RANCANG BANGUN *GAME* EDUKASI BIOLOGI SMP PENGENALAN SISTEM PENCERNAAN MANUSIA

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
INDRAHIDAYATI NUR ROHMAH
NIM: 115090607111013

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

Rancang Bangun Game Edukasi Biologi SMP Pengenalan Sistem Pencernaan Manusia

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Indrahidayati Nur Rohmah
NIM: 115090607111013

Skripsi ini telah diuji dan dinyatakan lulus pada
14 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Eriq Muh. Adams J., S.T, M.Kom	Wibisono Sukmo Wardhono, S.T, M.T
NIP: 19850410 201212 1 001	NIK: 201008 820404 1 001

Mengetahui
Ketua Program Studi Informatika/Ilmu Komputer

Marji, Drs., M.T
NIP: 19670801 199203 1 001

PERNYATAAN ORISINALITAS

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 disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

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

Malang, 26 Januari 2016

Indrahidayati Nur Rohmah

NIM: 115090607111013

KATA PENGANTAR

Puji syukur saya ucapkan kepada Allah SWT yang selalu melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Skripsi yang berjudul, "Rancang Bangun Game Edukasi Biologi SMP Pengenalan Sistem Pencernaan Manusia". Skripsi ini diajukan sebagai salah satu syarat untuk mendapatkan gelar sarjana S-1 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Malang.

Melalui kesempatan ini, penulis menyampaikan rasa hormat dan terima kasih penulis yang sebesar-besarnya kepada pihak yang telah memberikan bantuan lahir maupun batin selama penulisan skripsi ini. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Bapak Eriq M. Adams J. ST, M.Kom dan Bapak Wibisono Sukmo Wardhono, ST, MT selaku dosen pembimbing Penulis. Terima kasih atas semua bimbingan, kritik serta saran, dan dorongan semangatnya.
2. Bapak Drs. Marji, M.Si. dan Issa Arwani, ST., MT. selaku Ketua dan Sekretaris Program Studi Informatika serta segenap Bapak/Ibu Dosen, Staff Administrasi dan Perpustakaan Program Studi Informatika Universitas Brawijaya.
3. Seluruh dosen Program Studi Informatika atas keikhlasan membagi ilmunya kepada penulis.
4. Semua Asisten Ka. Lab serta Laboran dari Laboratorium Program Studi Informatika yang telah memberikan banyak bantuan dan dukungan dalam menyelesaikan skripsi ini.
5. Kepada orang tua penulis dan seluruh keluarga yang senantiasa tiada henti hentinya memberikan do'a demi terselesainya skripsi ini.
6. Kakak saya Indrajayanti Ratnaningsih yang telah memberikan ide skripsi, bimbingan, serta bantuannya.
7. Teman – teman Computer Science 2011 (Ilkomp Uno) dan Teknik Informatika UB yang selalu memberikan semangat.
8. Eureka Kurniasari yang telah membantu dalam pembuatan *assets* Skripsi ini. Fadel Trivandi atas bantuannya dalam *coding.* Dan juga kepada Evita Devi, Rasuna Rizky, dan Harinda Bonita yang selalu memberikan semangat dan dorongan untuk menyeleseikan Skripsi.
9. Seluruh civitas akademis "SMP PGRI 1 Buduran Sidoarjo" yang telah mengizinkan penulis melakukan penelitian.
10. Teman-teman LKI AMD yang telah memberikan semangat.
11. Teman – teman di laboratorium game yang telah memberikan masukan dalam pengerjaan Skripsi ini.
12. Serta semua pihak yang namanya tidak bisa penulis sebutkan satu persatu. Terima kasih atas do'a dan dukungannya.

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

Malang, 26 Januari 2016

Penulis

indra.arrohmah@gmail.com

ABSTRAK

Indrahidayati Nur Rohmah. 2015. : Rancang Bangun Game Edukasi Biologi SMP Pengenalan Sistem Pencernaan Manusia. Skripsi Program Studi Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing: Eriq M. Adams J., ST., Mkom. dan Wibisono Sukmo Wardhono, ST., MT

Sistem pencernaan makanan pada manusia terdiri dari beberapa organ, secara berurutan dari awal sampai akhir adalah rongga mulut, esophagus/ kerongkongan, lambung, usus halus, usus besar, rectum, anus. Fungsi dari sistem pencernaan manusia ini adalah untuk menerima makanan dari luar dan mempersiapkannya untuk diserap oleh tubuh dengan jalan proses pencernaan (penguyahan, penelanan, dan pencampuran) dengan enzim zat cair yang terbentang mulai dari mulut sampai anus. Metode pembelajaran sangat menentukan hasil belajar dari para siswa. Pada umumnya metode pembelajaran yang digunakan oleh guru adalah berupa ceramah dan diskusi. Menurut penelitian yang dilakukan di SMP sebanyak 46% koresponden menyatakan bahwa pelajaran biologi sulit, dan sebesar 86% menyatakan yang membuat pelajaran ini sulit adalah karena banyak istilah yang sulit dimengerti. Sebanyak 36% koresponden menyatakan bahwa materi sistem pencernaan manusia merupakan materi yang paling sulit dalam pelajaran biologi. *Game* edukasi merupakan salah satu alat bantu bagi siswa untuk lebih mudah mengenal materi dalam suatu proses pembelajaran. *Game* edukasi yang dibuat difokuskan untuk mengenalkan nama zat dan enzim yang berpengaruh dalam proses pencernaan. Dari hasil pengujian terhadap siswa SMP menunjukkan bahwa *game* edukasi ini dapat diterima dengan baik.

Kata kunci: *Game,* Edukasi, Sistem pencernaan manusia

ABSTRACT

Human digestive system consists of mouth, oesophagus, stomach, small intestine, large intestine, rectum, and anus. The function of human digestive system is accepting foods from outside and preparing it to be absorbed by body with several digestive processes (chewing, swallowing, and mixing) with the enzyme that located in mouth until anus. Learning method that used will determine students' learning outcome. In general learning methods that used by teachers are lecture and discussion. According to research that have been done in Junior High School found that 46% of the correspondences said that biology is hard, and 86% said that too many foreign term makes it harder to understand, 36% said that human digestive system is the hardest lesson in biology. Education game is one of tool that will help student to know more about the lesson. Education game will be focused at introduce enzyme's name and substance name in the digestive system. From the results of the testing of junior high school students showed that the educational game is well received.

Keyword: Game, Education, Human digestive system

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Metode pembelajaran sangat menentukan hasil belajar dari para siswa. Hasil belajar belum optimal, jika dalam proses pembelajaran model yang dianut para guru didasarkan pada asumsi bahwa "pengetahuan dapat dipindahkan secara utuh dari pikiran guru ke pikiran siswa" (Citrawathi, 2006). Pada umumnya metode pembelajaran yang digunakan oleh guru adalah berupa ceramah dan diskusi.

Salah satu bab di dalam materi IPA terpadu kelas delapan adalah sistem pencernaan. Sistem pencernaan atau sistem gastroinsestinal adalah sebuah sistem dari organ-organ di dalam tubuh mengambil makanan, yang mengambil intisari, menyerap nutrisi dan energi, serta membuang sisa dari makanan tersebut (Carlson, 2004). Dalam bab ini siswa akan belajar mengenai perjalanan kandungan makanan yang masuk ke dalam tubuh manusia. Pada bab ini banyak peserta didik yang merasa kesulitan untuk menghafalkan nama-nama bagian tubuh, enzim-enzim yang diproduksi dan komponen lain yang terdapat di dalam proses pencernaan. Ditambah dengan metode pembelajaran yang dilakukan guru selama ini belum efektif jika dilihat dari hasil belajar dan pengalaman belajar yang didapat oleh siswa. Siswa cenderung hanya menghafal fakta dan konsep, akibatnya mereka merasa pelajaran biologi membosankan, dan kurang menarik (Adnyana & Citrawathi, 2008).

Menurut penelitian yang dilakukan di SMP PGRI 1 Buduran Sidoarjo sebanyak 46% koresponden menyatakan bahwa pelajaran biologi sulit, dan sebesar 86% menyatakan yang membuat pelajaran ini sulit adalah karena banyak istilah yang sulit dimengerti. Sebanyak 36% koresponden menyatakan bahwa materi sistem pencernaan manusia merupakan materi yang paling sulit dalam pelajaran biologi. Pembelajaran dengan metode ceramah masih menjadi metode yang sering digunakan oleh pengajar, dengan memperoleh persentase sebanyak 54%. Sayangnya, metode pembelajaran dengan menggunakan permainan tidak mendapat satu suara pun, padahal sebanyak 34% koresponden menyatakan bahwa mereka ingin suasana pembelajaran yang menyenangkan dan menggunakan bantuan multimedia seperti permainan atau video agar proses pelajaran mudah dipahami.

Perkembangan teknologi yang pesat juga mempengaruhi dalam dunia pendidikan. Teknologi yang dapat dipakai dalam pendidikan seperti *game* edukasi. Dengan bermain *game* dapat membuat siswa dalam meningkatkan kemampuan belajar karena akan tercipta emosional yang akan membuatnya terikat dengan *game* tersebut dan mereka akan memiliki keinginan untuk menyeleseikan permainan dengan baik. Hal ini sulit didapat dari metode pembelajaran yang lainnya (Ault, 2004). *Game* edukasi merupakan multimedia dengan sistem penyajian yang menggunakan berbagai jenis bahan ajar yang membentuk satu kesatuan atau paket yang berupa perangkat lunak dalam proses pembelajaran

(Warsita, 2008). Dengan menggunakan *game* edukasi ini, siswa dapat memahami pelajaran dengan lebih menyenangkan.

Berdasarkan uraian di atas maka penulis mengembangkan *game* edukasi biologi sistem pencernaan manusia dengan judul "Nutriman Evolution". Dengan membuat *game* edukasi yang menarik dan menghibur dapat membantu mereka dalam memepelajari materi pelajaran yang sulit. Dengan demikian, membangun *game* edukasi tentang sistem pencernaan manusia yang dapat dioperasikan di telepon pintar dapat membantu para peserta didik untuk meningkatkan kemampuan pemahaman terhadap materi sistem pencernaan pada manusia.

1.2 Rumusan masalah

Berdasarkan paparan latar belakang tersebut, maka rumusan masalah adalah sebagai berikut:

1. Bagaimana merancang *gameplay* "Nutriman Evolution" agar menjadi permainan yang menarik dan edukatif?
2. Bagaimana mengimplementasikan *gameplay* "Nutriman Evolution" agar menjadi permainan yang menarik dan edukatif?
3. Bagaimana mengetahui apakah *game* "Nutriman Evolution" berjalan sesuai dengan kebutuhan dan dapat diterima menjadi media pendamping pembelajaran yang menarik?

1.3 Tujuan

Tujuan umum dari topik skripsi ini adalah untuk membangun game edukasi biologi SMP Pengenalan Sistem Pencernaan Manusia.

Sedangkan tujuan khusus dari pengerjaan topik skripsi ini adalah merancang dan mengimplementasikan *gameplay* dari *game* edukasi biologi sistem pencernaan manusia. Selain itu juga untuk mengetahui apakah fungsionalitas dari *game* sudah terpenuhi serta untuk mengetahui apakah *game* edukasi biologi sistem pencernaan manusia SMP tidak hanya menyenangkan dan menarik untuk dimainkan namun juga membantu siswa SMP dalam mengenal sistem pencernaan manusia.

1.4 Manfaat

Manfaat yang dapat diperoleh dari pengerjaan skripsi ini adalah:

1. Menyediakan *game* yang menarik dan juga memberikan ilmu tentang sistem pencernaan manusia.
2. Meningkatkan minat belajar siswa terhadap mata pelajaran biologi terutama sistem pencernaan manusia.

1.5 Batasan masalah

Untuk menghindari adanya kemungkinan semakin berkembangnya masalah, maka penulisan laporan ini hanya menitikberatkan permasalahan pada beberapa hal, diantaranya adalah :

1. Bagian yang akan diimplementasikan ke dalam *game* adalah perubahan zat makanan di setiap bagian organ, dan enzim yang berpengaruh terhadap zat makanan. Organ yang akan diimplementasikan dalam *game* adalah mulut, lambung, usus 12 jari, dan usus penyerapan. Sedangkan untuk zat makanan adalah karbohidrat, protein, lemak, dan vitamin.
2. Pembuatan *game* edukasi menggunakan *game engine* Unity 3D versi 5.2 dengan bahasa pemrograman C#.
3. *Game* yang akan dibangun menggunakan model dan animasi 2D.
4. *Game* akan dioperasikan pada telepon pintar dengan sistem operasi Android.

1.6 Sistematika pembahasan

Untuk mencapai tujuan yang diharapkan, maka sistematika penulisan yang disusun dalam tugas akhir ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini merupakan dasar dari penyusunan skripsi ini yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

BAB II LANDASAN KEPUSTAKAAN

Bab ini berisi kajian pustaka, referensi atau sumber-sumber yang berhubungan dengan permasalahan dalam skripsi.

BAB III METODOLOGI

Bab ini menjelaskan bagaimana metodologi yang digunakan untuk perancangan, proses perancangan, dan proses implementasi *game* edukasi biologi SMP sistem pencernaan manusia.

BAB IV PERANCANGAN DAN IMPLEMENTASI

Bab ini menjelaskan proses dan hasil pengujian dari implementasi *game* edukasi biologi SMP sistem pencernaan manusia.

BAB V PENGUJIAN

Bab ini menjelaskan proses dan hasil pengujian dari implementasi *game* edukasi biologi SMP sistem pencernaan manusia.

BAB VI PENUTUP

Bab ini menjelaskan kesimpulan yang dapat diambil dari implementasi *game* edukasi biologi SMP sistem pencernaan manusia dan disertai saran sebagai acuan untuk pengembangan selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Game

Game menurut Kamus Besar Bahasa Indonesia berarti permainan. *Game* adalah sebuah aktivitas yang mempunyai aturan, konflik, tujuan, dan mengandung pembuatan keputusan (Schreiber, 2009). *Game* harus memiliki minimal seorang pemain dan memiliki kondisi kemenangan.

2.1.1 Element Game

Di dalam sebuah game terdapat elemen yang menyusunnya yaitu player, objective (tujuan), rules, resources dan resources management, game state, information, sequencing, player interaction, theme, dan game as system (Schreiber, 2009).

1. *Player*

Player merupakan pemain atau peserta dari sebuah *game*. Ada beberapa komponen yang harus diperhatikan di bagian ini, seperti jumlah *player*, bisa ditentukan dari awal (misal hanya diperbolehkan empat *player*), atau bisa juga memilih dalam variabel (misal dua sampai lima), apakah *player* bisa memasuki dan meninggalkan permainan ketika bermain, dan juga bagaimana pengaruhnya. Relasi antar *player* juga harus diperhatikan, apakah mereka tim atau individual.

2. *Objective*

Objective merupakan tujuan dari *game*. Apa yang ingin dilakukan oleh *player* merupakan *objective*. *Objective* bisa dilakukan pada awal mendesain suatu *game* karena ketika *objective* sudah jelas maka hal-hal lain dapat terdefinisikan dengan sendirinya.

3. *Rules*

Rules merupakan aturan di dalam *game* yang dapat dibagi menjadi tiga ketegori, yaitu: *setup* (hal yang dilakukan oleh *player* pada saat awal permainan), *progression of play* (apa yang terjadi ketika permainan berlangsung), dan *resolution* (kondisi apa yang menyebabkan permainan berakhir, dan bagaimana hasil yang ditentukan berdasarkan kondisi permainan). R*ules* dapat berjalan secara otomatis, atau tergantung dengan pilihan atau aksi dari *player* yang dilakukan dalam permainan dan dapat berpengaruh terhadap kondisi permainan.

4. *Resources* dan *Resources Management*

Resources dan *resources management* merupakan sumber daya yang eksplisit (secara jelas diketahui oleh *player*). Hal yang harus didefinisikan dengan jelas oleh seorang *game designer* di bagian *resource* adalah jenis sumber daya yang bisa dikontrol oleh *player*, dan bagaimana sumber daya bisa memanipulasi permainan.

5. *Game State*

Game state merupakan kondisi dimana semua elemen *game* berkumpul, termasuk kondisi terkini dari sumber daya *player* dan semua hal pada suatu waktu tertentu. Dalam perkembangan video *games*, *game state* harus didefinisikan, karena *game state* ini mengandung semua data yang harus dapat terakses oleh komputer.

6. *Information*

Information merupakan *game state* yang dapat diketahui oleh setiap *player*. Mengubah jumlah *information* yang dapat diketahui oleh *player* memiliki efek yang drastic di dalam *game*. *Information* ada yang bersifat publik dan privat. Penggunaannya dapat disesuaikan dengan *game* yang akan dikembangkan. Misalnya dalam *game* Poker setiap *player* hanya bisa mengetahui kartu miliknya sendiri saja. Sedangkan dalam *game* Chess and Go semua *player* dapat mengetahui informasi yang sama. Untuk game satu lawan banyak *player* sering digunakan hanya satu *player* memiliki informasi pribadinya sendiri, sedangkan *player* lain tidak memiliki, contohnya seperti *game* Scotland Yard. *Game* dapat memuat *information* yang tersembunyi bagi semua *player* seperti Clue and Sleuth yang memiliki *information* yang disembunyikan berupa kondisi kemenganan yang ditemukan oleh *player*.

7. *Sequencing*

Sequencing merupakan alur yang dilakukan oleh *player* untuk mengambil suatu aksi, dan alur permainan dari satu aksi ke aksi yang lain. Sebuah *game* dapat berjalan dengan berbeda tergantung dengan struktur urutan yang digunakan.

8. *Player Interaction*

Player interaction merupakan interkasi yang dapat dilakukan oleh masing – masing *player*. Hal yang perlu ditentukan adalah bagaimana *player* berinteraksi dengan *player* yang lain, dan bagaimana mereka saling berpengaruh satu dengan lainnya. Interaksi bisa berupa konflik atau penyerangan secara langsung, negosiasi, penukaran barang, dan berbagi informasi.

9. *Theme*

Theme di dalam sebuah *game* mengandung narasi, *backstory* atau latar belakang dari sebuah karakter di dalam *game*, serta *setting* atau tempat dan keadaan dimana sesuatu berada dan terjadi suatu kejadian di dalamnya. *Theme* tidak berpengaruh secara langsung terhadap *game*.

10. *Game as System*

Game as system merupakan gabungan dari elemen-elemen dalam *game* yang saling berhubungan menjadi suatu kesatuan yang kompleks. Karena mengubah satu elemen bisa berpengaruh terhadap elemen lain. Dalam sebuah *game* dapat mengandung beberapa *system*. Misalnya World of Warcraft memiliki *system* bertarung, komunikasi, *quest* dan lain sebagainya.

2.1.2 *Game* Edukasi

Game edukasi adalah *game* yang didesain untuk mengajari manusia terhadap subjek yang spesifik dan untuk mengajarinya sebuah keahlian. Seperti yang disadari oleh pengajar, dan orang tua akan kebutuhan psikologis dan keuntungan dari permainan di dalam pembelajaran, alat pembelajaran ini menjadi hal yang umum. *Game* merupakan permainan yang interaktif dan mengajari kita tujuan, aturan, adaptasi, penyeleseian masalah, interaksi, dan apa saja yang disajikan sebagai suatu cerita. Hal ini memberikan kita kebutuhan mendasar dari mempelajari dengan cara menyediakan kesenangan, motivasi, adrenalin, kreativitas, interaksi sosial dan emosi (Keesee, 2011).

2.1.3 Genre Game

Genre atau jenis *game* telah dipecah kedalam beberapa jenis *genre* dan *subgenre*. *Genre* mendeskripsikan gaya permainan dari sebuah *game*, seperti menembak, simulasi, balapan, dan lain sebagainya. *Game* dapat memiliki lebihdari satu *genre* (hybrid). Berikut merupakan penjelasan dari setiap *genre* dari *game* (Rogers, 2010).

a. *Action*: *games* yang memerlukan koordinasi tangan atau mata untuk bermain. Memiliki *subgenre* yaitu *action – adventure*, *action – arcade*, *platformer*, *stealth*, *fighting*, dan *beat 'em up*/*hack 'n' slash.*
b. *Shooter*: memiliki focus utama untuk menembakkan proyektil kepada musuh. Memiliki *subgenre* yaitu *first person shooter*, *shoot 'em up*, dan *third person shooter*.
c. *Adventure*: berfokus pada penyeleseian teka-teki, koleksi barang dan manajemennya. Memiliki *subgenre* yaitu *graphical adventure, role-playing game* (RPG), *massively multiplayer online role-playing game* (MMORPG), *survival*/*horror.*
d. *Construction/management*: *genre ini* memiliki *player* yang membangun dan memperluas suatu lokasi dengan sumber daya yang terbatas.
e. *Life simulation*: mirip dengan *genre* manajemen, namun ditambah dengan adanya simulasi kehidupan sehari-hari. Memiliki *subgenre* yaitu *pet simulation*.
f. *Music/rhythm*: g*ame* dengan tujuan untuk mencocokkan ritme atau irama musik.
g. *Party*: *genre* ini di-*design* khusus untuk beberapa *player* dan berbasiskan pada bermain secara kompetitif.
h. *Puzzle*: g*ame* yang berbasiskan pada logika dan melengkapi pola. *Genre* ini bisa berbentuk lambat, memiliki metode, atau menggunakan koordinasi tangan/mata.
i. *Sports*: *game* ini berbasiskan pada kompetisi olahraga, bisa tradisional atau ekstrim. Memiliki *subgenre* yaitu *sports management* yang berfokus pada memanajemen tim.

j. *Strategy*: *game* yang mengharuskan *player* untuk berfikir dan merencanakan strategi. Memiliki *subgenre* yaitu *real time strategy* (RTS), *turn – based,* dan *tower defense.*
k. *Vehicle simulation*: *game* yang berbentuk simulasi mengendarai kendaraan, bisa berbentuk mobil balap atau pesawat ruang angkasa. Penekanan dalam *genre* ini adal menghadirkan pengalaman berkendara yang nyata. Memiliki *subgenre* yaitu *driving*, dan *flying*.

2.2 Sistem Pencernaan Manusia

Sistem pencernaan makanan pada manusia terdiri dari beberapa organ, secara berurutan dari awal sampai akhir adalah rongga mulut, esophagus/ kerongkongan, lambung, usus halus, usus besar, rectum, anus. Fungsi dari sistem pencernaan manusia ini adalah untuk menerima makanan dari luar dan mempersiapkannya untuk diserap oleh tubuh dengan jalan proses pencernaan (penguyahan, penelanan, dan pencampuran) dengan enzim zat cair yang terbentang mulai dari mulut sampai anus (Tim, 2014). Berikut fungsi-fungsi organ pencernaan dan enzim yang terdapat di dalamnya:

a. Mulut: di dalamnya terdapat ludah, lidah dan gigi. Ludah dihasilkan oleh tiga pasang kelenjar ludah yaitu parotis, submandibularis, dan sublingualis. Ludah berfungsi untuk membasahi makanan, dan ludah mengandung enzim amylase yang berguna untuk mengubah karbohidrat menjadi amilum dan maltose.
b. Kerongkongan: saluran pencernaan yang menghubungkan rongga mulut dengan lambung. Di pangkal faring terdapat katup epiglottis yang berfungsi menutup pangkal tenggorokan atau saluran pernapasan saat kita senfang menelan makaan.
c. Lambung: terdiri dari tiga bagian yaitu, kardiak, fundus, pilorus. Di dalamnya terdapat getah lambung yang terdiri dari air, lendir, asam (HCl), enzim pepsinogen, dan enzim renin. HCl berfungsi untuk mematikan kuman yang terbawa bersama makanan, dan mengaktifkan pepsin dari pepsinogen. Pepsin berfungsi untuk mengubah protein menjadi pepton. Ranin berfungsi untuk mengendapkan protein susu (kasein) dari air susu.
d. Usus halus terbagi menjadi tiga bagian, yaitu:
    1. Usus dua belas jari: di dalamnya terdapat enzim amilase untuk mengubah amilum menjadi zat gula yang lebih sederhana, enzim tripsin untuk mengubah pepton menjadi asam amino, dan enzim lipase untuk mengubah lemak menjadi asam lemak dan gliserol.
    2. Usus kosong/usus tengah: terdapat proses pencernaan secara kimiawi dan pelumatan lebih sempurna. Pada bagian ini makanan terakhir kali dicerna, hasil akhirnya adalah karbohidrat menjadi monosakarida, disakarida, dan glukosa, protein menjadi asam amino, lemak menjadi asam lemak dan gliserol. Sedangkan vitamin dan mineral tidak pengalami pencernaan dan dapat langsung diserap.

3. Usus Penyerapan: makana yang telah dicerna menjadi sari-sari makanan diserap oleh diding usus penyerapan. Darah di dalam pembuluh kapiler usus menyerap sari-sari makanan dan dibawa ke hati, lelu ke jantung, dan diedarkan ke seluruh tubuh. Sisa makanan yang tidak diserap oleh dinding usus akan masuk ke usus besar.

e. Usus besar: memiliki fungsi utama untuk mengatur kadar air sisa makanan. Terdapat bakteri pembususk, yaitu *Escherichia coli* yang membantu pembusukan makanan menjadi feses yang nantinya akan menuju rectum, lalu anus.

2.3 MDA Framework

MDA merupakan sebuah pendekatan atau metode untuk men-*design game* dan memiliki tujuan untuk menjembatani jarak antara *game design* dan pengembang. *Game* dapat didefinisikan ke dalam *mechanic, dynamic,* dan *aesthethic* (Hunicke, et al., 2004).

a. *Mechanic* merupakan sinonim dari *rule* sebuah *game* atau batasan dalam pengoperasian *game* seperti bagaimana jalannya *game*, aksi apa yang bisa diambil oleh *player* dan apa efeknya terhadap *game state* ketika aksi-aksi itu diambil, kapan permainan berakhir. Itu semua didefinisikan oleh *mechanic.*
b. *Dynamic* mendeskripsikan bagaimana memainkan *game* ketika *rule* berjalan. Strategi apa yang muncul dari *rule* tersebut serta bagaimana interaksi antar *player*.
c. *Aesthetic* (di dalam MDA) tidak merujuk kepada elemen visual dari sebuah *game*, namun kepada pengalaman *player* terhadap *game* tersebut.

2.4 Paper Prototyping

Paper prototyping merupakan membuat *prototype* (rancangan) *game* ke dalam bentuk non-digital dengan media kertas, kardus, atau benda lainnya. Tujuan dari pembuatan *paper prototyping* adalah untuk mensimulasikan dengan mudah dan minim biaya dari suatu permainan tanpa pembuatan *game* itu sendiri secara digital. Membuat *prototype* secara digital dengan membuat program bisa memakan waktu lebih lama dan menghabiskan lebih banyak biaya daripada *paper prototype* (Schreiber, 2009). Berikut merupakan keuntungan menggunakan *paper prototyping*:

- Murah. Kebanyakan sistem dapat dibuat *prototype*-nya dengan menggunakan tidak lebih dari pensil dan kertas.
- Cepat. Tidak perlu berurusan dengan program, hanya membutuhkan menulis beberapa kata di atas kertas.
- Mudah untuk diubah. Jika kita tidak menyukai suatu komponen, kita bisa menghapus dan menggantinya dengan mudah.
- Tidak akan merasa bersalah ketika membuat *paper prototype* ketika ide yang sudah ditulis tidak digunakan.
- Kertas dapat digunakan hamper di seluruh sistem *gamplay*.

- Dengan membuat sesuatu yang dapat dimainkan, seorang *designer* dipaksa untuk membuat *design* dari sistem yang sebenarnya.

2.5 Unity 3D

Unity merupakan perangkat lunak untuk mengembangkan *game multiplatform* yang dikembangkan oleh Unity Technologies (Riccitiello, 2014). Unity digunakan untuk mengembangkan video *game* untuk PC, konsol, perangkat *mobile*, serta *websites*. Ketika pertama kali diluncurkan hanya untuk OS X, pada Apple's Worldwide Developers Conference di tahun 2005 (Brodkin, 2013). Secara default, Unity telah diatur untuk pembuatan game bergenre First Person Shooter (FPS), namun Unity juga bisa digunakan untuk membuat game bergenre Role Playing Game (RPG), dan Real Team Strategy (RTS). Selain itu Unity merupakan sebuah engine multiplatform yang memungkinkan game yang telah dibuat dapat digunakan untuk berbagai platform seperti Windows, Mac, android, IOSS, PS3 dan juga Wii (Roedavan, 2014).

2.6 Bahasa Pemrograman C#

C# merupakan bahasa pemrograman yang sederhana, modern, bersifat pemrograman berbasis objek (OOP) (Rhône, 2006). C# memiliki akar di dalam keluarga Bahasa C dan dapat dengan mudah dikenali bagi para *programmer* C, C++, dan Java. C# dikembangkan oleh Microsoft di dalam .NET dan kemudian disetujui sebagai sebuah standar oleh Ecma (ECMA-334) dan ISO (ISO/IEC 23270:2006). Kompiler C# milik Microsoft untuk .NET Franework merupakan implementasi yang sesuai untuk kedua standar tadi (Hejlsberg, et al., 2011).

2.7 Pengujian Perangkat Lunak

Pengembangan sistem perangkat lunak melibatkan beberapa aktivitas produksi dimana kemungkinan terdapat kesalahan manusia sangat besar. Kesalahan bisa saja muncul di bagian awal dari proses dan tujuan akhir bisa jadi salah atau tidak sesuai dengan yang diinginkan. Karena ketidaksempurnaan manusia dalam melakukan sesuatu, maka pengembangan perangkat lunak disertai oleh aktivitas QA (*quality assurance*) atau pengecekan kualitas (Deutsch, et al., 1979). Pengujian perangkat lunak merupakan elemen yang penting dari QA dan merupakan ulasan dari spesifikasi, *design*, dan pembuatan kode program (Pressman, 2001).

Menurut Glen Myres, terdapat aturan yang dapat digunakan agar pengujian perangkat lunak dapat berjalan dengan baik (Myers, 2004), yaitu:

1. Pengujian merupakan proses dari eksekusi sebuah program dengan tujuan untuk menemukan kesalahan.
2. *Test case* yang baik adalah *test case* yang memiliki probabilitas tinggi dalam menemukan kesalahan yang belum pernah ditemukan.
3. Sebuah pengujian yang sukses adalah pengujian yang menemukan kesalahan yang belum pernah ditemukan.

Semua produk teknis dapat diuji dengan satu dari dua acara. Yang pertama dengan mengetahui fungsi spesifikasi dari produk yang telah di-*design* untuk melakukan sesuatu, pengujian dapat dilakukan dengan cara mendemonstrasikan setiap fungsi apakah berjalan sesuai dengan tujuan, di saat yang sama juga mencari kesalahan dari setiap fungsi. Dan yang kedua dengan mengetahui kinerja internal dari suatu produk, pengujian dapat dilakukan untuk memastikan "semua gerigi" yang merupakan operasi internal bekerja sesuai dengan spesifikasi dan semua komponen internal telah diuji dengan seksama. Pengujian yang pertama disebut *black-box testing* dan pengujian yang kedua disebut *white-box testing*.

2.7.2 Black-box Testing

Black-box testing, berfokus terhadap kebutuhan fungsional dari sebuah perangkat lunak. Pengujian ini dilakukan dengan cara memberikan inputan dengan berbagai kondisi yang akan melatih secara keseluruhan dari semua kebutuhan fungsional perangkat lunak. Pengujian ini bukanlah sebagai alternativ dari *white-box testing* namun sebagai pengujian yang melengkapi dan bertujuan untuk menemukan kesalah yang tidak ditemukan dari metode *white-box* (Pressman, 2001).

Hampir semua pengujian *game* menggunakan *black-box*, yaitu pengujian yang dilakukan di luar aplikasi. Penguji *game* tidak membutuhkan pengetahuan atau akses ke dalam *source code*. Penguji *game* pada dasarnya tidak mencari kesalahan dengan cara membaca barisan kode. Namun, penguji berusaha menemukan kesalahan menggunakan perangkat masukan yang tersedia bagi *player*. *Black-box testing* merupakan pengujian yang paling efektif untuk menguji sistem yang komplek sekali maupun *game* yang paling sederhana (Schultz, et al., 2005).

2.7.2.1 Test Flow Diagram

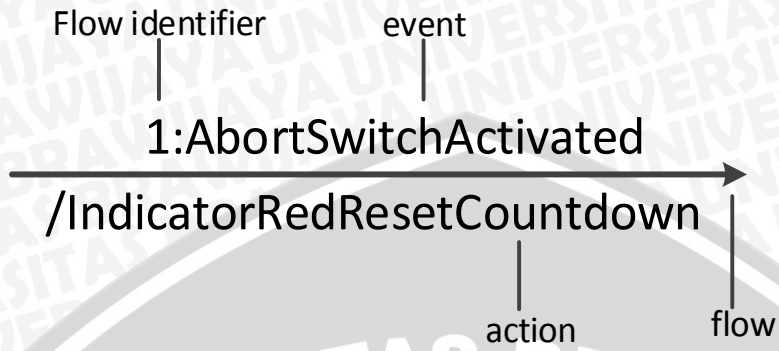
Test flow diagram (TFD) merupakan model grafis yang menggambarkan perilaku *game* dari perspektif *player* (Schlutz, et al., 2005). Pengujian dilakukan dengan cara menelusuri diagram untuk menguji *game* dengan jalan yang umum maupun jalan yang tidak terduga. Dengan penampilan yang berupa grafis, TFD memberikan penguji, pengembang, dan produser kemampuan untuk menganalisa dengan mudah, dan memberikan masukan pada desain pengujian.

Sebuah TFD dibuat dengan menyusun berbagai komponen gambar yang disebut dengan komponen. Elemen tersebut digambar, diberi label, dan saling berhubungan tergantung pada aturan tertentu. Dengan mengikuti aturan tersebut membuat tes lebih mudah dimengerti dan lebih mudah untuk digunakan kembali untuk projek yang selanjutnya. Berikut ini adalah elemen-elemen dari TFD.

1. Flow

   Flow digambarkan dengan sebuah garis yang menghubungkan satu *state* dari sebuah game ke *state* yang lain dengan anak panah menunjukkan arah dari *flow* tersebut. Setiap flow juga memiliki nomer identifikasi yang unik, sebuah *event*, dan sebuah *action*. Gram ":" mebatasi antara nama *event* dari nomer identitas *flow*. Dan garis miring "/" membatasi aksi dari *event*. Dalam proses

pengujian, penguji melakukan apa yang ditetukan oleh *event* dan kemudian mengecek *behavior* yang ditentukan baik oleh *action* dan *state* tujuan *flow*. Contoh dari flow bisa dilihat pada gambar 2.4.



Gambar 2.1 Contoh Flow

Sumber: (Schlutz, et al., 2005)

2. *Event*

   Event adalah operasi yang dilakukan oleh pemain. *Event* adalah sesuatu yang dilakukan secara eksplisit dalam game. Contohnya adalah mengambil sebuah benda, memilih mantra untuk diucapkan, mengirim pesan kepada pemain lain, dan waktu permainan yang habis.

3. *Action*

   Action menunjukkan *behavior* (perilaku) sementara atau transisi untuk merespon sebuah *event*. *Action* dapat dirasakan melalui indera manusia dan fasilitas *gaming platform*, termasuk suara, efek visual, umpan balik dari *controller game*, dan informasi yang dikirim melalui jaringan *game multiplayer*. *Action* tidak bertahan lama. *Action* dapat dirasakan, dideteksi, atau diukur ketika action tersebut sedang berlangsung namun tidak dapat dirasakan, dideteksi, atau diukur beberapa saat kemudian.

4. *State*

   State menunjukkan *behavior game* yang tetap dan terus berjalan. Selama *state* tersebut tidak dihentikan maka *behavior* yang sama akan terus muncul, dan setiap kembali ke *state* tersebut *behavior* tersebut akan terdeteksi. Sebuat *state* digambarkan dengan bentuk lingkaran dengan nama yang unik di dalamnya.

5. *Primitive*

   Event, action, dan *state* juga disebut sebagai *primitive*. Definisi dari *primitive* memberkan detail pada *behavior* yang ditampilkan melalui TFD tanpa mengacaukan diagram. Pengertian *primitive* membentuk sebuah kamus data untuk TFD. Pengertian tersebut dapat dalam bentuk teks (misalnya dalam bahasa Inggris), bahasa pemrograman (misalnya C#), atau sebuah bahasa pengujian.

6. *Terminator*

Terminator adalah kotak spesial yang ada pada TFD yang menunjukkan kapan tes dimulai dan berakhir. Hanya ada dua terminator dalam sebuah TFD. Yang pertama adalah "IN" yang pada umumnya hanya memiliki sebuah *flow* yang mengarah ke sebuah *state*. Sedangkan yang lain adalah "OUT", yang memiliki satu *flow* alau lebih dan berasal dari satu *state* atau lebih.

2.7.3 White-Box Testing

Berbeda dengan *black-box testing*, *white-box testing* memberikan kesempatan kepada penguji untuk menelusuri *source code* secara langsung yang mana bagian ini tidak dapat terakses oleh pengguna tingkat akhir (Schultz, et al., 2005). Menguji sebuah *game* menggunakan metode ini saja akan sangat sulit karena hal ini hampir tidak mungkin menghitung kompleksitas dari umpan balik *player*. Namun, terdapat beberapa situasi dimana *white-box testing* lebih dibutuhkan daripada *black-box testing*, seperti berikut:

- Pengujian dilakukan oleh pengembang untuk menambahkan kode baru untuk integrasi dengan keseluruhan *game*.
- Pengujian kode yang yang akan menjadi bagian dari *reusable library* dalam berbagai *game* atau/dan *platform*.
- Pengujian *method* atau fungsi dari kode yang merupakan bagian penting dari *game engine* atau produk *middleware* (perangkat lunak yang berfungsi sebagai jembatan antara sistem operasi atau *database* dengan aplikasi, terutama pada suatu jaringan).
- Pengujian kode di dalam *game* yang mungkin akan digunakan oleh pengembang pihak ketiga atau "*modders*" yang dapat mengembangkan atau memodifikasi *behavior* dari *game* sesuai keinginan mereka.
- Pengujian rutinitas level rendah yang biasanya *game* mendukung fungsi spesifik di dalam perangkat keras terbaru seperti kartu grafis atau *processors* audio.

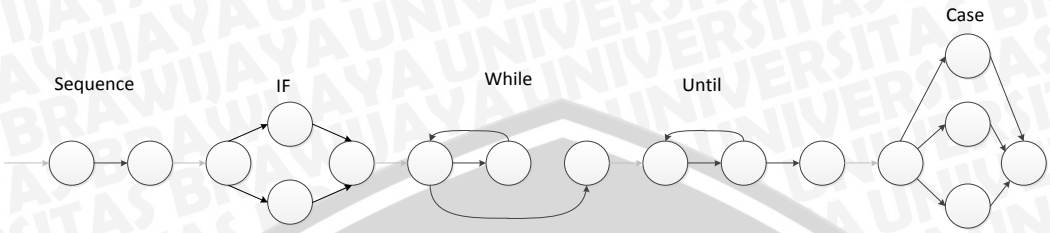
Dalam melakukan *white-box testing*, penguji akan mengkesekusi modul dan jalur bervariasi yang dapat diikuti oleh kode ketika penguji menggunakan modul dengan jalur bervariasi. Masukan dari pengujian ditentukan oleh tipe dan nilai dari data yang dapat dilalui oleh kode. Hasilnya akan diperiksa dengan memeriksa nilai yang dikembalikan oleh modul, variabel global yang dipengaruhi oleh modul, dan variabel local yang diproses di dalam modul.

2.7.3.1 Basis Path Testing

Basis path testing adalah metode pengujian *white box* yang diusulkan pertama kali oleh Tom McCabe. Metode *basis path* mengijinkan desainer *test case* untuk mengambil ukuran kompleksitas logika dari desain prosedural dan menggunakan ukuran ini sebagai acuan dalam mendefinisikan basis set. *Test case* diperoleh

untuk mencoba apakah basis set menjalankan setiap *statement* dalam program paling tidak sekali dalam proses *testing* (Pressman, 2001).

1. Notasi *Flow Graph*



Gambar 2.1 Notasi *Flow Graph*

Sumber: (Pressman, 2001)

Flow graph menggambarkan *logical control flow* menggunakan notasi yang diilustrasikan pada gambar 2.2. Setiap struktur memiliki simbol *flow graph* tersendiri.

2. *Cyclomatic Complexity*

Cyclomatic complexity atau kompleksitas siklomatik ada metrik perangkat lunak yang menunjukkan ukuran kuantitatif kompleksitas logika dari sebuah program. Dalam konteks metode *basis path testing*, nilai untuk *cyclomatic complexity* menujukkan nomer dari *path* independen dalam *basis set* sebuah program. *Path* independen adalah semua *path* yang ada dalam program yang paling tidak berupa satu set pemrosesan *statement* atau sebuah kondisi baru.

Perbedaan antara flowchart dan flow graph bisa dilihat pada gambar 2.3. *Cyclomatic complexity* memiliki fondasi yang kuat pada teori graph dan menyediakan matrik perangkat lunak yang sangat berguna. Tingkat kompleksitas bisa dihitung dengan salah satu dari tiga cara yang ada.

1. Jumlah *region* dari *flow graph*.
2. *Cyclomatic complexity*, V(G), untuk sebuah *flow graph* ditentukan dengan:

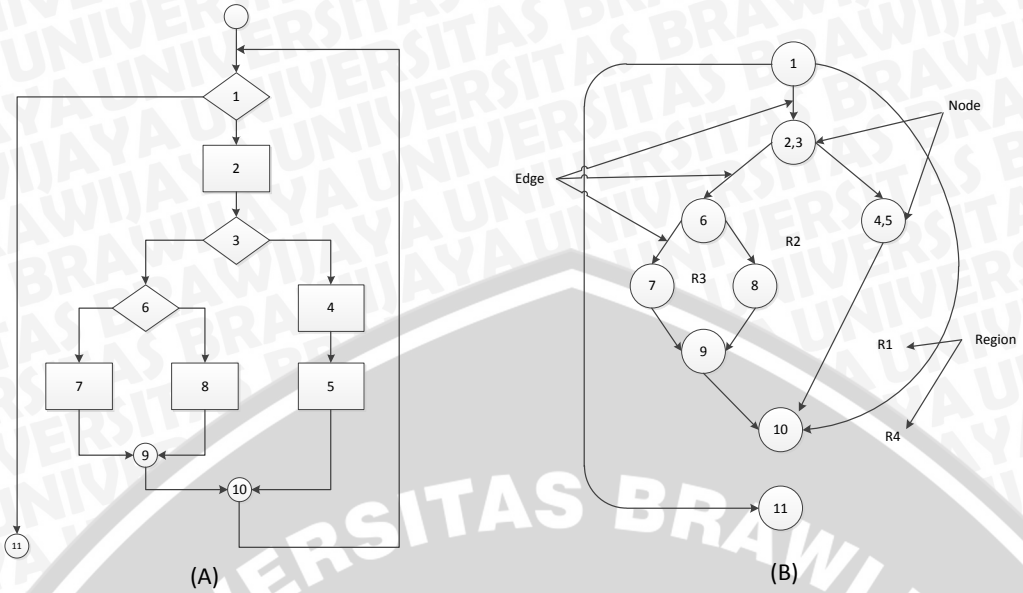
$$\mathbf{V(G) = E - N + 2} \quad \text{.....(1)}$$

Dimana E adalah jumlah *edge*, dan N adalah jumlah *node* dalam sebuah *flow graph*.

3. *Cyclomatic complexity*, V(G), untuk sebuah *flow graph* juga dapat ditentukan dengan cara:

$$\mathbf{V(G) = P + 1} \quad \text{.....(2)}$$

Dimana P adalah *predicate nodes* yang ada pada *flow graph*. *Predicate node* merupakan *node* yang memiliki kondisi.



Gambar 2.2 Flwowchart (A) Dan *Flow Graph* (B)

Sumber: (Pressman, 2001)

2.7.4 Playtesting pada Game

2.7.4.1 Jenis Playtest

1. *Bug Tester* (atau *Quality Assurance*)

   Quality Assurance atau QA bertujuan untuk menemukan kesalahan pada *game* yang berhubungan dengan desain atau kode porgram. Pada jenis playtest ini sama sekali tidak berusaha mencari tahu apakah game menarik atau menyenangkan. Jika program berjalan tidak sesuai dengan yang diharapkan oleh desainer maka hal ini dianggap sebagai *bug* yang harus diperbaiki.

2. Pengujian Terhadap Target Pengguna (*Focus Testing*)

   Pada *focus testing*, desainer mengumpulkan pemain yang termasuk dalam target pengguna untuk mengetahui seberapa berhasilkah *game* memenuhi kebutuhannya. Pengujian jenis ini biasanya digunakan untuk tujuan pemasaran.

3. *Usability Testing*

   Pengujian jenis ini dilakukan untuk mengetahui apakah pemain dapat mudah melakukan kontrol pada *game*. Hal ini biasanya dilakukan oleh industri pernagkat lunak untuk memastikan bahwa perangkat lunak yang dibuat mudah untuk dipelajari dan digunakan.

4. *Balance Testing*

Digunakan untuk mengetahui apakah *game* sudah seimbang. Pemain akan merasa frustasi jika game yang dibuat terlalu sulit dan akan mudah bosan jika *game* terlalu mudah. Jika ditemukan ketidakseimbangan, maka desainer dapat segera memperbaikinya.

5. *Fun Testing*

   Game dapat diperbaiki namun kadang masih bisa tidak menarik. "*Fun Factor*" adalah hal yang sulit untuk didesain, namun akan menjadi sangat jelas saat melihat pemain memainkan *game* tersebut. Menarik atau tidaknya sebuah *game* bisa dilihat dari reaksi orang yang memainkannya. Sangat penting untuk mengetahui aspek apa yang membuat sebuah *game* menarik.

2.7.4.2 Jenis Playtester

Terdapat beberapa jenis *playtester* yang memiliki kelebihan dan kekurangan masing-masing, dan beberapa lebih penting daripada yang lain (Schreiber, 2009).

1. Diri Sendiri/Desainer

   Playtester ini merupakan yang paling berharga. Orang yang membuat dan merancang sebuah game memahami lebih baik dari pada siapapun bagaimana *game* yang dirancang berjalan.

2. Desainer *Game* Lain

   Desainer *game* lain dapat melakukan analisa secara kritis dan memberikan solusi yang baik.

3. Teman Dekat, Keluarga, Dan Kenalan

   Jenis ketiga ini adalah orang-rang disekitar yang mudah untuk dimintai tolong untuk melakukan pengujian. Mereka dapat berperan sebagai tester tahap awal pengembangan.

4. Pemain *Game* Berpengalaman

   Para pemain *game* berpengalaman sangat cocok untuk balance testing, karena mereka baik dalam menemukan eksploitasi dan strategi dominan dalam *game*.

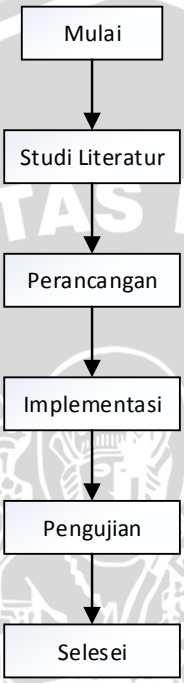
5. Orang Tidak Dikenal

   Orang tidak dikenal cenderung lebih objektif dalam memberikan masukan. Temukan orang-orang asing yang ada dalam target pengguna, karena mereka sangat cocok untuk untuk melakukan *focus testing* dan *usabiliy testing*.

BAB 3 METODOLOGI

3.1 Metode Penelitian

Skripsi ini disusun dengan beberapa langkah yaitu studi literatur, perancangan, implementasi, dan pengujian. Pada gambar 3.1 menunjukkan diagram metode penelitian.



Gambar 3.1 Diagram Alur Metodologi Penelitian

Sumber: Perancangan

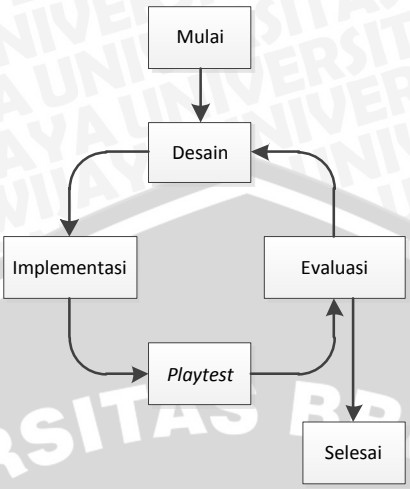
Studi literatur merupakan dasar teori yang digunakan untuk mendukung pengerjaan skripsi ini. Berikut ini beberapa teori yang digunakan.

1. *Game*
2. Sistem Pencernaan Manusia
3. *MDA Framework*
4. *Paper Prototyping*
5. Unity
6. Bahasa Pemrograman C#
7. Pengujian Perangkat Lunak

3.2 Perancangan *Game*

Metode yang digunakan dalam perancangan *game* edukasi biologi SMP sistem pencernaan manusia adalah *iterative with rapid prototyping*. Perbedaan metode ini dengan metode *waterfall* adalah jika metode *waterfall* prosesnya hanya dapat berjalan satu arah, metode *iterative* dapat kembali ke proses sebelumnya untuk melakukan perbaikan. Sehingga apabila ada elemen dalam *game* yang ingin

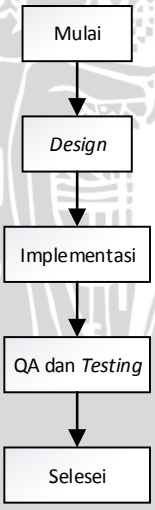
diubah atau diperbaiki, hal tersebut bisa segera dilakukan dengan mudah. Berikut ini diagram alur dari metode perancangan *iterative*.



Gambar 3.2 Diagram Alur Metodologi Perancangan Iterative

Sumber: (Schreiber, 2009)

Pada proses perancangan *game* sebaiknya dilakukan iterasi atau perulangan. Proses perulangan ini harus melalui semua alur yang terdapat pada diagram pada gambar 3.1, yaitu desain, implementasi, *playtest*, dan evaluasi. Pada proses ini sebaiknya menggunakan *paper prototyping* karena akan membuat proses iterasi berjalan lebih cepat.



Gambar 3.3 Diagram Alur Metodologi Perancangan *Iterative with Rapid Prototyping*

Gambar 3.2 menunjukkan proses implementasi secara digital setelah melakukan perancangan. Proses implementasi digital bisa lebih cepat dan jelas karena sudah dijabarkan pada proses *prototyping*. Setelah mengimplementasikan secara digital, pengembang dapat mengevaluasi dan menentukan apakah perlu melakukan perbaikan. Setelah implementasi digital dirasa sudah sesuai dengan target maka proses bisa berhenti.

BAB 4 PERANCANGAN DAN IMPLEMENTASI

4.1 Elemen Formal *Game*

Sebuah *game* haru memiliki elemen-elemen tertentu sehingga bisa disebut sebagai *game*. Elemen-elemen ini harus bisa didefinisikan secara jelas. Elemen-elemen formal tersebut adalah *player, goal, resource* dan *resource management*, *game state, information, sequencing, player interaction, theme*, dan *game as system*.

4.1.1 Deskripsi Game

Game "Nutriman Evolution" adalah *game* ber*genre platform* dengan konten edukasi untuk mengenalkan sistem pencernaan manusia pada siswa SMP. Pada tabel 3.1 ditunjukkan gambaran umum tentang game "Nutriman Evolution".

Tabel 4.1 Gambaran Game "Nutriman Evolution"

No	Elemen	Keterangan
1.	Judul Game	Nutriman Evolution
2.	Platform	Android
3.	Target Usia	12-16
4.	Rating ESRB	E (*Everyone*)
5.	Genre	*Shooter*, Edukasi
6.	Gameplay	Menembak dan menghindari halangan.
7.	*Unique Selling Point* (USP)	1. Sebagai media pembelajaran berbentuk *game*. 2. Sebagai media alternatif pembelajaran mengenal sistem pencernaan manusia.

Sumber: Perancangan

4.1.1.1 Player

Untuk *game* edukasi ini dirancang untuk dimainkan hanya untuk satu *player* atau pemain. Pemain akan bermain melawan sistem. Identifikasi aktor ditunjukkan pada tabel 3.2.

Tabel 4.2 Identifikasi Aktor

No	Aktor	Deskripsi
1.	Pemain	*Player* bermain untuk menembaki musuh dan menghindari halangan, serta membunuh *boss* di akhir permainan. Jika *boss* dapat

		terkalahkan maka *player* akan mendapatkan enzim yang disimpan oleh *boss*, kemudian enzim tersebut digunakan untuk berevolusi.

Sumber: Perancangan

4.1.1.2 Goal

Untuk menyelesaikan permainan, *player* harus menyelesaikan seluruh level yang ada dalam *game*. *Player* harus menyelesaikan pertarungan dengan musuh untuk lolos dari setiap level.

4.1.1.3 Rules

Berikut penjelasan dari *rule* pada *gameplay* dalam *game* ini, yaitu:

1. **Memilih Karakter**
   Setiap level memiliki karakter yang cocok untuk dimainkan. *Player* diharuskan untuk memilih karakter yang cocok itu. Karakter digambarkan sebagai zat makanan yaitu karbohidrat, protein, lemak, dan vitamin. Jika *player* salah memilih karakter akan berimbas pada ketidakoptimalan proyektil yang ditembakkan untuk membunuh musuh serta akan membuat lambat pergeseran karakter untuk menghindari halangan.
2. **Membunuh Musuh**
   Di setiap level akan muncul musuh yang menghalangi pergerakan *player.* Musuh ini diibaratkan sebagai bakteri yang mengganggu di dalam sistem pencernaan manusia. Setiap musuh yang dibunuh akan menambah nilai untuk *player*. Dan jika *player* menabrak musuh, maka permainan berakhir. Di akhir permainan *player* juga harus membunuh musuh yang lebih kuat (bos). Bos ini menyimpan enzim yang berguna bagi karakter untuk berevolusi. Tidak semua level menyimpan enzim, ketersediannya tergantung dengan kondisi sebenarnya di dalam tubuh manusia.
3. **Menghindari Halangan**
   Selain musuh, akan muncul halangan. Namun halangan ini tidak dapat dihancurkan dengan proyektil, namun harus dihindari. Setiap kali bermain, *player* akan diberi tiga nyawa. Setiap menabrak halangan, nyawa akan berkurang satu. Jika nyawa habis maka permainan akan berakhir.

4.1.1.4 Resource And Resource Management

Resource yang ada dalam game ini adalah nyawa dan juga karakter. Seperti dijelaskan pada sub bab *rules* bahwa setiap bermain *player* akan diberi tiga nyawa yang akan berkurang jika menabrak halangan atau terkena tembakan bos. Sedangkan karakter harus dipilih dengan cermat oleh *player* agar *game* dapat berjalan dengan optimal.

4.1.1.5 Game State

Pada *game* ini, *state* yang ada adalah level *lock* (Terkunci)/*unlock* (tidak terkunci), karakter dan enzim. Jika pemain telah menyelesaikan sebuah level,

maka *state* level selanjutnya berubah menjadi *unlock* yang berarti level tersebut sudah dapat dimainkan. Karakter dan enzim juga akan terbuka jika sudah mencapai level tertentu.

4.1.1.6 Information

Pada *game* ini *player* akan mengetahui semua informasi seperti karakter dan enzim. Namun untuk membuka semua informasi ini *player* harus menyeleseikan semua level dalam *game*.

4.1.1.7 Sequencing

Game ini dirancang untuk dimainkan oleh satu *player* saja. Sehingga tidak ada giliran bermain antar *player*.

4.1.1.8 Player Interaction

Game ini dirancang untuk dimainkan oleh satu *player* saja. Sehingga tidak ada interaksi antar *player*.

4.1.1.9 Theme

Tema dalam *game* ini adalah untuk membantu karakter berevolusi hingga akhir permainan. Karakter pada mulanya berbentuk zat karbohidrat, protein, lemak, dan vitamin. Dan enzim untuk berevolusi telah disembunyikan oleh bos yang harus dibunuh per level. Tiap level yang menggambarkan organ pencernaan manusia menyimpan enzim khusus untuk berevolusi. Enzim dan nama dari evolusi karakter disesuaikan dengan kondisi nyata pada sistem pencernaan manusia.

4.1.1.10 Game as System

Di dalam *game* ini tidak terdapat fitur dan sistem yang dapat dipadukan menjadi sebuah sistem yang kompleks.

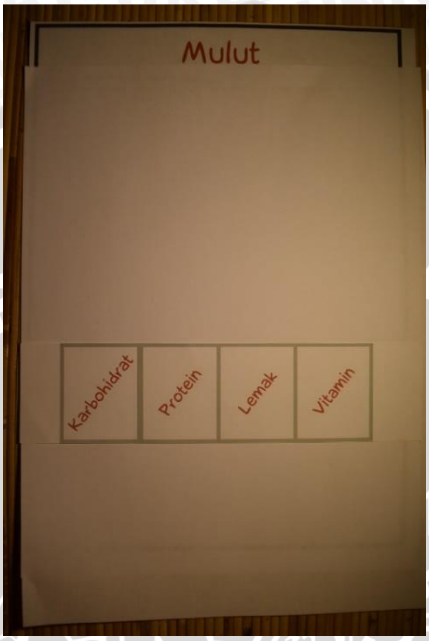
4.1.2 Paper Prototyping

Disini akan dijelaskan iterasi-iterasi yang dilakukan dalam melakukan *paper prototyping*. Dalam menentukan *gamplay* yang tepat untuk game "Nutriman Evolutin" ini dilakukan dua kali iterasi. *Paper prototyping* diujikan kepada sesame *game designer* agar mengetahui apakah *gameplay* sudah menyenangkan atau belum.

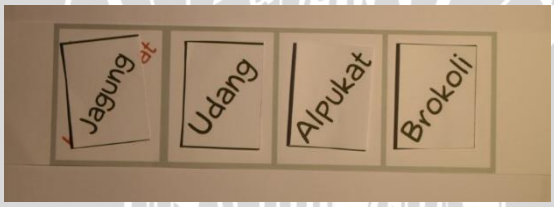
4.1.2.1 Iterasi Pertama

Gambar 4.1 menunjukkan *paper prototype* pada iterasi pertama. Awalnya game edukasi pengenalan sistem pencernaan manusia ini akan menggunakan *gameplay* mencocokkan bahan makanan dengan kandungan zat yang dimilikinya, misal nasi dan kentang masuk ke dalam golongan karbohidrat, ikan dan daging masuk dalam kategori protein. Setelah itu, muncul enzim yang terdapat di dalam organ, misal level satu di dalam mulut maka enzim yang muncul adalah amilase. *Player* kemudian mencocokkan enzim ini dengan bahan makanan yang cocok yaitu karbohidrat. Khusus untuk karbohidrat di dalam level ini akan berubah menjadi

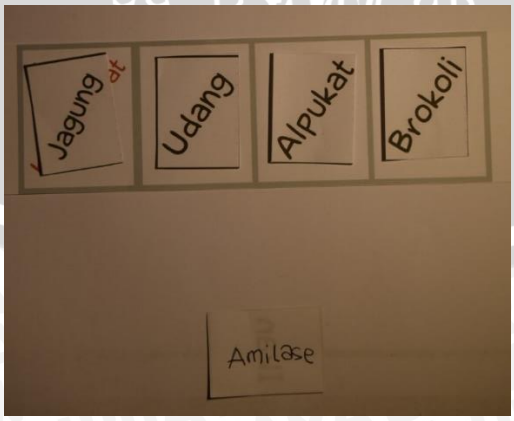
maltosa dan amilum. Lalu makanan yang sudah dikelompokkan tadi akan dipergunakan sebagai proyektil untuk menembak musuh. Akan muncul musuh yang harus dibunuh, dan setiap musuh memiliki kelemahan dalam proyektil tertentu, ada yang hanya bisa dibunuh dengan karbohidrat, protein, lemak, dan vitamin saja. Kemudian unsur yang tidak bisa digunakan untuk membunuh musuh harus ditembakkan menjauh dari musuh.



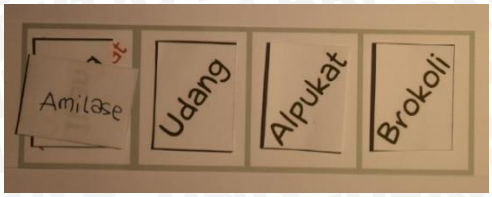
Gambar 4.1 Paper prototyping iterasi pertama



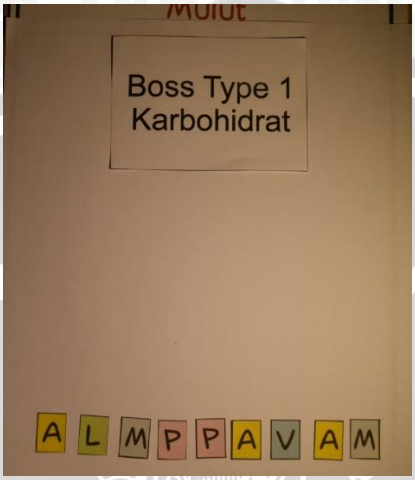
Gambar 4.2 Contoh kartu makanan yang dikelompokkan berdasarkan zat yang terkandung



Gambar 4.3 Contoh kartu enzim yang akan dicocokkan dengan zat



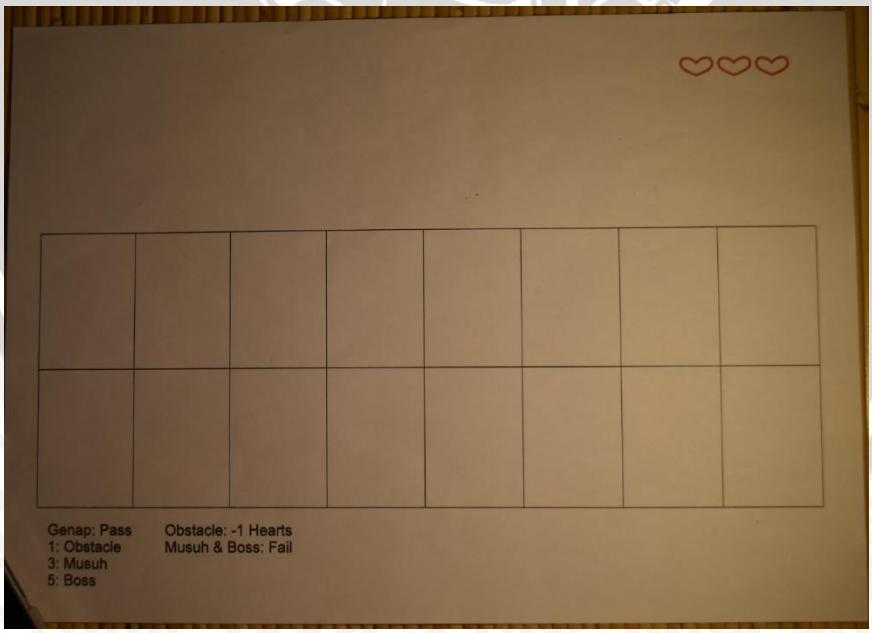
Gambar 4.4 Kartu enzim yang dimasukkan ke zat makanan



Gambar 4.5 Kartu makanan berubah menjadi kartu zat

Pada gambar 4.5 arti dari huruf A adalah amilum, L adalah lemak, M adalah maltose, P adalah protein, dan V adalah vitamin. Karbohidrat di level mulut berganti dengan amilum dan maltosa, sedangkan zat lain tetap.

4.1.2.2 Iterasi Kedua



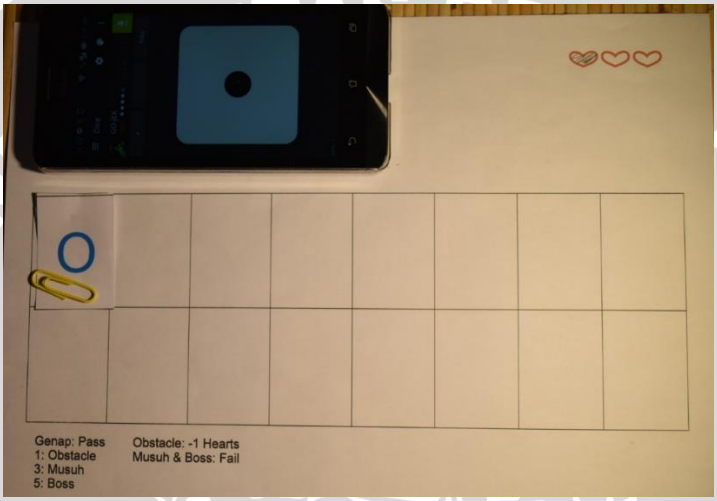
Gambar 4.6 *Paper prototyping* pada iterasi kedua

Gambar 4.6 menunjukkan *paper prototyping* pada iterasi kedua. Pada iterasi ini genre game adalah *shooting* dan menghindari *obstacle*, musuh, dan boss. Pada

iterasi ini bermain menggunakan dadu, jika dadu menunjukkan nilai genap maka *player* dapat berjalan satu blok tanpa halangan. Jika dadu menunjukkan nilai satu maka *player* menabrak *obastacle* dan mengurangi satu nyawa. Jika dadu menunjukkan nilai tiga maka *player* menabrak musuh dan permainan berakhir. Pada bagian kotak atas hanya bisa muncul *obstacle,* musuh dan nilai genap. Sedangkan di bagian kotak bawah hanya bisa muncul boss dan nilai genap. Jika dadu menunjukkan nilai lima maka akan mengurangi satu nyawa. Permainan berakhir jika nyawa habis atau dadu mengeluarkan angka tiga.

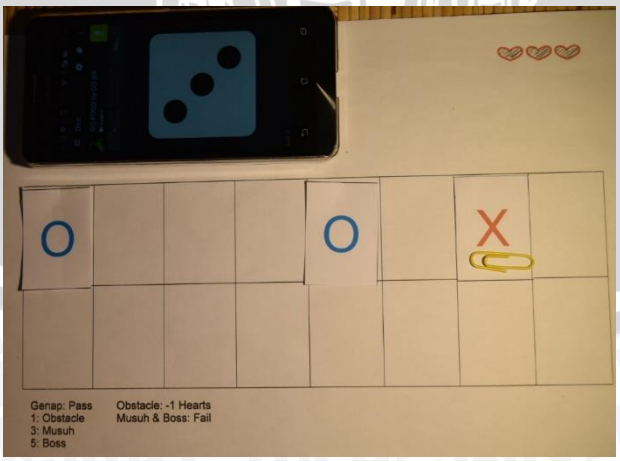
4.1.3 Playtesting

Pada bagian ini hanya akan dijelaskan *playtesting* pada iterasi kedua, karena iterasi ini yang akan diimplementasikan ke dalam bentuk digital.



Gambar 4.7 Dadu menunjukkan angka satu

Gambar 4.7 menunjukkan *paper prototyping* pada iterasi kedua. Pada awal permainan mendapatkan nomor dadu satu, maka nyawa berkurang satu dan muncul kartu *obstacle*.



Gambar 4.8 Dadu menunjukkan angka tiga

Gambar 4.8 menunjukkan permainan berakhir karena muncul dadu bernilai tiga yang berarti menabrak musuh.

4.2 Implementasi Game

Setelah melakukan perancangan dengan paper prototype, tahapan selanjutnya adalah implementasi game ke bentuk digital dengan menggunakan game engine. Tidak ada iterasi dalam proses iterasi. Pada subbab ini akan dijelaskan batasan implementasi dan implementasi gameplay baik implementasi proses maupun implementasi antarmuka.

4.2.1 Batasan Implementasi

Disini ini akan dijelaskan perangkat keras dan perangkat lunak yang digunakan untuk mengembangakan game edukasi kimia SMP pengenalan sistem periodik unsur. Ada beberapa batasan dalam proses implementasi, berikut ini beerapa batasan tersebut.

1. Bahasa pemrograman yang digunakan adalah C#.
2. Game dikembangkan untuk perangkat Android.

4.2.1.2 Spesifikasi Perangkat Keras

Berikut ini merupakan lingkungan perangkat keras yang digunakan untuk mengembangakan game edukasi kimia SMP pengenalan sistem periodik unsur.

1. Prosesor : Inter(R) Core(TM) i7-2670QM CPU @2.20-GHz
2. RAM : 8,00 GB
3. Graphic Card : Intel® HD Graphics 3000
4. Monitor : 14” HD LED LCD
5. Kapasistas Harddisk : 740 GB

4.2.1.3 Spesifikasi Perangkat Lunak

Berikut ini merupakan perangkat lunak yang digunakan untuk mengembangakan game edukasi kimia SMP pengenalan sistem periodik unsur.

1. Sistem Operasi Windows 7 Ultimate 64 bit.
2. Unity 5.2.1f1 Personal
3. MonoDevelop
4. Adobe Photoshop CS6.

4.2.2 Implementasi Gameplay

Pada tahap implementasi gameplay terdapat dua hal yang harus diimplementasikan. Yang pertama adalah implementasi porsedur program. Sedangkan yang kedua adalah imlementasi antarmuka.

4.2.2.1 Implementasi Prosedur Game

Berikut ini akan dijelaskan pseudocode dari metode-metode yang digunakan dalam proses implementasi *game* edukasi “Nutriman Evolution”. Metode-metode yang dijelaskan disini adalah metode yang paling penting dalam program.

1. Implementasi *Gameplay*

Pada kelas SelectCharacter berperan sebagai menentukan karakter apa saja yang tampil, yang dapat dipilih, dan menyimpan karakter yang dipilih oleh *player*. Method Awake() mengatur karakter apa saja yang bisa ditampilkan kepada *player*.

Tabel 4.3 Baris Program Awake()

No.	Method Awake()
1	PlayerPrefs.SetInt ("Karbo1", 1);
2	for (int i=1; i<=5; i++) {
3	if(PlayerPrefs.GetInt("Karbo"+i.ToString()) == 0)
4	karbo[i-1].interactable = false;
5	if (i<4){
6	if(PlayerPrefs.GetInt("Protein"+i.ToString()) == 0)
7	protein[i-1].interactable = false;
8	if(PlayerPrefs.GetInt("Lemak"+i.ToString()) == 0)
9	lemak[i-1].interactable = false;
10	}
11	}

Tabel 4.4 Baris Program setCharacterKarbo(int i)

No.	Method setCharacterKarbo(int i)
1	PlayerPrefs.SetInt ("KarboPil" + (i+1).ToString (), 1);
2	karPil [0].gameObject.GetComponent<Image> ().sprite = karbo [i].gameObject.GetComponent<Image> ().sprite;
3	ok.interactable = true;

Tabel 4.5 Baris Program setCharacterProtein(int i)

No.	Method setCharacterProtein(int i)
1	PlayerPrefs.SetInt("ProteinPil"+(i+1).ToString (), 1);
2	karPil [1].gameObject.GetComponent<Image> ().sprite = protein [i].gameObject.GetComponent<Image> ().sprite;

Tabel 4.6 Baris Program setCharacterLemak(int i)

No.	Method setCharacterLemak(int i)
1	PlayerPrefs.SetInt("LemakPil"+(i+1).ToString (), 1);
2	karPil [2].gameObject.GetComponent<Image> ().sprite = lemak [i].gameObject.GetComponent<Image> ().sprite;

Tabel 4.7 Baris Program setCharacterVitamin()

No.	Method setCharacterVitamin()
1	PlayerPrefs.SetInt("VitaminPil", 1);
2	karPil [3].gameObject.GetComponent<Image> ().sprite = vitamin.gameObject.GetComponent<Image> ().sprite;

Tabel 4.4, 4.5, 4.6, dan 4.7 untuk memasukkan karakter pilihan *player* ke dalam permainan. Setelah *player* memilih karakter yang akan dimainkan maka kelas GController akan berjalan. Pada kelas ini berfungsi untuk menampilkan musuh, *obstacle*, dan boss seuai dengan level yang dimainkan, serta menambahkan nilai jika membunuh musuh dan boss.

Tabel 4.8 Baris Program Start()

No.	Method Start()
1	pSprite = GameObject.Find ("Player").GetComponent<PlayerSprite> ();
2	score = 0;
3	updateScore();
4	currentLevelName = Application.loadedLevelName;
5	if (currentLevelName.Contains("Level1"))
6	StartCoroutine (spawnWaves1 ());
7	else if (currentLevelName.Contains("Level2"))
8	StartCoroutine (spawnWaves2 ());
9	else if (currentLevelName.Contains("Level3"))
10	StartCoroutine (spawnWaves3 ());
11	else if (currentLevelName.Contains("Level4"))
12	StartCoroutine (spawnWaves4 ());
13	else if (currentLevelName.Contains("Level5"))
14	StartCoroutine (spawnWaves5 ());

Tabel 4.8 menunjukkan Coroutine mana yang akan dijalankan sesuai dengan level yang dipilih oleh *player*.

Tabel 4.9 Baris Program spawnWaves1()

No.	Method spawnWaves1()
1	if (PlayerPrefs.GetInt ("KarboPil1") == 1)
2	karakterPilihan = true;
3	yield return new WaitForSeconds (startWait);
4	for (int c=1; c<=5; c++) {
5	for (int i=0; i<miniBossCount; i++) {
6	ranX = Random.Range (-spawnValuesMiniBoss.x, spawnValuesMiniBoss.x); //random posisi obstacle
7	Vector2 spawnObsPosition;
8	Vector2 spawnMiniPosition;
9	spawnObsPosition = new Vector2 (ranX, spawnValuesObs.y);
10	obs = Instantiate (obstacle, spawnObsPosition, Quaternion.identity) as GameObject;
11	moverObs = obs.GetComponent<MoverObsMini>();
12	if (ranX <= 0) { //obstacle kiri
13	ranX2 = Random.Range (ranX + 1.7f, spawnValuesMiniBoss.x);
14	} else if (ranX > 0) { //obstacle kanan
15	ranX2 = Random.Range (-spawnValuesMiniBoss.x, ranX - 1.7f);
16	}
17	spawnMiniPosition = new Vector2 (ranX2, spawnValuesMiniBoss.y);
18	mini = Instantiate (miniBoss, spawnMiniPosition, Quaternion.identity) as GameObject;
19	moverMini = mini.GetComponent<MoverObsMini>();
20	if (i==0){
21	sp = moverObs.speed + (float) c/2;
22	moverObs.speed = sp;
23	moverMini.speed = sp;
24	}else{
25	moverObs.speed = sp;

26	moverMini.speed = sp;
27	}
28	
29	yield return new WaitForSeconds (spawnWait);
30	}
31	yield return new WaitForSeconds (waveWait);
32	}
33	Vector2 spawnBossPostition = new Vector2 (0, 12f);
34	Instantiate (boss, spawnBossPostition, transform.rotation);

Tabel 4.9 menunjukkan bagaimana musuh akan ditampilkan untuk level satu. Pertama-tama akan dicek pada baris 1 dan 2 apakah karakter yang dipilih sesuai dengan seharusnya, jika betul maka akan menyimpan nilai benar ke dalam variabel karakterPilihan. Variabel ini nantinya akan diakses oleh kelas lain untuk menentukan akurasi tembakan *player* dan menentukan nilai yang akan diberikan jika boss berhasil terbunuh.

Tabel 4.10 Baris Program waitEndSukses()

No.	Method waitEndSukses()
1	yield return new WaitForSeconds (3.0f);
2	canvasSukses.enabled = true;
3	scoreAKhir.text = score.ToString ();
4	hearts = pSprite.heart * 20;
5	heartsAkhir.text = pSprite.heart+"x20 = "+hearts;
6	total = score + hearts;
7	totalText.text = total.ToString ();
8	unlockLevel ();
9	Time.timeScale = 0;

Tabel 4.10 untuk menampilkan panel kemenangan jika *player* ber hasil menyeleseikan permainan. Akan ada perhitungan nilai terakhir, dan juga membuka level selanjutnya.

Tabel 4.11 Baris Program waitEndFail()

No.	Method waitEndFail()
1	yield return new WaitForSeconds (2.0f);
2	canvasFail.enabled = true;
3	cleanCharacterSelect ();
4	Time.timeScale = 0;

Tabel 4.11 akan menampilkan panel kalah jika *player* gagal menyeleseikan level.

Tabel 4.12 merupakan baris kode untuk method OnTriggerEnter2D (Collider2D other) Kelas DestroyByContact yang berisi method untuk menghancurkan musuh jika terkena tembakan ataupun player jika bertabrakan dengan musuh.

Tabel 4.12 Baris Program OnTriggerEnter2D(Collider2D other)

No.	Method OnTriggerEnter2D(Collider2D other)
1	if (other.tag == "Boundary") {

```
2       return;
3   }
4   if (other.tag == "Player") {
5       Destroy (other.gameObject); //destroy player
6       Destroy (gameObject); //destroy musuh
7       ex = Instantiate(explosion, transform.position, transform.rotation) as
    GameObject;
8       Destroy(ex, 5);
9       gController.updateFail();
10  }
11  if (gController.karakterPilihan == true) {
12      Destroy(other.gameObject); //destroy bolt
13      Destroy(gameObject); //destroy musuh
14      ex = Instantiate(explosion, transform.position, transform.rotation) as
    GameObject;
15      Destroy(ex, 5);
16      gController.AddScore (10);
17  } else if (gController.karakterPilihan == false) {
18      Destroy(other.gameObject); //destroy bolt
19      if(countBolt > 0){
20          countBolt--;
21      } else if (countBolt == 0){
22          Destroy(gameObject); //destroy musuh
23          ex = Instantiate(explosion, transform.position, transform.rotation) as
    GameObject;
24          Destroy(ex, 5);
25          gController.AddScore (10);
26      }
27  }
```

Tabel 4.13 merupakan baris kode untuk method OnTriggerEnter2D (Collider2D other) Kelas DestroyByContactObs yang berisi method untuk menghancurkan tembakan yang dikeluarkan oleh *player* jika bertabrakan dengan *obstacle* serta menghancurkan *obstacle* jika bertabrakan dengan *player*.

Tabel 4.13 Baris Program OnTriggerEnter2D(Collider2D other)

No.	Method OnTriggerEnter2D(Collider2D other)
1	if (other.tag == "Boundary"){
2	return;
3	}
4	if (other.tag == "Player") {
5	ex = Instantiate(explosion, transform.position, transform.rotation) as GameObject;
6	Destroy(gameObject); //destroy obs
7	Destroy(ex, 5);
8	}
9	if (other.tag == "Bolt") {
10	Destroy (other.gameObject);
11	}

12	}

Tabel 4.14 merupakan baris kode untuk method OnTriggerEnter2D (Collider2D other) Kelas PlayerSprite yang berisi method untuk mengurangi nyawa dan menghancurkan *player* jika *player* bertabrakan dengan *obstacle* atau boss, serta untuk mengganti *sprite* gambar *player*.

Tabel 4.14 Baris Program OnTriggerEnter2D(Collider2D other)

No.	Method OnTriggerEnter2D(Collider2D other)
1	if (other.tag == "Boundary"){
2	return;
3	}
4	if (other.tag == "Obstacle" || other.tag == "Bolt") {
5	if (heart == 3){
6	player.GetComponent<SpriteRenderer>().sprite= sprite1;
7	} else if (heart ==2){
8	player.GetComponent<SpriteRenderer>().sprite= sprite2;
9	} else{
10	ex = Instantiate(explosion, transform.position, transform.rotation) as GameObject;
11	Destroy(gameObject);
12	Destroy(ex,5);
13	gController.updateFail();
14	}
15	heart--;
16	heartText.text = "x"+heart.ToString();
17	}

Tabel 4.15 merupakan baris kode untuk method OnTriggerEnter2D (Collider2D other) Kelas DestroyByContactBoss yang berisi method untuk menghancurkan tembakan yang dikeluarkan oleh *player* jika bertabrakan dengan boss, serta menentukan poin yang didapat oleh *player* jika berhasil membunuh boss.

Tabel 4.15 Baris Program OnTriggerEnter2D(Collider2D other)

No.	Method OnTriggerEnter2D(Collider2D other)
1	if (other.tag == "Boundary"){
2	return;
3	}
4	if (gController.karakterPilihan == true){
5	damagePoint = 2;
6	point = 50;
7	} else if (gController.karakterPilihan == false){
8	damagePoint =1;
9	point = 0;
10	}
11	if (other.tag == "Bolt") {
12	if (gController.karakterPilihan == true){
13	damagePoint = 2;
14	point = 50;

```
15      } else if (gController.karakterPilihan == false){
16          damagePoint =1;
17          point = 0;
18      }
19      if (hp > 0)
20          hp-=damagePoint;
21      else if (hp==0){
22          ex = Instantiate(explosion, transform.position, transform.rotation) as
    GameObject;
23          Destroy(gameObject);
24          Destroy(ex, 5);
25          gController.AddScore(point);
26          gController.updateSukses();
27      }
28      Destroy(other.gameObject);
29  }
```

Tabel 4.16 merupakan method Update() dari Kelas ButtonMover yang berisi method untuk menggerakkan capsulship berdasarkan inputan berupa tombol dari *player*.

Tabel 4.16 Baris Update()

No.	Method Update()
1	if (moveLeft && !moveRight) {
2	pos = pos - 0.01f;
3	float current = rb.position.x;
4	Debug.Log(current);
5	if(rb.position.x > posMin){
6	rb.AddForce(Vector2.left * speed);
7	rb.position = new Vector3 (Mathf.Clamp(rb.position.x, posMin,
8	posMax),
9	0.0f);
10	}
11	if (rb.position.x <= posMin){
12	fix = new Vector2 (posMin, 0.0f);
13	rb.velocity = fix * 0;
14	rb.position = new Vector3 (Mathf.Clamp(rb.position.x, posMin,
15	posMax),
16	0.0f);
17	}
18	}
19	if (moveRight && !moveLeft) {
20	pos = pos + 0.01f;
21	float current = rb.position.x;
22	Debug.Log(current);
23	if(rb.position.x < posMax){
24	Debug.Log("Jalan Kanan");
25	rb.AddForce(Vector2.right * speed);
26	rb.position = new Vector3 (Mathf.Clamp(rb.position.x, posMin,
27	posMax),

```
28              0.0f);
29      }
30      if (rb.position.x >= posMax){
31          fix = new Vector2 (posMax, 0.0f);
32          rb.velocity = fix * 0;
33          rb.position = new Vector3 (Mathf.Clamp(rb.position.x, posMin,
34  posMax),
35              0.0f);
36      }
37  }
```

2. Implementasi Penyimpanan Data

Pada *game* ini sistem akan menyimpan data. Data yang disimpan adalah level, enzim dan karakter yang dapat diakses. Pada table 4.17 menunjukkan baris program penyimpanan data pada kelas GController yang terdapat pada method unlockLevel().

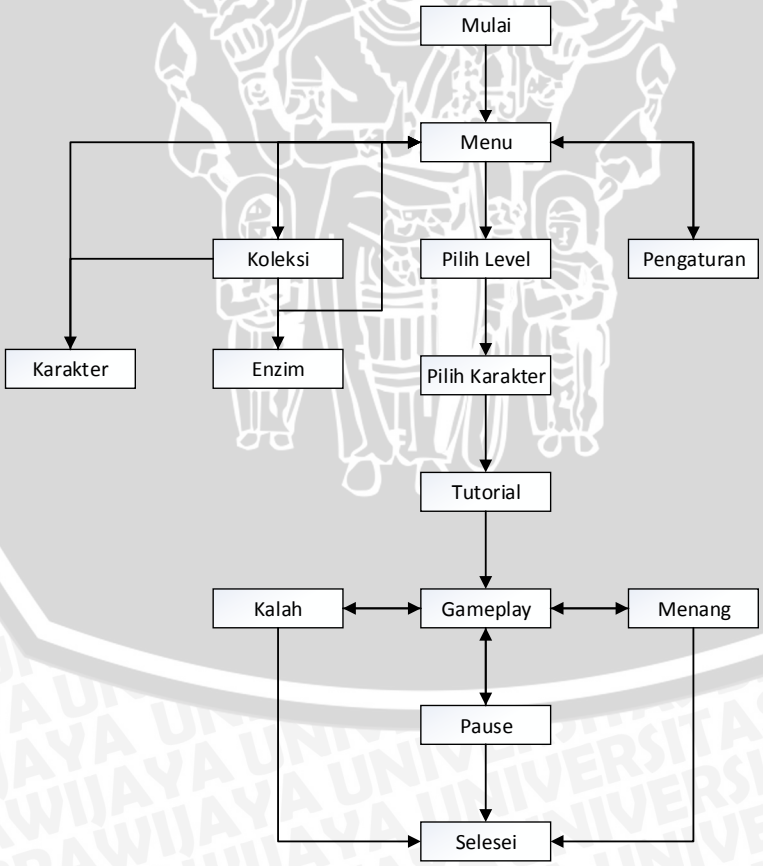
Tabel 4.17 Baris Program unlockLevel()

No.	Method unlockLevel()
1	for (int i=1; i<=6; i++) {
2	if(currentLevelName=="Level"+i.ToString()){
3	int indexLevel=(i+1);
4	PlayerPrefs.SetInt("Level"+indexLevel.ToString(),1);
5	}
6	}
7	
8	if (PlayerPrefs.GetInt("Level2") == 1){
9	if (PlayerPrefs.GetInt("Protein1") == 0
10	PlayerPrefs.SetInt("Enzim1", 1
11	PlayerPrefs.SetInt("Karbo2", 1);
12	PlayerPrefs.SetInt("Protein1", 1);
13	}
14	}
15	if (PlayerPrefs.GetInt ("Level3") == 1) {
16	if (PlayerPrefs.GetInt("Protein2") == 0){
17	PlayerPrefs.SetInt("Enzim2", 1);
18	PlayerPrefs.SetInt("Protein2", 1);
19	PlayerPrefs.SetInt("Lemak1", 1);
20	}
21	}
22	if (PlayerPrefs.GetInt ("Level4") == 1) {
23	if (PlayerPrefs.GetInt("Enzim3") == 0){
24	PlayerPrefs.SetInt("Enzim3", 1);
25	PlayerPrefs.SetInt("Enzim4", 1);
26	PlayerPrefs.SetInt("Karbo3", 1);
27	PlayerPrefs.SetInt("Protein3", 1);
28	PlayerPrefs.SetInt("Lemak2", 1);
29	}

```
30  }
31  if (PlayerPrefs.GetInt ("Level5") == 1) {
32      if (PlayerPrefs.GetInt("Karbo4") == 0){
33          PlayerPrefs.SetInt("Karbo4", 1);
34          PlayerPrefs.SetInt("Vitamin", 1);
35      }
36  }
37  if (PlayerPrefs.GetInt ("Level6") == 1) {
38      if (PlayerPrefs.GetInt("Karbo5") == 0){
39          PlayerPrefs.SetInt("Karbo5", 1);
40          PlayerPrefs.SetInt("Lemak3", 1);
41      }
42  }
43  cleanCharacterSelect ();
```

Method ini berfungsi untuk menyimpan data pemain. Baris 1-6 berfungsi untuk membuka level selanjutnya. Sedangkan baris 8-42 berfungsi untuk membuka enzim dan juga karakter sesuai dengan level yang terbuka. Sebelum membuka enzim dan karakter terdapat penyecekan seperti pada baris 9, 16, 23, 32, dan 37. Jika enzim dan karakter sebelumnya sudah terbuka maka tidak akan dibuka.

4.2.2.2 Implementasi Antar Muka

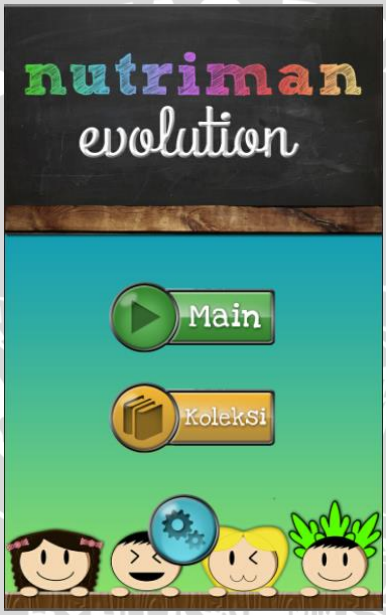


Gambar 4.9 Diagram alur antarmuka

Gambar 4.9 menunjukkan hasil implementasi antarmuka *game* edukasi "Nutriman Evolution". Ada beberapa halaman antarmuka pada *game* ini antara lain halaman main menu, pilih level, pilih karakter, *gameplay*, koleksi enzim dan karakter, dan pengaturan, dan tutorial.

1. Implementasi Halaman Awal

   Gambar 4.10 menunjukkan implementasi halaman awal. Dari halaman ini *player* dapat menuju halaman pilih level dengan menekan tombol main berwarna hijau, halaman koleksi dengan menekan tombol koleksi berwarna kuning, dan halaman pengaturan dengan menekan tombol berwarna biru. Pada halaman ini terdapat musik yang dapat dimatikan melalui menu pengaturan.



Gambar 4.10 Tampilan halaman awal

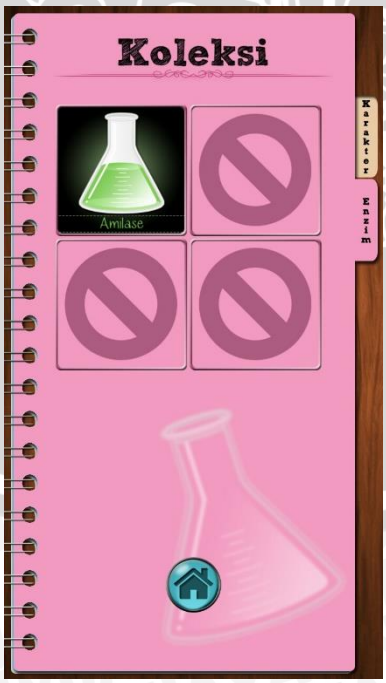
2. Implementasi Halaman Koleksi Karakter

   Gambar 4.11 menunjukkan implementasi halaman koleksi karakter. Dari halaman awal memilik tombol koleksi akan masuk ke halaman koleksi karakter karbohidrat. Arah panah ke kanan untuk melihat karakter lain seperti protein, lemak, dan vitamin. Pada menu ini akan ditampilkan karakter yang sudah terbuka. Pada gambar 4.11 tampak bahwa karakter karbohidrat masih terbuka satu, itu menunjukkan bahwa level yang terbuka hanya level satu. Jika pemain berhasil membuka semua level maka seluruh karakter akan ditampilkan. Tombol biru bergambar rumah untuk kembali ke halaman awal. Di samping terdapat tombol untuk berpindah ke koleksi karakter atau enzim.

Gambar 4.11 Implementasi halaman koleksi karakter

3. Implementasi Halaman Koleksi Enzim

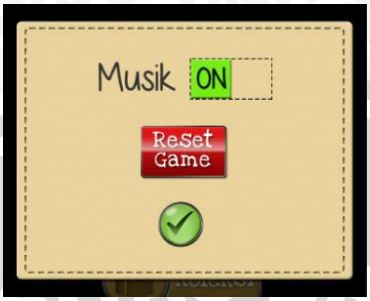
   Gambar 4.12 menunjukkan implementasi halaman koleksi enzim. Dari halaman awal memilik tombol koleksi akan masuk ke halaman koleksi karakter karbohidrat lalu memilih menu enzim di sebelah kanan. Pada menu ini akan ditampilkan enzim yang telah terbuka. Di samping terdapat tombol untuk berpindah ke koleksi karakter atau enzim.



Gambar 4.12 Implementasi halaman koleksi enzim

4. Implementasi Halaman Pengaturan

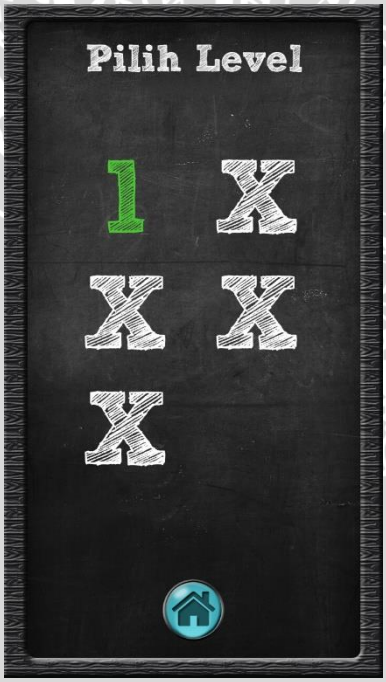
   Gambar 4.13 menunjukkan implementasi halaman pengaturan. Dari halaman utama pilih tombol biru bergambar *gear*. *Player* dapat mematikan dan menghidupkan musik yang ada di permainan dan juka me-*reset* ulang permainan. *Reset* akan berakibat hilangnya data permainan.



Gambar 4.13 Implementasi halaman pengaturan

5. Implementasi Halaman Pilih Level

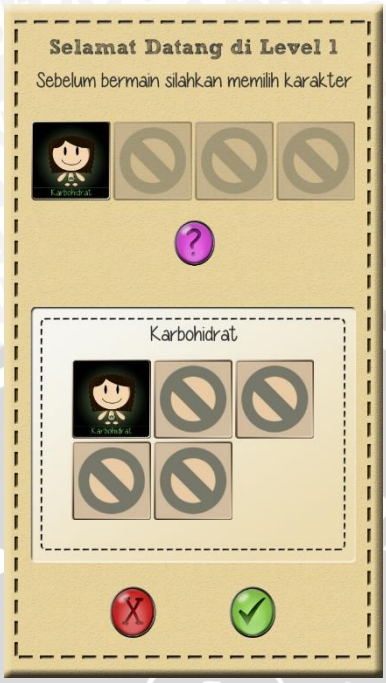
   Gambar 4.14 menunjukkan implementasi halaman pilih level. Dari halaman awal memilik tombol Main berwarna hijau. Pada menu ini akan ditampilkan level yang telah terbuka dan yang bisa dimainkan.



Gambar 4.14 Implementasi halaman pilih level

6. Implementasi Halaman Pilih Karakter

   Gambar 4.15 menunjukkan implementasi halaman pilih karakter. Setelah memilih level yang ingin dimainkan, *player* harus memilih karakter yang harus dibawa untuk bermain. Kesalahan memilih karakter akan berimbas kepada ketidak optimalan dalam permainan.



Gambar 4.15 Implementasi halaman pilih karakter

7. Implementasi Halaman Tutorial

   Gambar 4.16 menunjukkan implementasi halaman tutorial. Setelah memilih karakter yang akan dimainkan, halaman ini akan muncul.

Gambar 4.16 Implementasi halaman tutorial

8. Implementasi Halaman *Gameplay*

   Gambar 4.17 menunjukkan implementasi halaman *gameplay*.

Gambar 4.17 Implementasi halaman *gameplay*

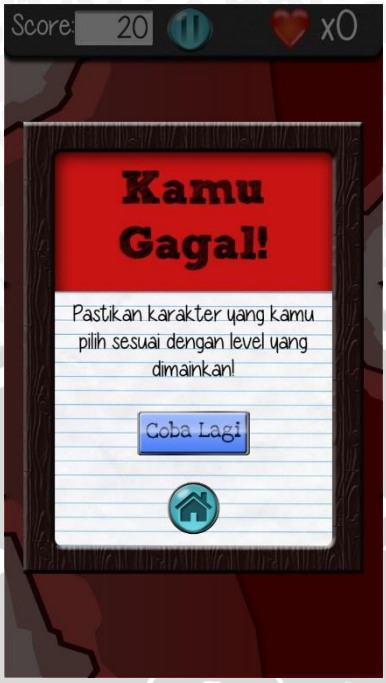
9. Implementasi Kondisi Menang

   Gambar 4.18 menunjukkan implementasi kondisi menang.

Gambar 4.18 Implementasi halaman menang

10. Implementasi Halaman Kalah

    Gambar 4.19 menunjukkan implementasi kondisi kalah.

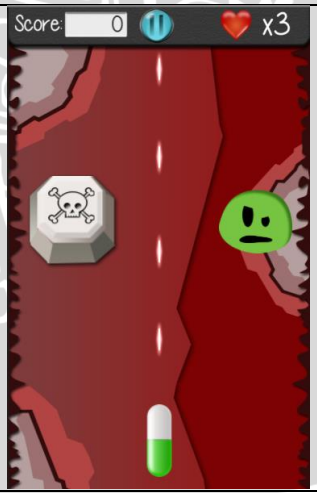


Gambar 4.19 Implementasi halaman kalah

4.2.2.3 Implementasi Level

1. Level 1

Tabel 4.17 Implementasi Level 1

Level 1	
Keterangan	Karakter yang harus dipilih: karbohidrat Enzim yang didapatkan: amilase, untuk mengubah karbohidrat menjadi amilum Untuk menyeleseikan level ini *player* harus menghindari *obastacle*, membunuh musuh dan *boss*.

2. Level 2

Tabel 4.28 Implementasi Level 2

Level 2	
Keterangan	Karakter yang harus dipilih: amilum, dan protein. Enzim yang didapatkan: pepsin, untuk mengubah protein menjadi pepton. Untuk menyeleseikan level ini *player* harus menghindari *obastacle*, membunuh musuh dan *boss*.

3. Level 3

Tabel 4.39 Implementasi Level 3

Level 3	
Keterangan	Karakter yang harus dipilih: amilum, pepton, dan lemak. Enzim yang didapatkan: tripsin, untuk mengubah pepton menjadi asam amino; lipase, untuk mengubah lemak menjadi asam lemak. Untuk menyeleseikan level ini *player* harus menghindari *obastacle*, membunuh musuh dan *boss*.

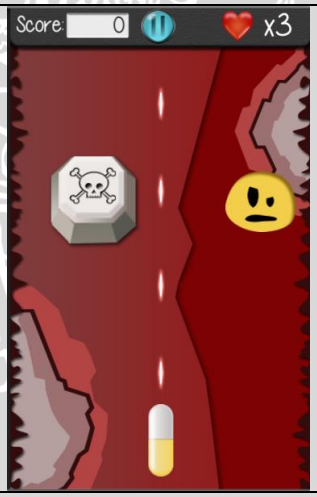
4. Level 4

Tabel 4.20 Implementasi Level 4

Level 4	
Keterangan	Karakter yang harus dipilih: maltosa, asam amino, dan asam lemak. Tidak enzim yang didapatkan dari level ini, namun maltose berubah menjadi monosakarida, asam lemak menjadi gliserol. Untuk menyeleseikan level ini *player* harus menghindari *obastacle*, membunuh musuh dan *boss*.

5. Level 5

Tabel 4.21 Implementasi Level 5

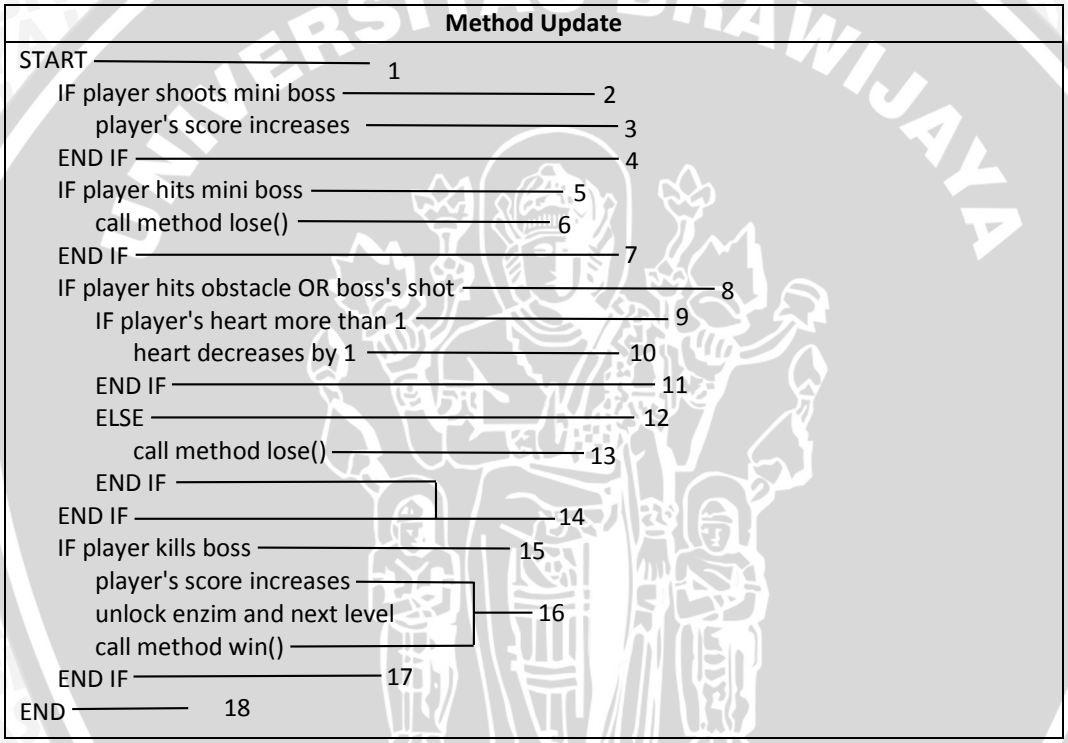
Level 5	
Keterangan	Karakter yang harus dipilih: monosakarida, asam amino, asam lemak, dan vitamin mineral. Untuk menyeleseikan level ini *player* harus menghindari *obastacle*, membunuh musuh dan *boss*.

BAB 5 PENGUJIAN

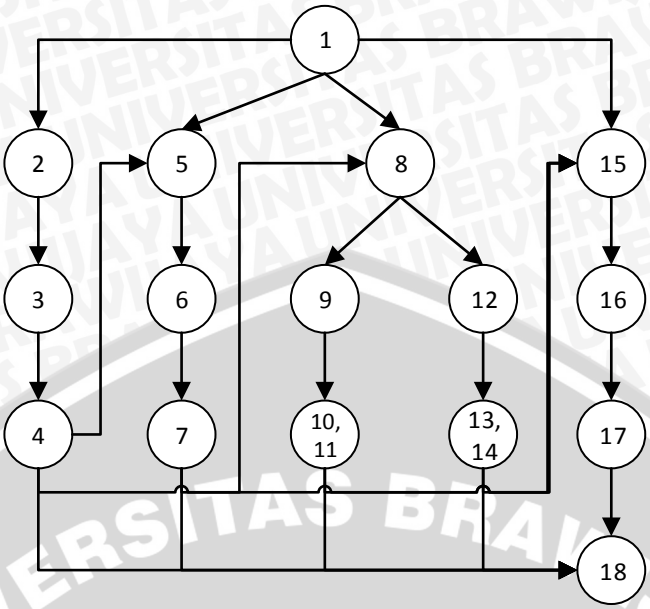
5.1 Hasil Pengujian Dengan *White Box* Testing

Pengujian unit adalah salah satu bentuk pengujian *white box* dengan menggunakan teknik *basis path testing*. Pada basis path testing, pengujian dibuat berdasarkan ukuran tingkat kompleksitas dari algoritma hasil perancangan. Pengujian unit yang dilakukan untuk penulisan laporan skripsi ini hanyalah pada fungsi-fungsi paling penting dalam program. Cara melakukan pengujian ini adalah dengan memanipulasi parameter atau inputan langsung ke dalam baris kode dan melihat hasilnya tanpa harus menjalankan aplikasi secara keseluruhan.

5.1.1 Pengujian *Gameplay*



Gambar 5.1 Pengujian Unit *Gameplay*



Gambar 5.2 *Flow Graph* Method Update

Gambar 5.2 Menunjukkan *flow graph* method Update. Untuk menghitung kompleksitas siklomatik (*Cyclomatic comlplexity*), digunakan persamaan V(G)=P+1. Dimana P adalah *predicate node* atau *node* yang memiliki kondisi.

V(G)=P+1

V(G)=5+1

V(G)=6

Jalur independen:

1. 1-2-3-4-5-6-7-18
2. 1-2-3-4-8-12-13-14-18
3. 1-2-3-4-8-9-10-11-15-16-17-18
4. 1-5-6-7-18
5. 1-8-9-10-11-15-16-17-18
6. 1-15-16-17-18

Pada tabel 5.1 adalah hasil pengujian yang dilakukan pada method Update. Jalur diperoleh dari perhitung kompleksitas siklomatik dari method Update. Ada 8 kasus uji sesuai dengan jalur yang diperoleh pada perhitungan sebelumnya. Untuk lebih jelasnya silakan lihat pada tabel 5.1.

Tabel 5.1 Pegunjian Unit Kondisi Menang Dan Kalah

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	*Player* menembak musuh	Mendapat 10 point	Mendapat 10 point
2	*Player* menabrak *obstacle* atau tembakan dari boss	Permainan berakhir	Permainan berakhir

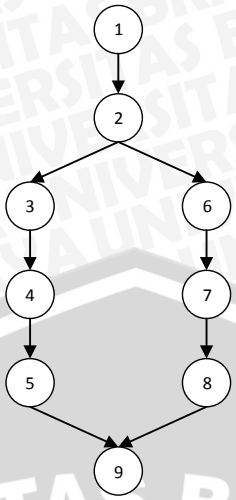
	ketika nyawa hanya tersisa satu		
3	*Player* menabrak *obstacle* atau tembakan dari boss	Nyawa berkurang satu	Nyawa berkurang satu
4	*Player* menabrak musuh	Permainan berakhir	Permainan berakhir
5	*Player* membunuh boss ketika nyawa kurang dari tiga	Permainan berakhir, *player* mendapatkan *reward* berupa enzim, dan level selanjutnya terbuka	Permainan berakhir, *player* mendapatkan *reward* berupa enzim, dan level selanjutnya terbuka
6	*Player* membunuh boss ketika nyawa masih utuh	Permainan berakhir, *player* mendapatkan *reward* berupa enzim, dan level selanjutnya terbuka	Permainan berakhir, *player* mendapatkan *reward* berupa enzim, dan level selanjutnya terbuka

5.1.2 Pengujian Pemilihan Karakter

Method Update

```
START ——— 1
Get the choosen characters ——— 2
IF chosen characters are correct ——— 3
    player can kill mini boss with one shot ——— 4
    set score for killing bos 50 ———
END IF ——— 5
ELSE ——— 6
    player has to shoot 3 times to kill mini boss ——— 7
    no additional score will be added when killing the boss ———
END IF ——— 8
END ——— 9
```

Gambar 5.3 Pengujian pemilihan karakter



Gambar 5.4 *Flow Graph* pemilihan karakter

V(G)=P+1

V(G)=1+1

V(G)=2

Jalur independen:

1. 1-2-3-4-5-9
2. 1-2-6-7-8-9

Pada tabel 5.3 adalah hasil pengujian yang dilakukan untuk mengetes unit pemilihan karakter. Jalur diperoleh dari perhitung kompleksitas siklomatik dari method Update. Ada 2 kasus uji sesuai dengan jalur yang diperoleh pada perhitungan sebelumnya yang dijabarkan pada tabel 5.2.

Tabel 5.2 Hasil Unit Pemilihan Karakter

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Pemilihan karakter sesuai dengan level yang dimainkan	*Player* dapat membunuh mini boss dengan sekali tembak, dan jika membunuh boss mendapatkan bonus nilai.	*Player* dapat membunuh mini boss dengan sekali tembak, dan jika membunuh boss mendapatkan bonus nilai.
2	Pemilihan karakter tidak sesuai dengan level yang dimainkan	*Player* membutuhkan tiga tembakan untuk membunuh mini boss, dan jika membunuh boss	*Player* membutuhkan tiga tembakan untuk membunuh mini boss, dan jika membunuh boss

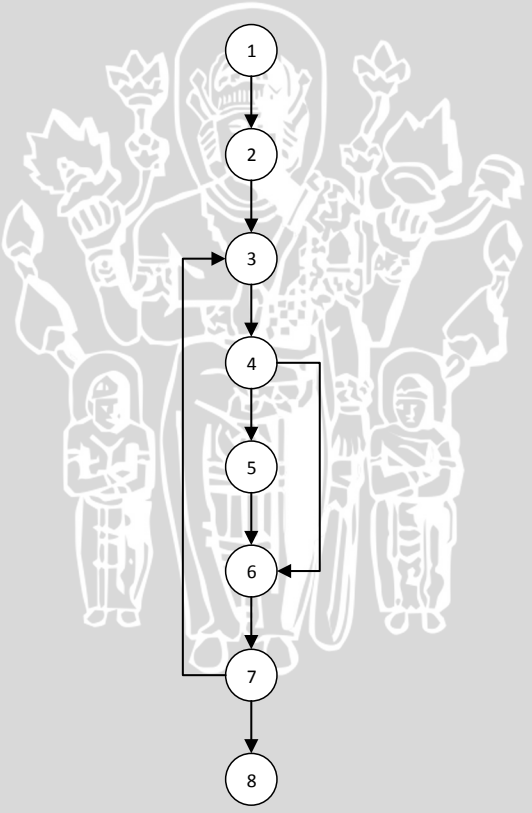
		mendapatkan tidak bonus nilai.	mendapatkan tidak bonus nilai.

5.1.3 Pengujian Membuka Level

Method UnlockLevel

```
START ——— 1
Get current  level name ——— 2
FOR i=0, i<8, increment 1 ——— 3
    IF (current level name equals with "Level_"+i) ——— 4
    THEN
        Change next level state to unlocked ——— 5
    END IF ——— 6
END FOR ——— 7
END ——— 8
```

Gambar 5.5 Pengujian Membuka Level



Gambar 5.6 *Flow Graph* Method UnlockLevel

V(G)=E-N+2

V(G)=9-8+2

V(G)=3

Jalur independen:

1. 1-2-3-4-5-6-7-8
2. 1-2-3-4-5-6-7-3-4-5-6-7-8
3. 1-2-3-4-6-7-8

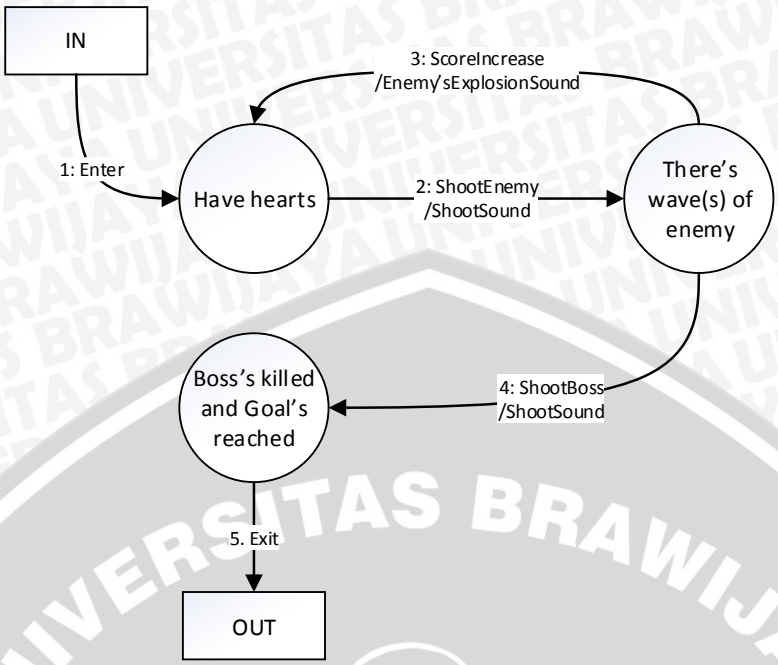
Pada tabel 5.3 adalah hasil pengujian yang dilakukan untuk mengetes unit membuka level. Jalur diperoleh dari perhitung kompleksitas siklomatik dari method UnlockLevels. Ada 3 kasus uji sesuai dengan jalur yang diperoleh pada perhitungan sebelumnya yang dijabarkan pada tabel 5.3.

Tabel 5.3 Hasil Unit Penyimpanan Data Level

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Level 1 berhasil Diselesaikan	Level 2 pada halaman pilih level terbuka	Level 2 pada halaman pilih level terbuka
2	Level berhasil diselesaikan	Level selanjutnya pada halaman pilih level terbuka	Level selanjutnya pada halaman pilih level terbuka
3.	Level tidak berhasil diselesaikan	Level selanjutnya pada halaman pilih level tidak terbuka	Level selanjutnya pada halaman pilih level tidak terbuka

5.2 Hasil Pengujian Dengan *Black Box* Testing

Pengujian black box dilakukan dengan metode *Test Flow Diagram* (TFD). TFD adalah model grafis yang merepresentasikan *behavior* dari perspektif pemain. TFD menyediakan pendekatan formal untuk mengetes sebuah desain.



Gambar 5.3 Pengujian Gameplay Nutriman Evolution dengan TFD

Test case yang dihasilkan :

1. Termintor "In", Event: Enter
   a. Menjalankan *game* pada android
   b. Memilih tombol main
   c. Memilih level untuk memulai permainan
2. Event "ShootEnemy", Action: ShootSound
   a. Menembak musuh (mini boss) dan memainkan suara tembakan
3. Event "ScoreIncrease", Action: Enemy'sExplosionSound
   a. Menambah nilai dan memainkan suara musuh yang meledak.
4. Event "ShootBoss", Action: ShootSound
   a. Menembak boss dan memainkan suara tembakan
5. Event "Exit"
   a. Mengakhiri proses
6. Terminator "Out"
   a. Keluar dari kondisi "Boss's Killed and Goal's reached"

Pada pengujian dengan TFD hanya dihasilkan satu jalur. Hasil pengujian yang diperoleh berdasarkan kasus uji dari jalur yang didapatkan bisa dilihat pada tabel 4.3

Tabel 5.2 Pengujian Pada *Gameplay*

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Pengujian prosedur pada *gameplay*	Menghasilkan hasil yang sesuai pada kebutuhan dan fungsionalitas tanpa ada *bug* pada setiap prosedur.	Menghasilkan hasil yang sesuai pada kebutuhan dan fungsionalitas tanpa ada *bug* pada setiap prosedur.

5.3 Hasil Pengujian Kepada Pengguna (*Focus Testing*)

Pengujian kepada penguna digunakan untuk memastikan bahwa *game* yang dibuat telah sesuai dengan tujuan saat perancangan, yaitu membuat *game* edukasi yang tidak hanya menyenangkan saat dimainkan tetapi juga mengedukasi. Pengujian dilakukan kepada siswa SMP PGRI 1 Buduran Sidoarjo. Tabel 5.4 menunjukkan hasil kuesioner pengujian yang diisi oleh 20 orang siswa.

Tabel 5.5 Hasil Pengujian Game Terhadap Beberapa Siswa SMP

No	Pertanyaan	Jumlah Jawaban	
		Ya	Tidak
1.	Apa kamu suka bermain game?	20	0
2.	Apakah game ini menarik?	19	1
3.	Apakah setelah bermain game ini kamu mendapatkan pengetahuan baru?	19	1
4.	Apakah game ini dapat membantumu dan memudahkan dalam pelajaran IPA bab sistem pencernaan manusia?	19	1
5.	Apakah kamu ingin menyeleseikan game ini jika sudah dikembangkan lagi?	18	2

5.4 Analisa Pengujian

Berikut ini adalah pembahasan analisa pengujian game edukasi "Nutriman Evolution". Pengujian yang dilakukan ada dua jenis yaitu *white box* dan *black box*. Pengujian *white box* dilakukan dengan pengujian unit menggunakan metode *basic path testing*. Sedangkan untuk metode *black box* dilakukan dengan metode *test flow diagram*, dan pengujian terhadap pengguna (*focus testing*).

5.4.1 Hasil Analisa *White Box Testing*

Berdasarkan hasil pengujian unit yang dilakukan, jumlah jalur pada logika setiap *method* sesuai dengan pehitungan *cyclomatic complexity*. Setiap kasus uji yang dibuat telah sesuai berdasarkan jumlah jalur yang telah diuji dan memberikan hasil sesuai dengan yang diharapkan. Dari hasil tersebut dapat ditarik

kesimpulan bahwa unit modul dari program sudah memenuhi kebutuhan fungsional yang telah dirancang pada tahap perancangan.

5.4.2 Hasil Analisa *Black Box Testing*

Berdasarkan kesesuaian antara hasil uji terhadap implementasi dan fungsionalitas setiap prosedur yang diuji pada "Nutriman Evolution", menghasilkan output yang diharapkan dalam daftar kebutuhan game ini. Maka dapat disimpulkan bahwa implementasi dan fungsionalitas game telah memenuhi kebutuhan yang telah dijabarkan dalam daftar kebutuhan. Hasil Analisa *Focus Testing*

Berdasarkan hasil pengujian yang dilakukan terhadap siswa SMP PGRI 1 Buduran Sidoarjo game edukasi "Nutriman Evolution" dapat diterima dengan cukup baik yang bisa dilihat pada tabel 5.4. Jumlah jawaban "ya" pada semua pertanyaan melebihi 90%. Jawaban "tidak" pada semua pertanyaan tidak melebihi 10%.

Tabel 5.6 Analisa Hasil Pengujian Terhadap Pengguna

No	Pertanyaan	Jumlah Jawaban	
		Ya	Tidak
1.	Apa kamu suka bermain game?	100%	0%
2.	Apakah game ini menarik?	95%	5%
3.	Apakah setelah bermain game ini kamu mendapatkan pengetahuan baru?	95%	5%
4.	Apakah game ini dapat membantumu dan memudahkan dalam pelajaran IPA bab sistem pencernaan manusia?	95%	5%
5.	Apakah kamu ingin menyeleseikan game ini jika sudah dikembangkan lagi?	90%	10%

Dari hasil yang didapatkan bisa ditarik kesimpulan bahwa *game* telah sesuai dan memenuhi hasil yang diharapkan. *Game* "Nutriman Evolution" bisa dijadikan sarana untuk memperkenalkan dan mempelajari sistem pencernaan manusia.

BAB 6 PENUTUP

6.1 Kesimpulan

Berdasarkan hasil analisa, implementasi, dan pengujian yang telah dilakukan, bisa diambil kesimpulan sebagai berikut:

1. Berdasarkan analisa dan proses perancangan dengan paper prototype, *gameplay* yang digunakan untuk game edukasi biologi SMP pengenalan sistem pencernaan manusia adalah *shooter*. *Gameplay* ini mengharuskan *player* untuk menembak musuh dan menghindari *obstacle*. Setiap musuh yang ditembak akan menambah nilai akhir. Di beberapa level terdapat boss di bagian akhirnya yang menyimpan enzim. Enzim ini digunakan untuk mengubah zat makanan.
2. Game ”Nutriman Evolution” berhasil diimplemetasikan menggunakan game engine Unity dengan bahasa pemrograman C#.
3. Berdasarkan hasil pengujian yang telah dilakukan, didapatkan hasil bahwa fungsionalitas game edukasi biologi SMP pengenalan sistem pencernaan manusia telah terpenuhi.
4. Berdasarkan pengujian terhadap pengguna, game *“*Nutriman*”* mendapat tanggapan yang positif yang bisa dilihat dari jawaban yang diberikan setelah memainkan permainan. Sekitar lebih dari 90% pengguna menyatakan bahwa game “Nutriman Evolution” menyenangkan untuk dimainkan dan memberikan informasi tambahan seputar sistem pencernaan manusia.

6.2 Saran

Saran yang bisa diberikan untuk pengembangan selanjutnya adalah:

1. Dibuat *power-up* agar permainan semakin menarik.
2. Dibuat level *endless* dibagian akhir menggunakan zat-zat yang tidak bisa dipakai di level-level sebelumnya.

DAFTAR PUSTAKA

Adnyana, P. B. & Citrawathi, D. M., 2008. Pengembangan Modul Biologi Berorientasi Siklus Belajar untuk Meningkatkan Penalaran dan Keterampilan Inkuiri Siswa SMA. *Jurnal Penelitian dan Pengembangan,* Issue 3.

Ault, M., 2004. *How Games Can Engage Students and Improve Learning.* [Online] Available at: http://www.eschoolnews.com/2014/06/06/games-engage-students-241/
[Diakses 3 December 2015].

Brodkin, J., 2013. *How Unity3D Became a Game-Development Beast.* [Online] Available at: http://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/
[Diakses 10 November 2015].

Carlson, B. M., 2004. Human Embryology and Developmental Biology, 3rd edition. In: Saint Louis: Mosby.

Citrawathi, D. M., 2006. Pengembangan Pembelajaran Biologi dengan Menggunakan Modul Berorientasi Siklus Belajar dan Pengaruhnya Terhadap Hasil Belajar Siswa di SMA.

Deutsch, M., Jensen, R. & Tonies, C., 1979. "Verification and Validation," in Software Engineering. In: US: Prentice-Hall, p. 329–40.

Hejlsberg, A., Torgersen, M., Wiltamuth, S. & Golde, P., 2011. *The C# Programming Language.* 4th ed. Boston: Addison-Wesley Professional.

Hunicke, R., LeBlanc, M. & Zubek , R., 2004. MDA: A Formal Approach to Game Design and Game Research.

Hunicke, R., LeBlanc, M. & Zubek, R., 2004. MDA: A Formal Approach to Game Design and Game Research. *the AAAI Workshop on Challenges in Game AI (Vol. 4)..*

Keesee, G. S., 2011. *Educational Games.* [Online] Available at: http://teachinglearningresources.pbworks.com/w/page/35130965/Educational%20Games
[Diakses 17 Desember 2015].

Myers, G. J., 2004. *The Art of Software Testing.* 2nd ed. New Jersey: John Wiley & Sons.

Pressman, R. S., 2001. *Software Engineering: A Practitioner's Approach, 7th edition.* s.l.:Mc Graw Hill.

Pressman, R. S. P., 2001. *Software Engineering, A Practitioner's Approach.* 5th ed. New York: Thomas Casson.

Rhône, R. d., 2006. *The C# Programming Language (Covering C# 4.0).* 4 ed. Geneva: Ecma International.

Riccitiello, J., 2014. *John Riccitiello sets out to identify the engine of growth for Unity Technologies* [Interview] (23 Oktober 2014).

Roedavan, R., 2014. *UNITY Tutorial Game Engine.* Bandung: Informatika.

Rogers, S., 2010. Level Up! Guide to Great Video Game Design. In: US: Wiley, p. 9.

Schlutz, C. P., Bryant, R. & Langdell, T., 2005. *Game Testing All in One.* Boston: Thomson Course TEchnology.

Schreiber, I., 2009. *Game Design Concepts.* New York: Creative Commons Attribution 3.0.

Schultz, C. P., Bryant, R. & Langdell, T., 2005. *Game Testing All in One.* 2 ed. US: Thomson/Course Technology.

Sunyono, 2005. *Optimalisasi Pembelajaran Kimia Kelas XI SMA Swadipa Natar Melalui Penerapan Metode Eksperimen Menggunakan Bahan-Bahan yang ada Di Lingkungan Laporan Penelitian Tindakan Kelas,* Lampung: Fakultas Keguruan dan Ilmu Pendidikan Universitas Lampung..

Warsita, 2008. *Teknologi Pembelajaran Landasan dan Aplikasinya.* Jakarta: Rhineka Cipta.