	Hanya satu data nilai tertinggi tertulis dalam file dan status bernilai true.	Hanya satu data nilai tertinggi tertulis dalam file dan status bernilai true.
3	Tidak ada kesalahan dalam penulisan file dan isi listFiveScore lebih dari 1.	Semua data nilai tertinggi tertulis dalam file dan status bernilai true.	Semua data nilai tertinggi tertulis dalam file dan status bernilai true.

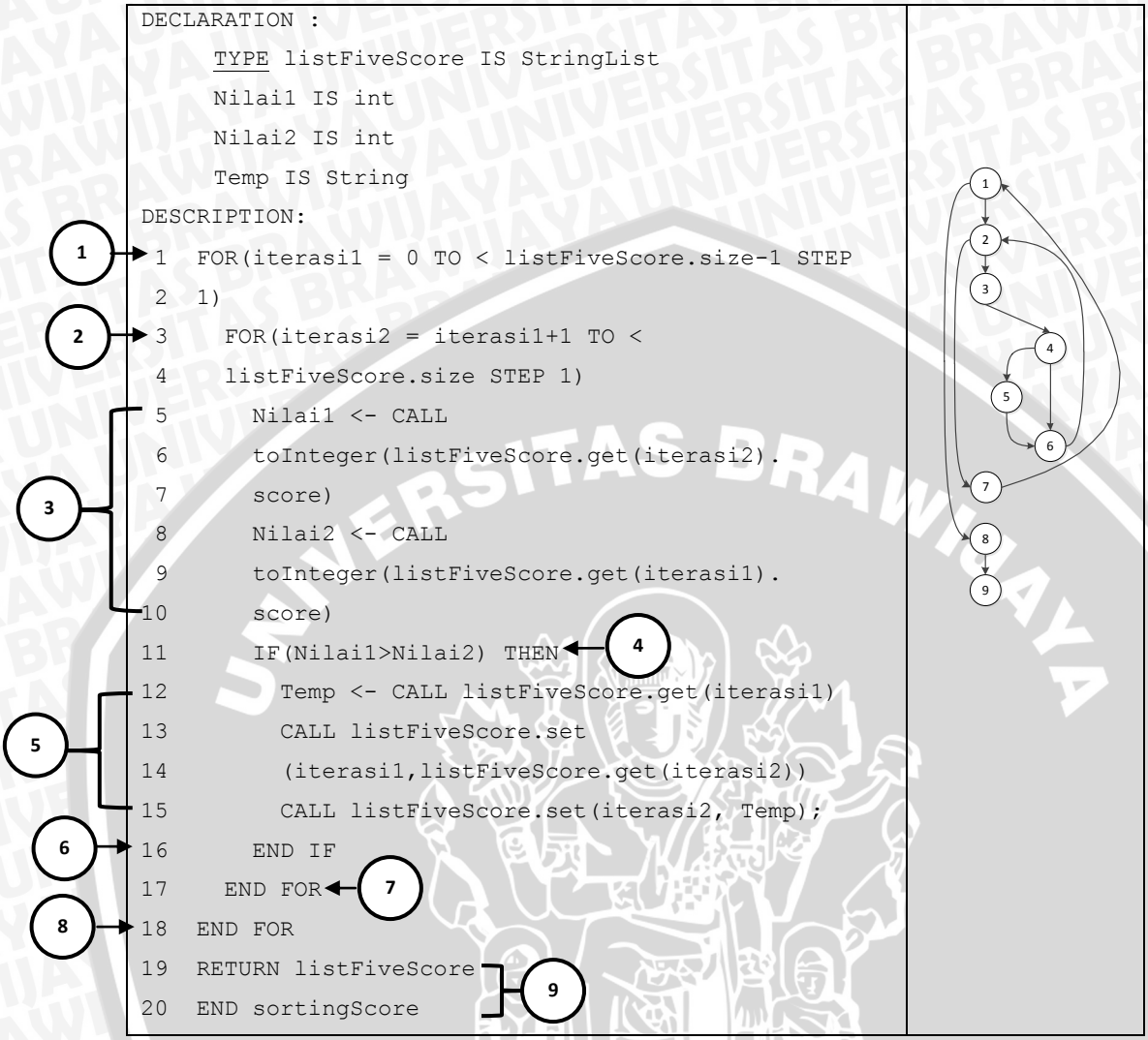
Sumber: Pengujian dan Analisis

6.1.1.6. Pengujian Unit Operasi `sortingScore()`

Operasi `sortingScore()` merupakan implementasi algoritma *sorting score*. Operasi ini terdapat dalam kelas HighScoreGenerator. Pemodelan operasi `sortingScore()` dalam bentuk *flow graph* ditunjukkan pada tabel 6.11.

Tabel 6.11. Pemodelan *Flow Graph* Algoritma `sortingScore()`

Pseudocode	*Flow Graph*
`Methode : sortingScore PARAMETER listFiveScore IS StringList`	`Node (N)= 9` `Edge (E)= 11`



Sumber : Pengujian dan Analisis

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

V(G) = E – N + 2

= 11 – 9 + 2 = 4

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

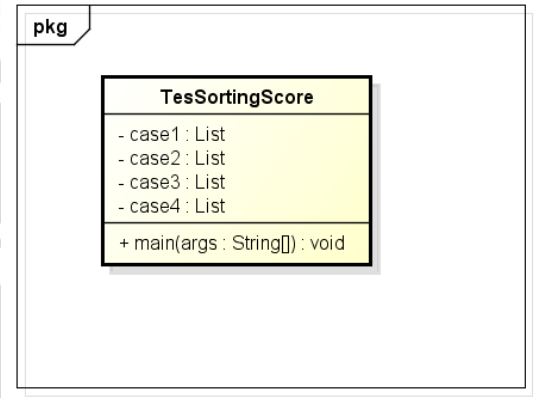
Jalur 1: 1 – 8 – 9

Jalur 2: 1 – 2 – 7 – 1 – …

Jalur 3: 1 – 2 – 3 – 4 – 5 – 6 – 2…

Jalur 4: 1 – 2 – 3 – 4 – 6 – 2…

Proses pengujian operasi `sortingScore()` di kelas HighScoreGenerator menggunakan sebuah klas *dummy* yaitu kelas TesSortingSkor. Atribut dan operasi yang ada dalam kelas TesSortingSkor seperti ditunjukkan dalam gambar 6.6.



Gambar 6.6. Diagram Kelas TesSortingSkor

Sumber : Pengujian dan Analisis

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing – masing kasus uji dijelaskan pada tabel 6.12.

Tabel 6.12. *Test Case* Pengujian Unit Operasi `sortingScore()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	listFiveScore tidak berisi data skor.	Mengembalikan listFiveScore tanpa diproses *sorting.*	Mengembalikan listFiveScore tanpa diproses *sorting.*
2	listFiveScore berisi satu data skor.	Mengembalikan listFiveScore tanpa diproses *sorting.*	Mengembalikan listFiveScore tanpa diproses *sorting.*
3	listFiveScore berisi lebih dari satu data skor dan terdapat	Mengembalikan listFiveScore yang telah diproses *sorting*, data	Mengembalikan listFiveScore yang telah diproses *sorting*,

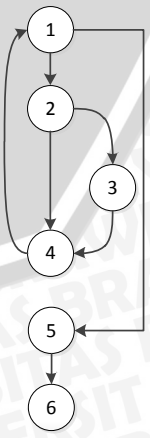
	data yang belum urut.	yang belum urut diurutkan secara *descending*.	data yang belum urut diurutkan secara *descending*.
4	listFiveScore berisi lebih dari satu data skor dan terdapat data yang telah urut.	Mengembalikan listFiveScore yang telah diproses *sorting*, data yang telah urut tidak perlu diproses *sorting*.	Mengembalikan listFiveScore yang telah diproses *sorting*, data yang telah urut tidak perlu diproses *sorting*.

Sumber: Pengujian dan Analisis

6.1.1.7. Pengujian Unit Operasi `isTopFive()`

Operasi `isTopFive()` merupakan implementasi algoritma *score filter*. Operasi ini terdapat dalam kelas HighScoreGenerator. Pemodelan operasi `isTopFive()` dalam bentuk *flow graph* ditunjukkan pada tabel 6.13.

Tabel 6.13. Pemodelan *Flow Graph* Algoritma `isTopFive()`

Pseudocode	*Flow Graph*
Methode : isTopFive PARAMETER listFiveScore, nameNscore IS boolean DECLARATION : TYPE listFiveScore IS StringList nameNscore IS String nameNscoreOld IS String DESCRIPTION: 1 FOR (iterasi = 0 TO < 5 STEP 1) 2 IF(CALL toInteger(listFiveScore.get(iterasi).score) 3 < CALL toInteger(nameNscore.score)) THEN 4 nameNscoreOld <- CALL listFiveScore.get(iterasi) 5 CALL listFiveScore.set(iterasi, NameNscore) 6 CALL geser(listFiveScore,nameNscoreOld, iterasi) 7 RETURN true 8 END IF 9 END FOR	Node (N)= 6 Edge (E)= 7 

Sumber : Pengujian dan Analisis

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

V(G) = E – N + 2

= 7 – 6 + 2 = 3

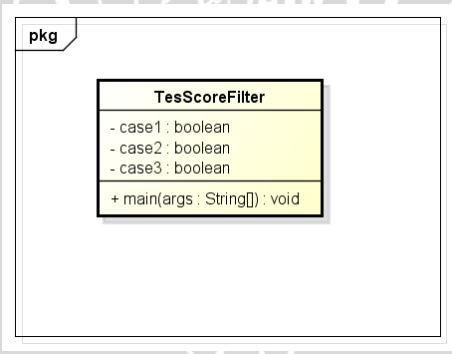
Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan tiga buah basis set dari jalur independent yaitu:

Jalur 1: 1 – 5 – 6

Jalur 2: 1 – 2 – 3 – 4 – 1 – …..

Jalur 3: 1 – 2 – 4 – 1 – …..

Proses pengujian operasi `isTopFive()` di kelas HighScoreGenerator menggunakan sebuah klas *dummy* yaitu kelas TesFilterSkor. Atribut dan operasi yang ada dalam kelas TesFilterSkor seperti ditunjukkan dalam gambar 6.7.



Gambar 6.7. Diagram Kelas TesFilterSkor

Sumber : Pengujian dan Analisis

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing – masing kasus uji dijelaskan pada tabel 6.14.

Tabel 6.14. *Test Case* Pengujian Unit Operasi `isTopFive()`

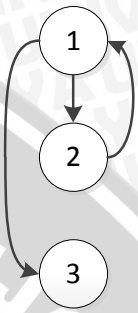
Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Iterasi telah sampai batas iterasi.	Mengembalikan nilai false yang berarti nilai pemain tidak termasuk lima nilai tertinggi.	Mengembalikan nilai false yang berarti nilai pemain tidak termasuk lima nilai tertinggi.
2	Pada iterasi tertentu skor pemain masih lebih kecil dari nilai pemain pada urutan iterasi.	Mengulang iterasi sampai iterasi selesai, tetapi jika belum mencapai batas iterasi dan nilai pemain lebih tinggi dari nilai pada urutan iterasi maka akan mengembalikan nilai true dan nilai pemain masuk lima nilai tertinggi menggeser nilai di bawahnya.	Mengulang iterasi sampai iterasi selesai, tetapi jika belum mencapai batas iterasi dan nilai pemain lebih tinggi dari nilai pada urutan iterasi maka akan mengembalikan nilai true dan nilai pemain masuk lima nilai tertinggi menggeser nilai di bawahnya.
3	Pada iterasi tertentu skor pemain lebih besarl dari nilai pemain pada urutan iterasi.	Mengembalikan nilai true dan nilai pemain masuk lima nilai tertinggi menggeser nilai di bawahnya.	Mengembalikan nilai true dan nilai pemain masuk lima nilai tertinggi menggeser nilai di bawahnya.

Sumber: Pengujian dan Analisis

6.1.1.8. Pengujian Unit Operasi `geser()`

Operasi `geser()` merupakan implementasi algoritma geser skor. Operasi ini terdapat dalam kelas HighScoreGenerator. Pemodelan operasi `geser()` dalam bentuk *flow graph* ditunjukkan pada tabel 6.15.

Tabel 6.15. Pemodelan *Flow Graph* Algoritma `geser()`

Pseudocode	*Flow Graph*
Methode : geser PARAMETER listFiveScore,nameNscore, pos DECLARATION : TYPE listFiveScore IS StringList pos IS int curPos IS int nameNscore IS String curNameNscore IS String DESCRIPTION: 1 IF (pos < 4) THEN (1) 2 curPos <- pos+1 3 curNameNscore <- CALL listFiveScore.get(curPos) 4 CALL listFiveScore.set (curPos,nameNscore) 5 CALL geser(listFiveScore,curNameNscore,curPos) 6 END IF (3) 7 END geser (2)	Node (N)= 3 Edge (E)= 3 

Sumber : Pengujian dan Analisis

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

V(G) = E – N + 2

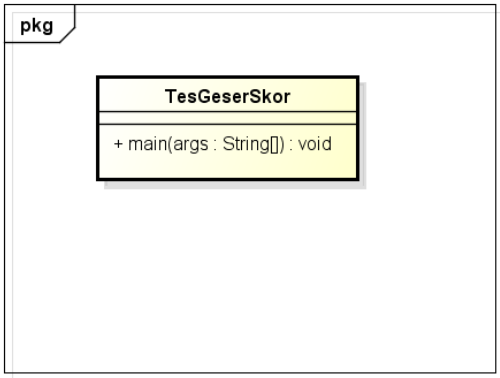
= 3 – 3 + 2 = 2

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

Jalur 1: 1 – 3

Jalur 2: 1 – 2 – 1 – …..

Proses pengujian operasi `geser()` di kelas HighScoreGenerator menggunakan sebuah klas *dummy* yaitu kelas TesGeserSkor. Atribut dan operasi yang ada dalam kelas TesGeserSkor seperti ditunjukkan dalam gambar 6.8.



Gambar 6.8. Diagram Kelas TesGeserSkor

Sumber : Pengujian dan Analisis

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing – masing kasus uji dijelaskan pada tabel 6.16.

Tabel 6.16. *Test Case* Pengujian Unit Operasi `geser()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Jika posisi yang digeser adalah posisi terakhir atau >= 4.	Tidak dilakukan proses geser skor.	Tidak dilakukan proses geser skor.
2	Jika posisi yang digeser adalah posisi terakhir < 4.	Menggeser skor dan menghapus nilai terendah.	Menggeser skor dan menghapus nilai terendah.

Sumber: Pengujian dan Analisis

6.1.1.9. Pengujian Unit Operasi `moveKursor()`

Operasi `moveKursor()` merupakan implementasi algoritma gerakan kursor pemain. Operasi ini terdapat dalam kelas InGame. Pemodelan operasi `moveKursor()` dalam bentuk *flow graph* ditunjukkan pada tabel 6.17.

Tabel 6.17. Pemodelan *Flow Graph* Algoritma `moveKursor()`

Pseudocode	*Flow Graph*

`METHODE: moveKursor PARAMETER posX, posy`
 	
`kursorItem IS Spatial`
 	
`DECLARATION:`
`TYPE posX IS float`
`posY IS float`
 	
`DESCRIPTION:`
`1 CALL kursorItem.setLocalTranslation(posX,posY,0)`
`2 END moveKursor`
	`Node (N)= 1`
	`Edge (E)= 0`
	
	1

Sumber : Pengujian dan Analisis

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

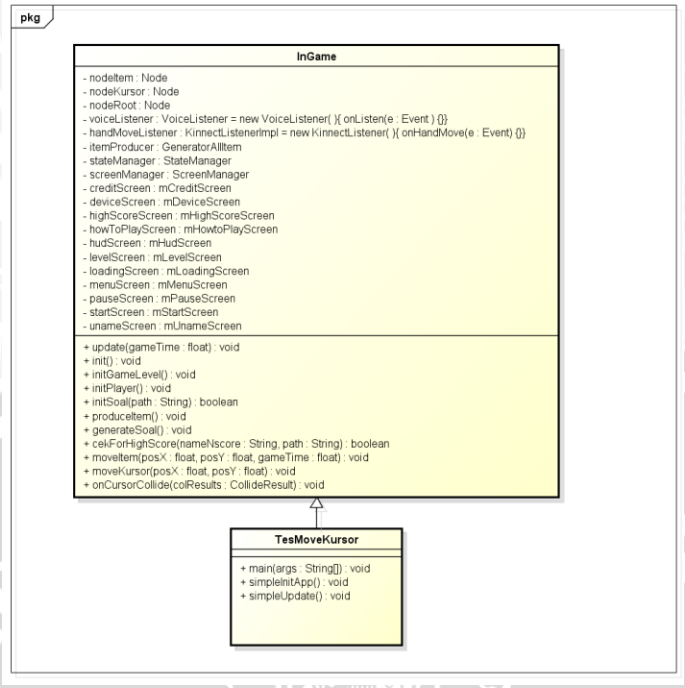
V(G) = E – N + 2

= 0 – 1+ 2 = 1

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

Jalur 1: 1

Proses pengujian operasi `moveKursor()` dilakukan pada kelas TesMoveKursor. Atribut dan operasi yang ada dalam kelas TesMoveKursor seperti ditunjukkan dalam gambar 6.9.



Gambar 6.9. Diagram Kelas InGame

Sumber : Pengujian dan Analisis

Penentuan kasus uji dan hasil eksekusi dijelaskan pada tabel 6.18.

Tabel 6.18. *Test Case* Pengujian Unit Operasi `moveKursor()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Memberikan masukan nilai float tertentu ke parameter koordinat x dan y.	Kursor tangan berada pada posisi x dan y sesuai nilai yang diberikan di parameter.	Kursor tangan berada pada posisi x dan y sesuai nilai yang diberikan di parameter.

Sumber: Pengujian dan Analisis

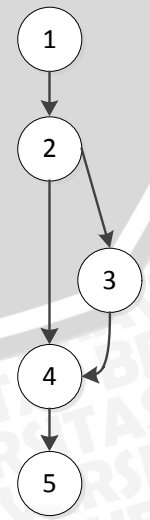
6.1.2. Pengujian Integrasi

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

6.1.2.1. Pengujian Integrasi Operasi `Initsoal()`

Operasi `InitSoal()` merupakan implementasi algoritma inisialisasi soal. Operasi ini terdapat dalam kelas InGame. Pemodelan operasi `initSoal()` dalam bentuk *flow graph* ditunjukkan pada tabel 6.19.

Tabel 6.19. Pemodelan *Flow Graph* Algoritma `initSoal()`

Pseudocode	*Flow Graph*
METHODE: initSoal PARAMETER pathSoal IS boolean DECLARATION: TYPE pathSoal IS String curSoal IS String DESCRIPTION: 1 curSoal <- "" 2 IF(CALL setSoal(pathSoal) > 0) THEN 3 curSoal <- CALL getRandomSoal() 4 RETURN true 5 END IF 6 RETURN false 7 END initSoal	Node (N)= 5 Edge (E)= 5 

Sumber : Pengujian dan Analisis

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

V(G) = E – N + 2

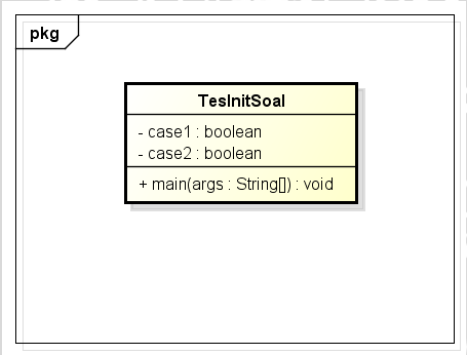
= 5 – 5 + 2 = 2

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 – 5

Jalur 2: 1 – 2 – 4 – 5

Proses pengujian operasi `initSoal()` dilakukan pada kelas TesInitSoal. Atribut dan operasi yang ada dalam kelas TesInitSoal seperti ditunjukkan dalam gambar 6.10.



Gambar 6.10. Diagram Kelas TesInitSoal

Sumber : Pengujian dan Analisis

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing – masing kasus uji dijelaskan tabel 6.20.

Tabel 6.20. *Test Case* Pengujian Integrasi Operasi `initSoal()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Parameter lokasi file yang diberikan salah	Mengembalikan nilai false dan objek curSoal	Mengembalikan nilai false dan objek

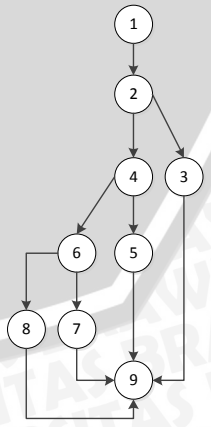
	sehingga file tidak ditemukan atau file tidak berisi soal .	tidak berisi soal.	curSoal tidak berisi soal.
2	Parameter lokasi file yang diberikan benar dan file berisi soal.	Mengembalikan nilai true dan objek curSoal berisi soal pertama secara acak.	Mengembalikan nilai true dan objek curSoal berisi soal pertama secara acak.

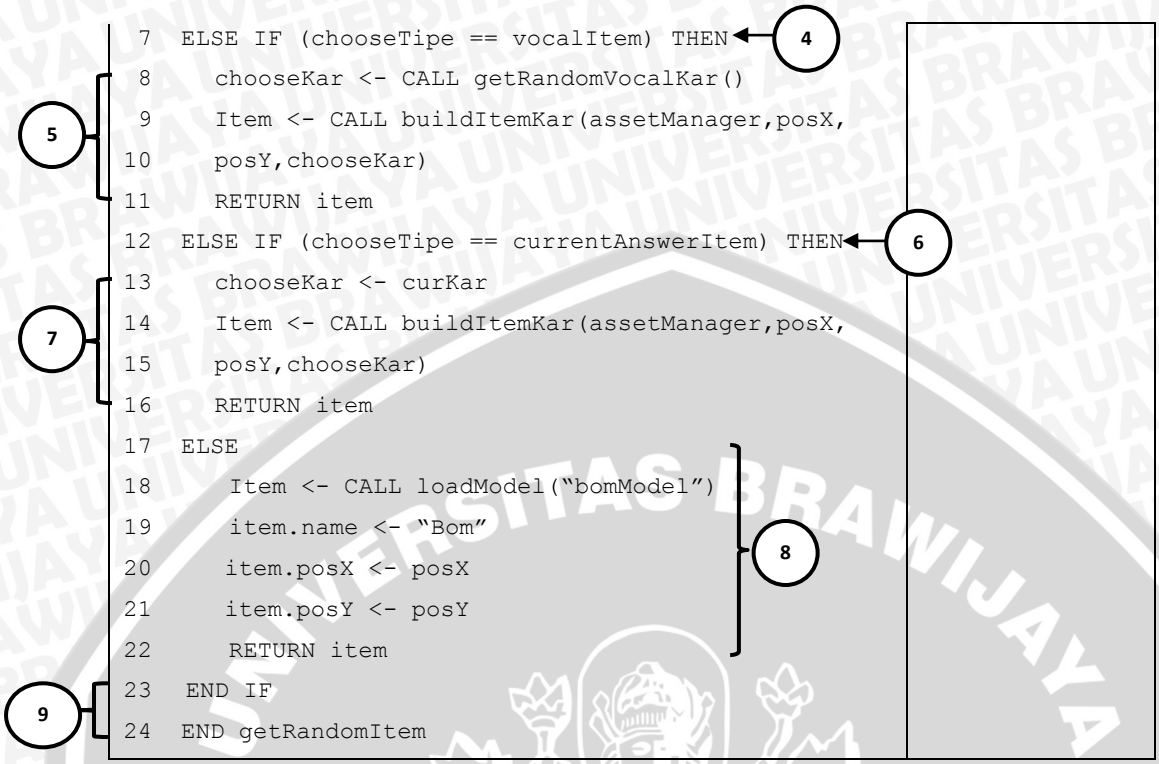
Sumber: Pengujian dan Analisis

6.1.2.2. Pengujian Integrasi Operasi `getRandomItem()`

Operasi `getRandomItem()` merupakan implementasi algoritma *generate random item*. Operasi ini terdapat dalam kelas RandomerCharNBom. Pemodelan operasi `getRandomItem()` dalam bentuk *flow graph* ditunjukkan pada tabel 6.21.

Tabel 6.21. Pemodelan *Flow Graph* Algoritma `getRandomItem()`

Pseudocode	*Flow Graph*
Methode: getRandomItem PARAMETER assetManager, posX, posY,curKar IS Spatial DECLARATION: TYPE assetManager IS AssetManager posX IS float posY IS float curKar IS char chooseTipe IS ItemTipe chooseKar IS char item is Spatial DESCRIPTION: 1 chooseTipe <- CALL randomItemTipe() 2 IF(chooseTipe == konsonanItem) THEN 3 chooseKar <- CALL getRandomKonsonanKar() 4 Item <- CALL buildItemKar(assetManager,posX, 5 posY,chooseKar) 6 RETURN item	Node (N)= 9 Edge (E)= 11 



Sumber : Pengujian dan Analisis

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

V(G) = E – N + 2

= 11 – 9 + 2 = 4

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

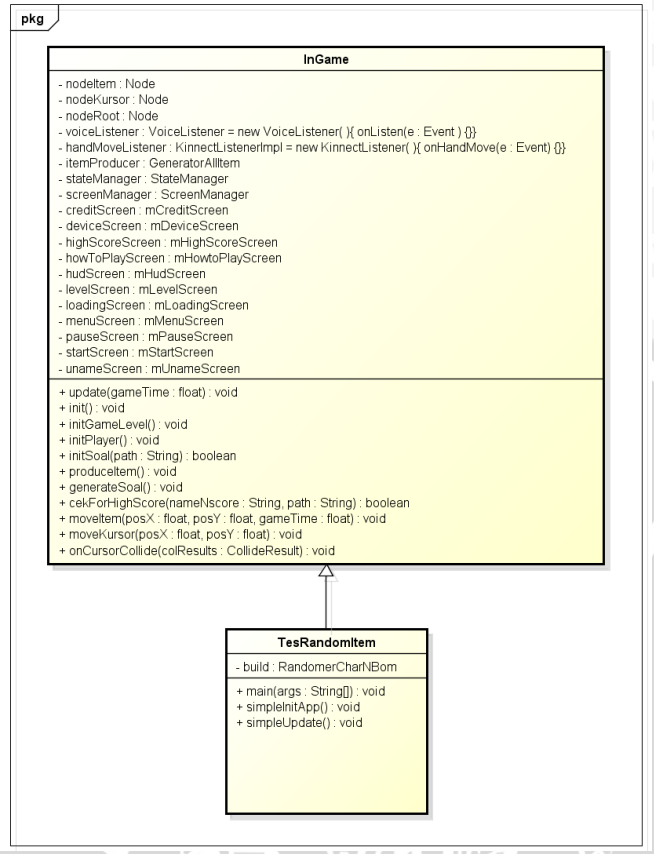
Jalur 1: 1 – 2 – 3 – 9

Jalur 2: 1 – 2 – 4 – 5 – 9

Jalur 3: 1 – 2 – 4 – 6 – 7 – 9

Jalur 4: 1 – 2 – 4 – 6 – 8 – 9

Proses pengujian operasi `getRandomItem()` dilakukan pada kelas TesRandomItem. Atribut dan operasi yang ada dalam kelas TesRandomItem seperti ditunjukkan dalam gambar 6.11.



Gambar 6.11. Diagram Kelas TesRandomItem

Sumber : Pengujian dan Analisis

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing – masing kasus uji dijelaskan tabel 6.22.

Tabel 6.22. *Test Case* Pengujian Integrasi Operasi `getRandomItem()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Hasil random Objek chooseTipe berisi konsonan item.	Mengembalikan item huruf konsonan acak.	Mengembalikan item huruf konsonan acak.
2	Hasil random Objek chooseTipe berisi vocal item.	Mengembalikan item huruf vocal acak.	Mengembalikan item huruf vocal acak.

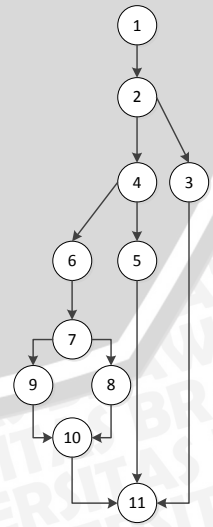
3	Hasil random Objek chooseTipe berisi current answer item.	Mengembalikan item huruf jawaban.	Mengembalikan item huruf jawaban.
4	Hasil random Objek chooseTipe berisi bom.	Mengembalikan item bom.	Mengembalikan item bom.

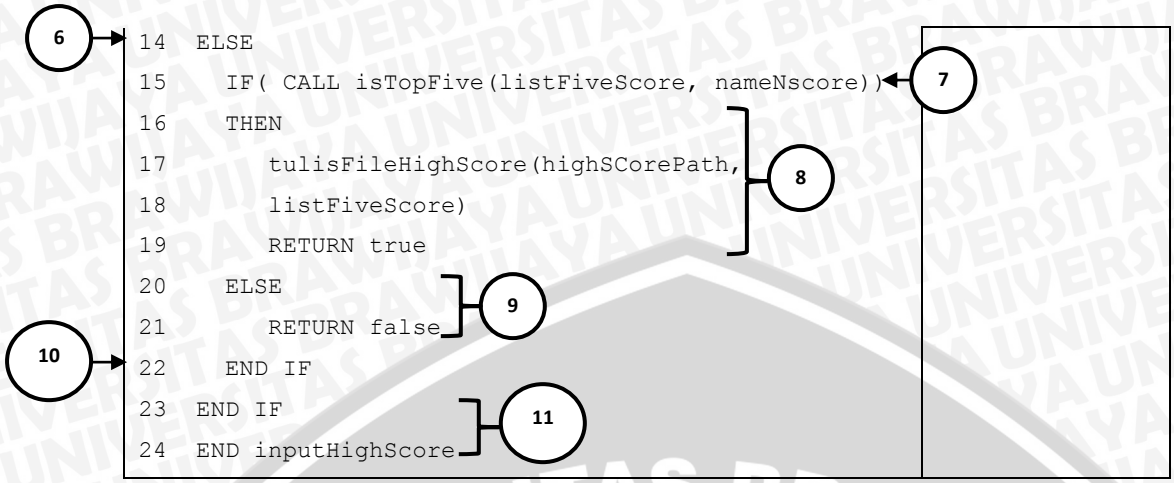
Sumber: Pengujian dan Analisis

6.1.2.3. Pengujian Integrasi Operasi `inputHighScore()`

Operasi `inputHighScore()` merupakan implementasi algoritma *input high score*. Operasi ini terdapat dalam kelas HighScoreGenerator. Pemodelan operasi `inputHighScore()` dalam bentuk *flow graph* ditunjukkan pada tabel 6.23.

Tabel 6.23. Pemodelan *Flow Graph* Algoritma `inputHighScore()`

Pseudocode	*Flow Graph*
METHODE: inputHighScore PARAMETER nameNscore, highScorePath IS boolean DECLARATION: TYPE nameNscore IS String highScorePath IS String listFiveScore IS StringList nameNscoreOld IS String DESCRIPTION: 1 listFiveScore <- CALL bacaHighScore() 2 IF(listFiveScore.size <=0) THEN 3 CALL listFiveScore.add(nameNscore) 4 CALL tulisFileHighScore(highSCorePath, 5 listFiveScore) 6 RETURN true 7 ELSE IF (listScore.size < 5 AND listScore.size 8 >0) THEN 9 CALL listFiveScore.add(nameNscore) 10 CALL sortingScore(listFiveScore) 11 CALL tulisFileHighScore(highSCorePath, 12 listFiveScore) 13 RETURN true	Node (N)= 11 Edge (E)= 13 



Sumber : Pengujian dan Analisis

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

V(G) = E – N + 2

= 13 – 11 + 2 = 4

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

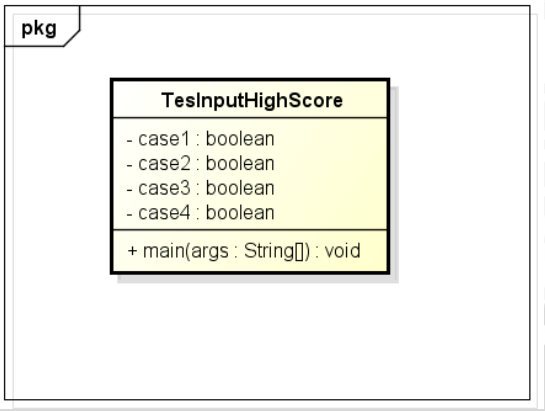
Jalur 1: 1 – 2 – 3 – 11

Jalur 2: 1 – 2 – 4 – 5 – 11

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

Jalur 4: 1 – 2 – 4 – 6 – 7 – 9 – 10 – 11

Proses pengujian operasi `inputHighScore()` dilakukan pada kelas TesInputHighScore. Atribut dan operasi yang ada dalam kelas TesInputHighScore seperti ditunjukkan dalam gambar 6.12.



Gambar 6.12. Diagram Kelas TesRandomItem
Sumber : Pengujian dan Analisis

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing – masing kasus uji dijelaskan tabel 6.24.

Tabel 6.24. *Test Case* Pengujian Integrasi Operasi `inputHighScore()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Belum ada nilai tertinggi yang disimpan.	Nilai baru langsung disimpan sebagai nilai tertinggi dan operasi ini mengembalikan nilai true.	Nilai baru langsung disimpan sebagai nilai tertinggi dan operasi ini mengembalikan nilai true.
2	Telah terdapat minimal 1 nilai dan maksimal 4 nilai tertinggi .	Nilai baru langsung disimpan sebagai lima nilai tertinggi, nilai – nilai yang telah tersimpan di *sorting* secara descending dan operasi ini mengembalikan nilai true.	Nilai baru langsung disimpan sebagai lima nilai tertinggi, nilai – nilai yang telah tersimpan di *sorting* secara descending dan operasi ini

			mengembalikan nilai true.
3	Telah terdapat 5 nilai tertinggi dan skor baru lebih tinggi dari nilai terendah atau masuk dalam lima nilai tertinggi.	Nilai baru langsung disimpan sebagai lima nilai tertinggi dan menggeser nilai – nilai dibawahnya. Nilai terendah akan dihapus, sehingga tetap hanya ada lima nilai tertinggi. Operasi ini mengembalikan nilai true.	Nilai baru langsung disimpan sebagai lima nilai tertinggi dan menggeser nilai – nilai dibawahnya. Nilai terendah akan dihapus, sehingga tetap hanya ada lima nilai tertinggi. Operasi ini mengembalikan nilai true.
4	Telah terdapat 5 nilai tertinggi dan skor baru lebih rendah dari nilai terendah.	Operasi ini mengembalikan nilai false, dan nilai baru tidak disimpan sebagai lima nilai tertinggi.	Operasi ini mengembalikan nilai false, dan nilai baru tidak disimpan sebagai lima nilai tertinggi.

Sumber: Pengujian dan Analisis

6.1.2.4. Pengujian Integrasi Operasi `onPointUpdate()`

Operasi `onPointUpdate()` merupakan implementasi algoritma *on hamd move*. Operasi ini terdapat dalam kelas InGame. Pemodelan operasi `onPointUpdate()` dalam bentuk *flow graph* ditunjukkan pada tabel 6.25.

Tabel 6.25. Pemodelan *Flow Graph* Algoritma `onPointUpdate()`

Pseudocode	*Flow Graph*

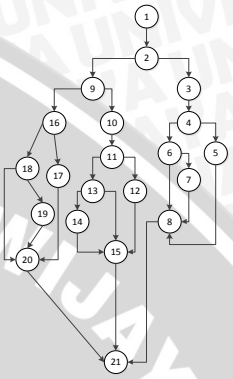
```
METHODE: onPointUpdateEvent PARAMETER name, evt,
fps

DECLARATION:
     TYPE evt IS HandEvent
     name IS String
     fps IS float
     posX IS float
     posY IS float

DESCRIPTION:
 1   posX <- CALL evt.getHand().getPosition().getX()
 2   posy <- CALL evt.getHand().getPosition().getY()
 3   IF (posX > batasPositiveX) THEN
 4     PosX <- batasPositiveX
 5     IF (posY > batasPositiveY) THEN
 6       PosY <- batasPositiveY
 7     ELSE IF (posY < batasNegativeY)
 8       PosY <- batasNegativeY
 9     END IF
10   ELSE IF (posX < batasNegativeX)
11     PosX <- batasNegativeX
12     IF (posY > batasPositiveY) THEN
13       PosY <- batasPositiveY
14     ELSE IF (posY < batasNegativeY)
15       PosY <- batasNegativeY
16     END IF
17   ELSE
18     IF (posY > batasPositiveY) THEN
19       PosY <- batasPositiveY
20     ELSE IF (posY < batasNegativeY)
21       PosY <- batasNegativeY
22     END IF
23   END IF
24   CALL moveKursor(posX,posy,0)
25   END onPointUpdateEvent
```

Node (N)= 21

Edge (E)= 28



Sumber : Pengujian dan Analisis

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

V(G) = E – N + 2

= 28 – 21 + 2 = 9

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

Jalur 1: 1 – 2 – 3 – 4 – 5 – 8 – 21

Jalur 2: 1 – 2 – 3 – 4 – 6 – 7 – 8 – 21

Jalur 3: 1 – 2 – 3 – 4 – 6 – 8 – 21

Jalur 4: 1 – 2 – 9 – 10 – 11 – 12 – 15 – 21

Jalur 5: 1 – 2 – 9 – 10 – 11 – 13 – 14 – 15 – 21

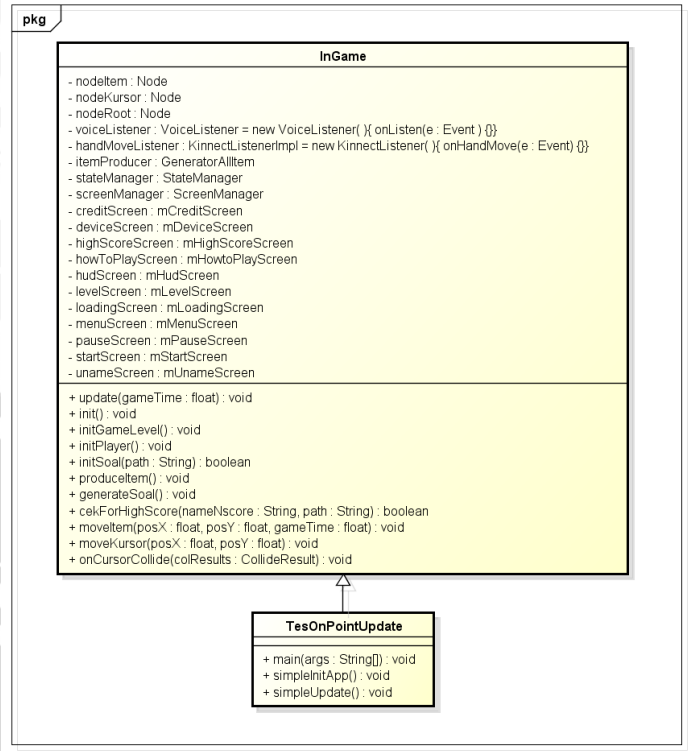
Jalur 6: 1 – 2 – 9 – 10 – 11 – 13 – 15 – 21

Jalur 7: 1 – 2 – 9 – 16 – 17 – 20 – 21

Jalur 8: 1 – 2 – 9 – 16 – 18 – 19 – 20 – 21

Jalur 9: 1 – 2 – 9 – 16 – 18 – 20 – 21

Proses pengujian operasi onPointUpdate() dilakukan pada kelas TesOnPointUpdate. Atribut dan operasi yang ada dalam kelas TesOnPointUpdate seperti ditunjukkan dalam gambar 6.13.



Gambar 6.13. Diagram Kelas TesOnPointUpdate

Sumber : Pengujian dan Analisis

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing – masing kasus uji dijelaskan tabel 6.26.

Tabel 6.26. *Test Case* Pengujian Integrasi Operasi `onPointUpdate()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Menggerakkan kursor tangan kearah kanan atas sampai melebihi batas kanan dan atas *game area.*	Kursor tangan tetap pada game area pada pojok kanan atas.	Kursor tangan tetap pada game area pada pojok kanan atas.
2	Menggerakkan kursor tangan kearah kanan bawah sampai	Kursor tangan tetap pada game area pada pojok kanan bawah.	Kursor tangan tetap pada game area pada pojok kanan bawah.

	melebihi batas kanan dan bawah *game area.*		
3	Menggerakkan kursor tangan kearah kanan sampai melebihi batas kanan *game area.*	Kursor tangan tetap pada game area pada batas kanan. X = batas kanan dan Y sesuai parameter masukan.	Kursor tangan tetap pada game area pada batas kanan. X = batas kanan dan Y sesuai parameter masukan
4	Menggerakkan kursor tangan kearah kiri atas sampai melebihi batas kiri dan atas *game area.*	Kursor tangan tetap pada game area pada pojok kiri atas.	Kursor tangan tetap pada game area pada pojok kiri atas.
5	Menggerakkan kursor tangan kearah kiri bawah sampai melebihi batas kiri dan bawah *game area.*	Kursor tangan tetap pada game area pada pojok kiri bawah.	Kursor tangan tetap pada game area pada pojok kiri bawah.
6	Menggerakkan kursor tangan kearah kiri sampai melebihi batas kiri *game area.*	Kursor tangan tetap pada game area pada batas kiri. X = batas kiri dan Y sesuai parameter masukan	Kursor tangan tetap pada game area pada batas kiri. X = batas kiri dan Y sesuai parameter masukan
7	Menggerakkan kursor tangan kearah atas sampai melebihi batas atas *game area*	Kursor tangan tetap pada game area pada batas atas. Y = batas atas dan X sesuai parameter masukan	Kursor tangan tetap pada game area pada batas atas. Y = batas atas dan X sesuai parameter masukan
8	Menggerakkan	Kursor tangan tetap	Kursor tangan tetap

	kursor tangan kearah bawah sampai melebihi batas bawah *game area*	pada game area pada batas bawah. Y = batas bawah dan X sesuai parameter masukan	pada game area pada batas bawah. Y = batas bawah dan X sesuai parameter masukan
9	Menggerakkan kursor tangan diantara batas game area	Kursor tangan tetap pada game dengan X dan Y sesuai parameter masukan	Kursor tangan tetap pada game dengan X dan Y sesuai parameter masukan

Sumber: Pengujian dan Analisis

6.1.3. Pengujian Validasi

Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang dibutuhkan. Daftar kebutuhan akan menjadi acuan untuk melakukan pengujian validasi. Pengujian validasi menggunakan metode pengujian *Black Box,* karena tidak diperlukan konsentrasi terhadap alur jalannya algoritma program dan lebih ditekankan untuk menemukan konformitas antara kinerja sistem dengan daftar kebutuhan. Pada skripsi ini dilakukan pengujian validasi terhadap *game* Slash Word.

6.1.3.1. Kasus Uji Validasi

Keseluruhan kasus uji validasi dijelaskan pada tabel 6.27.

Tabel 6.27. Kasus Uji Validasi.

No.	Kasus Uji	Objek Uji	Prosedur dan Input	Kondisi yang diharapkan	Kondisi Gagal
1	Melihat menu utama.	F01	Memulai *running* atau eksekusi *game* Slash Word.	Aplikasi dapat menampilkan halaman menu utama.	Aplikasi tidak dapat menampilkan halaman menu utama.

2	Melihat *high score*.	F02	Pemain memilih menu *high score* dengan perintah suara "*high score*" pada menu utama.	Aplikasi dapat menampilkan halaman *high score*.	Aplikasi tidak dapat menampilkan halaman *high score*.
3	Memulai permainan.	F03	Pemain melakukan perintah suara "*let's play*" pada menu utama.	Aplikasi dapat menampilkan halaman memilih level permainan.	Aplikasi tidak dapat menampilkan halaman memilih level permainan.
4	Memilih level permainan.	F04	Pemain melakukan perintah suara "*easy*" untuk memilih level *easy* pada menu level permainan.	Button *easy* melakukan animasi membesar lalu aplikasi menampilkan halaman memilih kendali permainan.	Button *easy* tidak melakukan animasi membesar, sehingga aplikasi juga tidak menampilkan halaman memilih kendali permainan.
			Pemain melakukan perintah suara "*medium*"	Button *medium* melakukan animasi membesar lalu	Button *medium* tidak melakukan animasi

			untuk memilih level *medium* pada menu level permainan.	aplikasi menampilkan halaman memilih kendali permainan.	membesar, sehingga aplikasi juga tidak menampilkan halaman memilih kendali permainan.
			Pemain melakukan perintah suara "*hard*" untuk memilih level *hard* pada menu level permainan.	Button *hard* melakukan animasi membesar lalu aplikasi menampilkan halaman memilih kendali permainan.	Button *hard* tidak melakukan animasi membesar, sehingga aplikasi juga tidak menampilkan halaman memilih kendali permainan.
5	Memilih kendali permainan.	F05	Pemain melakukan perintah suara "*use mouse*" untuk memilih kendali mouse.	Button *use mouse* melakukan animasi membesar lalu aplikasi menampilkan loading screen.	Button *use mouse tidak* melakukan animasi membesar, sehingga aplikasi juga tidak menampilkan

					loading screen.
			Pemain melakukan perintah suara "*use kinect*" untuk memilih kendali kinect.	Button *use kinect* melakukan animasi membesar lalu aplikasi menampilkan loading screen.	Button *use kinect* tidak melakukan animasi membesar sehingga aplikasi juga tidak menampilkan loading screen.
6	Melihat *loading screen.*	F06	Pemain memilih kendali permainan.	Aplikasi menampilkan loading screen setelah kendali permainan terpilih dan aplikasi menampilkan layar cara bermain setelah *loading screen* berakhir.	Aplikasi tidak menampilkan loading screen setelah kendali permainan terpilih, sehingga aplikasi juga tidak menampilkan layar cara bermain setelah *loading screen* berakhir.

7	Melihat cara bermain.	F07	Pemain memilih kendali permainan.	Aplikasi menampilkan cara bermain setelah loading screen selesai.	Aplikasi tidak menampilkan cara bermain setelah loading screen selesai.
8	Memainkan Permainan.	F08	Pemain melakukan perintah suara "*let's play*" pada layar cara bermain.	Aplikasi masuk ke *in game state* dan menampilkan HUD *screen.*	Aplikasi tidak masuk ke *in game state* sehingga tidak menampilkan HUD *screen.*
9	Melihat pause menu.	F09	Pemain melakukan perintah suara "*let's play*" pada *in game state*.	Aplikasi menghentikan sementara permainan yang berlangsung lalu menampilkan menu pause yang memberikan pilihan untuk kembali ke permainan atau keluar dari game.	Aplikasi tidak menghentikan sementara permainan yang berlangsung sehingga tidak menampilkan menu pause yang memberikan pilihan untuk kembali ke permainan atau keluar dari game.

10	Menggerakkan kursor tangan.	F10	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 pemain pada arah sumbu X dan Y.
11	Melihat nyawa pemain.	F11	Pemain memainkan permainan dan masuk *in game state*.	Aplikasi dapat menampilkan informasi sisa nyawa pemain pada HUD *screen*.	Aplikasi tidak dapat menampilkan informasi sisa nyawa pemain pada HUD *screen*.
12	Melihat skor pemain.	F12	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*.
13	Melihat sisa waktu.	F13	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*.

14	Melihat gambar soal.	F14	Pemain memainkan permainan dan masuk *in game state*.	Aplikasi dapat menampilkan gambar soal secara acak pada HUD *screen.*	Aplikasi tidak dapat menampilkan gambar soal secara acak pada HUD *screen.*
15	Melihat jawaban pemain.	F15	Pemain memainkan permainan dan masuk *in game state*.	Aplikasi dapat menampilkan jawaban pemain pada HUD *screen.*	Aplikasi tidak dapat menampilkan jawaban pemain pada HUD *screen.*
16	Melihat target item.	F16	Pemain memainkan permainan dan masuk *in game state*.	Aplikasi dapat menampilkan target item karakter atau bom pada *game area*.	Aplikasi tidak dapat menampilkan target item karakter atau bom pada *game area*.
17	Menghancurkan target item.	F17	Pemain mengarahkan kursor pemain menyentuh *target item* karakter pada *in game state.*	Aplikasi menghilangkan *target item* karakter yang bertabrakan dengan kursor tangan player.	Aplikasi tidak menghilangkan *target item* karakter yang bertabrakan dengan kursor tangan player.
			Pemain mengarahkan kursor pemain menyentuh	Aplikasi menghilangkan *target item* bom yang	Aplikasi tidak menghilangkan *target item* bom yang

			target item bom pada *in game state.*	bertabrakan dengan kursor tangan player lalu mengurangi satu nyawa pemain.	bertabrakan dengan kursor tangan player sehingga tidak mengurangi satu nyawa pemain.
18	Mengakhiri aplikasi.	F18	Pemain melakukan perintah suara ”*exit*” pada menu utama.	Aplikasi game Slash Word berhenti atau keluar.	Aplikasi game Slash Word tidak berhenti atau tidak keluar.

Sumber : Pengujian dan Analisis

6.1.3.2. Hasil Pengujian Validasi

Hasil pengujian validasi dijelaskan pada tabel 6.28.

Tabel 6.28. Hasil Uji Validasi.

No.	Kasus Uji	Objek Uji	Prosedur dan Input	Kondisi yang diharapkan	Hasil
1	Melihat menu utama.	F01	Memulai *running* atau eksekusi *game* Slash Word.	Aplikasi dapat menampilkan halaman menu utama.	Valid
2	Melihat *high score*.	F02	Pemain memilih menu *high score* dengan perintah suara “*high score*”	Aplikasi dapat menampilkan halaman *high score*.	Valid

			pada menu utama.		
3	Memulai permainan.	F03	Pemain melakukan perintah suara "*let's play*" pada menu utama.	Aplikasi dapat menampilkan halaman memilih level permainan.	Valid
4	Memilih level permainan.	F04	Pemain melakukan perintah suara "*easy*" untuk memilih level *easy* pada menu level permainan.	Button *easy* melakukan animasi membesar lalu aplikasi menampilkan halaman memilih kendali permainan.	Valid
			Pemain melakukan perintah suara "*medium*" untuk memilih level *medium* pada menu level permainan.	Button *medium* melakukan animasi membesar lalu aplikasi menampilkan halaman memilih kendali permainan.	Valid

			Pemain melakukan perintah suara "*hard*" untuk memilih level *hard* pada menu level permainan.	Button *hard* melakukan animasi membesar lalu aplikasi menampilkan halaman memilih kendali permainan.	Valid
5	Memilih kendali permainan.	F05	Pemain melakukan perintah suara "*use mouse*" untuk memilih kendali mouse.	Button *use mouse* melakukan animasi membesar lalu aplikasi menampilkan loading screen.	Valid
			Pemain melakukan perintah suara "*use kinect*" untuk memilih kendali kinect.	Button *use kinect* melakukan animasi membesar lalu aplikasi menampilkan loading screen.	Valid
6	Melihat *loading screen.*	F06	Pemain memilih kendali permainan.	Aplikasi menampilkan loading screen setelah kendali permainan	Valid

				terpilih dan aplikasi menampilkan layar cara bermain setelah *loading screen* berakhir.	
7	Melihat cara bermain.	F07	Pemain memilih kendali permainan.	Aplikasi menampilkan cara bermain setelah loading screen selesai.	Valid
8	Memainkan Permainan.	F08	Pemain melakukan perintah suara "*let's play*" pada layar cara bermain.	Aplikasi masuk ke *in game state* dan menampilkan HUD *screen.*	Valid
9	Melihat pause menu.	F09	Pemain melakukan perintah suara "*let's play*" pada *in game state.*	Aplikasi menghentikan sementara permainan yang berlangsung lalu menampilkan menu pause yang memberikan pilihan untuk	Valid

				kembali ke permainan atau keluar dari game.	
10	Menggerakkan kursor tangan.	F10	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
11	Melihat nyawa pemain.	F11	Pemain memainkan permainan dan masuk *in game state*.	Aplikasi dapat menampilkan informasi sisa nyawa pemain pada HUD *screen*.	Valid
12	Melihat skor pemain.	F12	Pemain memainkan permainan dan masuk *in game state*.	Aplikasi dapat menampilkan informasi skor pemain pada HUD *screen*.	Valid
13	Melihat sisa waktu.	F13	Pemain memainkan permainan dan masuk *in game state*.	Aplikasi dapat menampilkan informasi sisa waktu permainan pada HUD *screen*.	Valid

14	Melihat gambar soal.	F14	Pemain memainkan permainan dan masuk *in game state*.	Aplikasi dapat menampilkan gambar soal secara acak pada HUD *screen.*	Valid
15	Melihat jawaban pemain.	F15	Pemain memainkan permainan dan masuk *in game state*.	Aplikasi dapat menampilkan jawaban pemain pada HUD *screen.*	Valid
16	Melihat target item.	F16	Pemain memainkan permainan dan masuk *in game state*.	Aplikasi dapat menampilkan target item karakter atau bom pada *game area*.	Valid
17	Menghancurkan target item.	F17	Pemain mengarahkan kursor pemain menyentuh *target item* karakter pada *in game state.*	Aplikasi menghilangkan *target item* karakter yang bertabrakan dengan kursor tangan player.	Valid
			Pemain mengarahkan kursor pemain menyentuh *target item* bom pada *in game state.*	Aplikasi menghilangkan *target item* bom yang bertabrakan dengan kursor tangan player	Valid

				lalu mengurangi satu nyawa pemain.	

Sumber : Pengujian dan Analisis

6.1.4. Pengujian Kinerja

Pengujian kinerja *game* Slash Word dilakukan dengan melihat FPS (*Frame Per Second*) yang dihasilkan oleh *game*. Pengujian FPS pada *game* Slash Word dilakukan pada tujuh komputer dengan spesifikasi berbeda. Pengujian kinerja dilakukan dengan melihat FPS yang dihasilkan pada setiap halaman. Adapun halaman yang digunakan dalam pengujian ini adalah main menu, *high score*, halaman pemilihan level, pemilihan kendali permainan, dan halaman *In Game*. Hasil pengujian kinerja *game* Slash Word ditunjukkan pada tabel 6.29. Tabel 6.30 menunjukan pengaruh FPS terhadap kinerja game. Gambar 6.14 menunjukan grafik FPS *average* dari tiap halaman *game* Slash Words yang dihasilkan dari pengujian kepada tujuh komputer dengan spesifikasi berbeda.

Tabel 6.29. Hasil pengujian kinerja dengan melihat FPS

No	Spesifikasi Sistem	FPS				
		Main Menu	High Score	Menu Level	Menu Kontrol	In Game
1	Core I3 2.10 GHz, RAM 2G, VGA Shared Memory Up To 1G Sandy Bridge.	39.8	35.6	39.8	40.6	6
2	Core I3 2.10 GHz, RAM 4G, VGA Shared Memory Up To 1G Sandy Bridge.	90.6	92.2	99.8	105	34.6
3	Dual Core 2.0 GHz, RAM 2.5G, VGA Dedicated Memory 256 MB, Memory	89	89	94	90	47.8

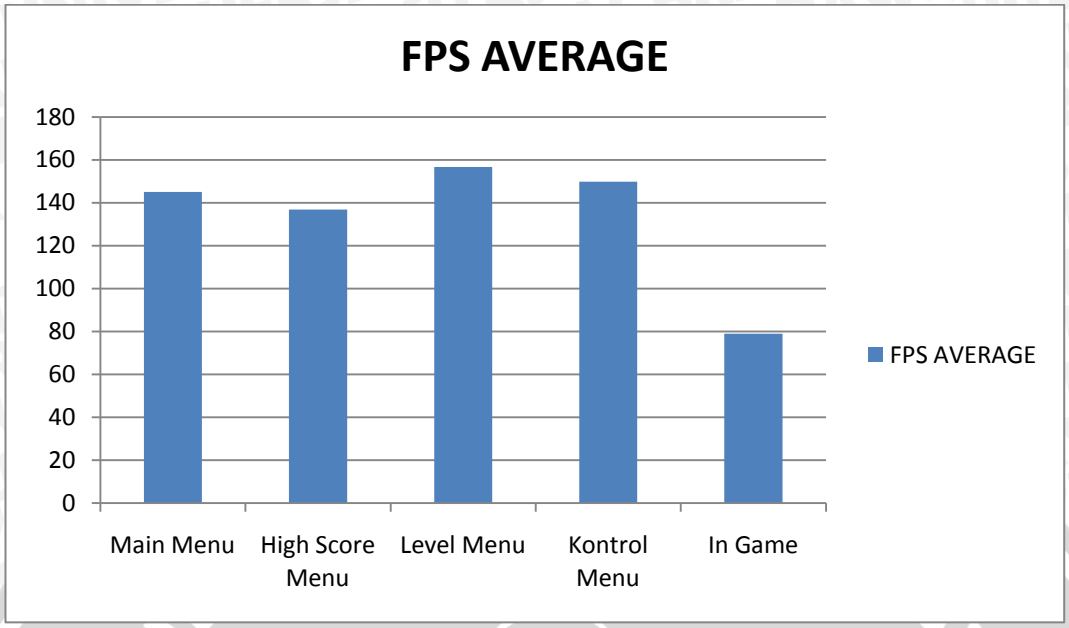
	Bus Width 64Bit GDDR3.					
4	Core 2 Duo 2.26 GHz, RAM 4G, VGA Dedicated Memory 128 MB, Memory Bus Width 64Bit GDDR3.	180.4	153	171	165.6	56.2
5	Quad Core A6 1.4GHz, RAM 8G, VGA Dedicated Memory 512MB, Memory Bus Width 64Bit GDDR3.	194.6	190.4	221.6	221	116.6
6	Core I5 2.4 GHz, RAM 4G, VGA Dedicated Memory 1G, Memory Bus Width 64Bit GDDR3.	190	200	219	212	138.2
7	Intel Core I5 2.5 GHz, Ram 4G, VGA Dedicated Memory 2G, Memory Bus Width 128Bit GDDR3.	231	197.4	251.6	215	153
FPS AVERAGE		**145.05**	**136.8**	**156.68**	**149.88**	**78.91**

Sumber: Pengujian dan Analisis

Tabel 6. 30. Pengaruh FPS Terhadap kinerja *Game*

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

Sumber: Pengujian dan Analisis



Gambar 6.14. Grafik FPS AVERAGE Halaman *Game* Slash Word

Sumber : Pengujian dan Analisis

6.2. Analisis

Analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian *game* Slash Word yang telah dilakukan. Analisis dilakukan terhadap hasil pengujian di setiap tahap pengujian. Proses analisis yang dilakukan meliputi analisis hasil pengujian unit, analisis hasil pengujian integrasi, analisis hasil pengujian validasi, dan analisis hasil pengujian kinerja.

6.2.1. Analisis Pengujian Unit

Hasil analisa yang didapatkan dari pengujian unit yaitu :

1. Berdasarkan kesesuaian antara hasil pengujian tiap unit dengan output unit yang diharapkan pada *game* Slash Word, maka dapat diambil kesimpulan bahwa unit modul dari program sudah sesuai dengan output yang diharapkan.
2. Berdasarkan hasil perhitungan *cyclomatic complexity* dari tiap *flow grap* kode unit, kode unit yang menghasilkan jalur kasus uji terbanyak adalah kode operasi `sortingScore()` sejumlah empat kasus uji. Jumlah kasus uji kode `sortingScore()` sebagian besar dipengaruhi dari logika

perulangan dan seleksi kondisi, karena kode ini berisi proses untuk mengurutkan skor dalam list skor.

6.2.2. Analisis Pengujian Integrasi

Hasil analisa yang didapatkan dari pengujian integrasi yaitu :

1. Berdasarkan kesesuaian antara hasil pengujian integrasi dengan output integrasi yang diharapkan pada *game* Slash Word, maka dapat diambil kesimpulan bahwa modul integrasi dari program sudah sesuai dengan output yang diharapkan.
2. Berdasarkan hasil perhitungan *cyclomatic complexity* dari tiap *flow grap* kode integrasi, kode integrasi yang menghasilkan jalur kasus uji terbanyak adalah kode operasi `onPointUpdate()` sejumlah sembilan kasus uji. Jumlah kasus uji kode `onPointUpdate()` sebagian besar dipengaruhi dari logika seleksi kondisi, karena kode ini berisi proses untuk menerima dan memfilter input lokasi koordinat sumbu X dan sumbu Y dari pergerakan telapak tangan pemain yang diterima oleh sensor Kinect. Koordinat sumbu X dan sumbu Y yang diterima oleh sensor Kinect akan difilter atau dinormalisasi sesuai dengan batas koordinat area permainan, sehingga kursor tangan tidak akan keluar area permainan.

6.2.3. Analisis Pengujian Validasi

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

6.2.4. Analisis Pengujian Kinerja

Hasil analisa yang didapatkan dari pengujian integrasi yaitu :

1. Proses analisa terhadap hasil pengujian peforma dilakukan dengan melihat FPS yang dihasilkan pada halaman main menu, *high score*, halaman pemilihan level, pemilihan kendali, dan halaman *in game*.

2. Berdasarkan hasil pengujian peforma yang telah dilakukan pada tujuh komputer dengan spesifikasi yang berbeda, didapatkan rata - rata FPS terendah dari tujuh komputer ada pada halaman *in game* yaitu sebesar 78.91 FPS, hal ini disebabkan karena pada halaman *in game* menampilkan banyak komponen gambar bergerak dan proses algoritma alur *game* diproses di halaman ini.
3. Proses analisa terhadap nilai FPS tiap halaman *game* dari setiap komputer uji, nilai FPS halaman *in game* pada komputer uji 1 berada di bawah standar minimal FPS untuk dapat menjalankan sebuah game dengan baik. Nilai FPS dibawah 30 FPS akan membuat permainan Slash Word berjalan patah – patah.
4. Spesifikasi komputer yang disarankan untuk dapat menjalankan *game* Slash Word adalah :
    a. Prosesor dua inti 2.0 GHz atau yang lebih cepat.
    b. Memori minimal 2.5 GB RAM.
    c. VGA *Dedicated* dengan memori minimal 256 MB.

BAB VII
PENUTUP

7.1. Kesimpulan

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

1. *Gameplay* dari game Slash Word yang menerapkan teknik interaksi natural adalah pemain menggerakan telapak tanganya kearah koordinat sumbu X dan sumbu Y untuk menggerakkan kursor tangan pemain hingga mengenai huruf yang tidak terpakai dalam pembentukan jawaban soal.
2. Materi pembelajaran kosa kata bahasa Inggris dalam *game* Slash Word adalah dengan menebak nama – nama dari gambar yang muncul di layar menggunakan bahasa Inggris.
3. Perancangan *game Slash Word* terdiri dari *game concept design* dan *technical design.*
4. Implementasi *game Slash Word* menggunakan jMonkey Engine 3 dan implementasi sensor kinect dalam game menggunakan *library* Kinect untuk jMonkey Engine 3 yang dibuat oleh Glauco Márdano dengan basis API Openni 1.5.2.23 dan NITE 1.5.2.21.
5. Pengujian unit *game* Slash Word dengan metode *White box* menghasilkan kesimpulan bahwa unit modul dari program sudah sesuai dengan output yang diharapkan dan kode unit dalam pengujian yang menghasilkan kasus uji terbanyak adalah kode operasi `sortingScore()`, yaitu sebanyak empat kasus uji.
6. Pengujian integrasi *game* Slash Word dengan metode *White box* menghasilkan kesimpulan bahwa integrasi modul dari program sudah sesuai dengan output yang diharapkan dan kode integrasi dalam pengujian yang menghasilkan kasus uji terbanyak adalah kode operasi `onPointUpdate()`, yaitu sebanyak sembilan kasus uji.
7. Berdasarkan hasil pengujian validasi menggunakan metode *blackbox testing*, didapatkan keseluruhan fungsional aplikasi permainan dapat berjalan sesuai daftar kebutuhan yang telah dibuat dan dimodelkan dalam *use case*.

8. Pengujian kinerja *game* Slash Word menghasilkan kesimpulan bahwa rata – rata FPS terendah ada pada halaman *in game* dan *game* Slash Word dapat berjalan optimal ketika *game* berjalan pada komputer atau *laptop* yang yang dapat menghasilkan *frame per second* lebih besar atau sama dengan 30 FPS ketika memainkan *game* Slash Word.

7.2. Saran

Saran untuk pengembangan game Slash Word lebih lanjut antara lain :

1. Dapat dilakukan pengembangan game Slash Word dengan menggunakan Kinect API yang berbeda seperti Microsoft Kinect SDK.
2. Penambahan fitur *Voice Answer* dengan memanfaatkan *microphone* yang sudah *include* di dalam kinect.
3. Penambahan fitur permainan 2 player *split screen* menggunakan Kinect API yang telah mendukung *multi user detection* seperti OpenNi dan Microsoft KinectSDK.

DAFTAR PUSTAKA

[ALH-03] Alhir, Sinan Si. 2003. *Learning UML*. O'Reilly Media, Inc.

[BET-03] Bethke, Erik. 2003. *Game Development and Production.* Wordware Publishing, Inc.

[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.

[DAV-05] Davison, Andrew. 2005. *Killer Game Programming in Java.* O'Reilly Media, Inc.

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

[HAP-11] Hapsari, Iriani I & Suminar, Dewi R. 2011. *Efektifitas Ludo Words Game (LWG) terhadap Peningkatan Kosakata Bahasa Inggris pada Anak, Studi Kasus Pada Siswa Kelas IV SD Muhammadiyah 4 Pucang*. Jurnal Universitas Airlangga.

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

[HEN-10] Henry, Samuel. 2010. *Cerdas dengan Game.* PT Gramedia Pustaka Utama.

[HUC-11] Huck, A.S., 2011. *Exploring Gesture Based Interaction and Visualizations for Supporting collaboration*. Thesis, Swedia : Bachelor Thesis, Linnaeus University.z

[LEW-02] Lewis, Michael & Jacobson, Jeffrey., Januari 2002. "Game Engines in Scientific Research". **Communications Of The Acm**, No. 1 Vol. 45, h. 27 – 31.

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

[MAH-10] Mahtarami, Affan & Ifansyah, M.N. 2010. *Pengembangan Game Pembelajaran Otomata Finit*. Makalah disajikan pada Seminar

Nasional Informatika, Jurusan Teknik Informatika Universitas Islam Indonesia, Yogyakarta, 22 Mei.

[MCM-10] McMahan, R.P., Alon A.J.D., Lazeem, Shaimaa *et al*. 2010. "Evaluating Natural Interaction Techniques in Video Games". IEEE Symposium on 3D User Interfaces, USA, Massachusetts.

[MUL-10] Muller-Clostermann, Bruno, Echtle, Klaus, Rathgeb, E.P. (Eds). 2010, *Measurement, Modelling, andEvaluation of Computing Systems and Dependability and Fault Tolerance*. Germany : Springer-Verlag Berlin Heidelberg.

[NOV-12] Novitasari, Denny R. 2012. *Pembangunan Media Pembelajaran Bahasa Inggris Untuk Siswa Kelas 1 Pada Sekolah Dasar Negeri 15 Sragen*. SPEED Jurnal FTI UNSA Edisi Web, 12 Februari.

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

[PIU-11] Piumsomboon, Thammathip, Clark, Adrian & Billinghurst , Mark. 2011. "Physically-based Interaction for Tabletop Augmented Reality Using a Depth-sensing Camera for Environment Mapping", Proc. Image and Vision Computing New Zealand, Auckland, h. 161-166.

[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

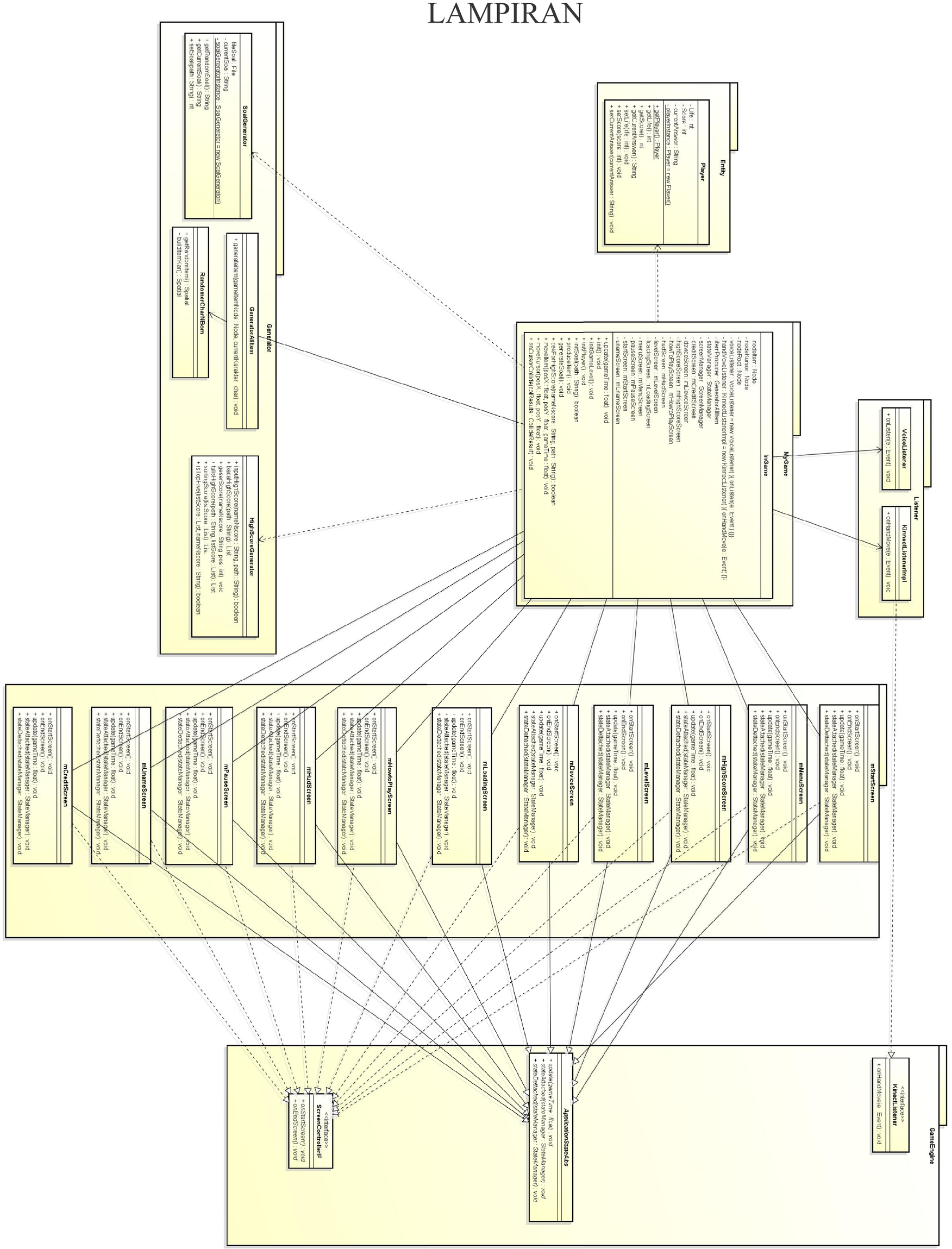
[ROL-03] Rollings, Andrew & Adams, Ernest. 2003. *Andrew Rollings and Ernest Adams on Game Design*. New Riders Publising.

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

[SUN-11] Sung, Kelvin. Februari 2011. "Recent Videogame Console Technologies". **IEEE Computer Society**, h. 91.

[SUP-08] Supendi, Pepen & Nurhidayat. 2008. *FUN GAME*. Penebar Swadaya, Jakarta.

[YOO-11] Yoo, Min-Joon, Beak Jin-Wook & Lee, In-kwoon. 2011, “Creating Musical Expression using Kinect”, Proceedings of the International Conference on New Interfaces for Musical Expression, Norway, Oslo, h. 324.



Gambar 4.19. Rancangan Diagram Kelas *Game* Slash Word