**RANCANG BANGUN GAME EDUKASI PENGENALAN DASAR PEMROGRAMAN KOMPUTER BERBASIS ANDROID**

**TUGAS AKHIR**



**FELIX TANDIONO**

**311110024**

**PRODI TEKNIK INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS MA CHUNG**

**JANUARI**

**2015**

**BAB I**

**PENDAHULUAN**

1. **Latar Belakang**

Perkembangan bidang teknologi informasi yang semakin pesat membuat semakin dibutuhkannya tenaga-tenaga terlatih dibidang teknologi informasi. Namun, pertumbuhan kebutuhan tenaga terlatih tersebut tidak diikuti dengan pertambahan jumlah tenaga terlatih yang siap untuk memenuhi kebutuhan tersebut. Menurut data yang diambil oleh Badan Ketenagakerjaan Amerika Serikat (*U.S.* *Bureau of Labor Statistics*) dalam kurun waktu 2010 hingga 2012, setiap tahunnya kebutuhan terhadap tenaga terlatih dibidang teknologi informasi meningkat sebanyak 136.620 pekerjaan, dengan rata-rata gaji per tahun $80.000 untuk setiap pekerja dibidang teknologi informasi. Sedangkan data dari *National Science Foundation* Amerika Serikat menunjukkan bahwa jumlah lulusan universitas dengan gelar yang berhubungan dengan bidang teknologi informasi hanya sekitar 40.000 siswa per tahunnya. Di Asia tenggara termasuk di Indonesia sendiri, menurut survei yang dilakukan oleh situs pencari lapangan pekerjaan jobstreet.com, permintaan akan tenaga ahli di bidang teknologi informasi semakin hari semakin meningkat, namun permintaan yang terus meningkat tersebut masih sulit untuk terpenuhi dikarenakan kurangnya pencari pekerjaan yang mumpuni dibidang teknologi informasi.

Salah satu tenaga terlatih yang sedang dibutuhkan dibidang teknologi informasi saat ini adalah pengembang perangkat lunak. Menurut data yang dirilis oleh Badan Ketenagakerjaan Amerika Serikat, prospek perkembangan jumlah kebutuhan akan seorang pengembang perangkat lunak dalam kurun waktu 2012-2022 adalah sebesar 22% dengan rata-rata prospek perkembangan seluruh jenis pekerjaan secara global sebesar 11%. Seorang pengembang perangkat lunak adalah tenaga terlatih yang berada dibalik terciptanya program komputer, kemampuan dibidang pemrograman komputer adalah salah satu kemampuan yang wajib dimiliki oleh seorang pengembang perangkat lunak. Sedangkan, pemrograman komputer sendiri adalah proses untuk menuliskan kode-kode perintah komputer dalam bahasa yang dapat dimengerti komputer dan menggunakan tata bahasa maupun sintaks yang dapat diikuti olehnya sehingga komputer dapat melakukan tugas-tugas yang bermanfaat untuk manusia (Kingsley-Hughes, 2005).

Melihat fakta mengenai perkembangan kebutuhan tenaga terlatih dibidang teknologi informasi dan ketersediaan tenaga terlatihnya, terutama kebutuhan akan tenaga terlatih sebagai pengembang perangkat lunak maka pembelajaran pemrograman komputer akan sangat bermanfaat apabila diajarkan pada peserta didik sejak dini. Pengetahuan mengenai pemrograman akan membuka peluang karir bagi anak-anak didik dibidang teknologi informasi di masa depan. Selain itu menurut Golpin (2014), mempelajari logika pemrograman dapat meningkatkan kemampuan seseorang dalam berpikir secara analitis dan memecahkan masalah, sehingga pembelajaran pemrograman komputer tidak hanya bermanfaat untuk masa depan tetapi juga akan memberikan manfaat bagi peserta didik di masa sekarang.

Pada tingkat dasar, pembelajaran pemrograman komputer dapat dimulai dengan pengenalan peserta didik pada logika dan algoritma dasar pemrograman. Algoritma adalah daftar sekuensial dari instruksi yang digunakan untuk menyelesaikan sebuah masalah (Shiffman, 2009). Dengan dasar algoritma yang baik, peserta didik kemudian dapat melanjutkan pembelajaran dengan menulis *pseudocode* (bahasa manusia yang dituliskan menggunakan tata bahasa komputer) sebelum kemudian menulis kode komputer yang sebenarnya.

Meskipun pembelajaran pemrograman komputer bermanfaat bagi peserta didik, pemrograman komputer masih jarang diajarkan di kebanyakan sekolah. Menurut data code.org, 9 dari 10 sekolah publik tidak menawarkan kelas yang berhubungan dengan pemrograman komputer. Hal ini membuat pemrograman komputer menjadi bidang yang asing bagi siswa dan kemudian berimplikasi pada munculnya pendapat bahwa pemrograman komputer adalah hal yang sulit untuk dipelajari. Oleh karenanya dibutuhkan metode pembelajaran dan pengenalan materi pemrograman yang lebih mudah diserap oleh siswa dan dapat digunakan sebagai media pembelajaran di sekolah publik, sehingga pada akhirnya dapat mengubah pendapat yang sudah terlanjur ada mengenai pemrograman komputer.

Dalam mengatasi masalah kurangnya minat peserta didik dalam proses belajar mengajar, *game* berbasis edukasi dapat menjadi solusi. Menurut Wastiau, dan kawan kawan (2009), penggunaan *game* dalam proses belajar mengajar memungkinkan peserta didik untuk berperan aktif (sebagai pengguna) dan mampu memberikan tujuan yang jelas dalam setiap tindakan yang diambil peserta didik, hal ini berdampak pada meningkatnya motivasi peserta didik dalam mengikuti dan memahami materi pembelajaran yang disampaikan. Berdasarkan hal tersebut maka dapat diusulkan sebuah *game* edukasi yang bertujuan untuk mengajarkan logika dan algoritma pemrograman untuk peserta didik di usia dini.

1. **Identifikasi Masalah**

Pertumbuhan kebutuhan akan tenaga terlatih dibidang teknologi informasi terutama pengembang perangkat lunak akan sangat berkembang dimasa depan, namun pertumbuhan kebutuhan ini tidak diiringi dengan pertumbuhan ketersediaan tenaga terlatihnya, hal ini dikarenakan pembelajaran mengenai dasar pemrograman komputer untuk peserta didik pada usia dini masih dinilai sulit dan kurang menarik.

1. **Batasan Masalah**

Adapun batasan masalah pada penelitian ini adalah sebagai berikut:

1. Peserta didik yang menjadi sasaran adalah peserta didik yang sedang menempuh pendidikan di tingkat 4 - 6 sekolah dasar.
2. *Game* yang dibuat berbasis sistem operasi android dengan menggunakanGodot *Game* Engine.
3. Perangkat android yang dapat menjalankan *game* memiliki sistem operasi Android versi 4.0 atau lebih tinggi.
4. **Rumusan Masalah**

Bagaimana merancang sebuah media pembelajaran yang dapat menarik minat peserta didik dan mempermudah proses pembelajaran dasar pemrograman komputer.

1. **Tujuan**

Menciptakan sebuah *game* edukasi berbasis android yang dapat menarik minat peserta didik dan mempermudah proses pembelajaran dasar pemrograman komputer.

1. **Manfaat**

Manfaat yang akan diperoleh dengan adanya penelitian ini antara lain:

1. Bagi guru, mendapat media pengajaran dasar pemrograman komputer yang menarik, sehingga dapat menjadi alat bantu untuk mengenalkan dasar pemrograman komputer kepada siswa
2. Bagi siswa, dengan terciptanya media pembelajaran dasar pemrograman komputer yang menarik, siswa tertarik untuk mengenal dasar pemrograman komputer sejak usia dini, sehingga dapat mempermudah apabila siswa ingin mendalami pemrograman komputer di masa depan
3. Bagi penelitian selanjutnya, dapat memberikan gambaran dan dasar untuk melakukan penelitian lebih lanjut mengenai media pembelajaran dasar pemrograman komputer.

**1.7 Sistematika Penulisan**

Bab 1: Pendahuluan

Bab ini menjelaskan latar belakang diangkatnya judul tugas akhir yaitu adanya permasalahan dalam pembelajaran dasar pemrograman komputer untuk peserta didik di usia dini. Selain itu juga dibahas identifikasi masalah, batasan masalah, rumusan masalah, tujuan, dan manfaat penelitian yang dilakukan.

Bab II: Tinjauan Pustaka

Bab ini menjelaskan mengenai dasar-dasar teori yang digunakan dalam proses desain dan pembuatan *game* edukasi seperti pengertian umum mengenai pemrograman komputer, elemen-elemen yang digunakan dalam pemrograman komputer dan teknologi, bahasa pemrograman, maupun *framework* yang akan digunakan dalam proses pembuatan *game* edukasi.

Bab III: Analisis dan Perancangan Sistem

Bab ini menjelaskan mengenai analisa dan desain sistem *game* edukasi yang akan dibuat. Analisa yang dilakukan terdiri dari analisa akan masalah dan analisa akan kebutuhan, sedangkan pada sub bab desain terdiri dari penjabaran konsep *game*, desain sistem dan desain antarmuka.

Bab IV: Hasil dan Pembahasan

Bab ini menjelaskan mengenai analisa dan desain sistem *game* edukasi yang akan dibuat. Analisa yang dilakukan terdiri dari analisa akan masalah dan analisa akan kebutuhan, sedangkan pada sub bab desain terdiri dari penjabaran konsep *game*, desain sistem dan desain antarmuka.

Bab V: Kesimpulan dan Saran

Bab ini menjelaskan mengenai analisa dan desain sistem *game* edukasi yang akan dibuat. Analisa yang dilakukan terdiri dari analisa akan masalah dan analisa akan kebutuhan, sedangkan pada sub bab desain terdiri dari penjabaran konsep *game*, desain sistem dan desain antarmuka.

**1.8 Jadwal Penelitian**

Jadwal penelitian disusun guna memastikan jalannya pelaksanaan penelitian secara tepat waktu. Jadwal penelitian ini direpresentasikan dalam bentuk tabel sebagai berikut:

Tabel 1.1 Jadwal Penelitian

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Kegiatan** | **Februari** | | | | **Maret** | | | | | **April** | | | | | **Mei** | | | | | **Juni** | | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |
| Studi Literatur |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Penulisan proposal |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Pembuatan Aplikasi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Penulisan laporan TA |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Revisi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**BAB II**

**TINJAUAN PUSTAKA**

1. **Pemrograman Komputer**

Menurut Kingsley-Hughes (2005), pemrograman komputer adalah proses untuk menuliskan kode-kode perintah komputer dalam bahasa yang dapat dimengerti komputer dan menggunakan tata bahasa maupun sintaks yang dapat diikuti olehnya sehingga komputer dapat melakukan tugas-tugas yang bermanfaat untuk manusia. Dalam pemrograman komputer, *programmer* selalu memerintahkan komputer untuk melakukan sesuatu yang kemudian dijalankan oleh komputer satu per satu secara sekuensial. Kode perintah yang ditulis juga harus tepat dan tidak bersifat ambigu.

Menurut Shaun Bebbington (2014), pemrograman komputer adalah sebuah cara bagi manusia untuk memberikan perintah kepada mesin untuk melakukan tugasnya, memecahkan masalah, dan untuk menciptakan interaksi antara manusia dan mesin, sehingga mesin dapat berguna bagi manusia.

* + 1. **Bahasa Pemrograman**

Dalam pemrograman komputer tidak digunakan bahasa manusia, karena pada dasarnya komputer bersifat mati otak, oleh karenanya untuk membuat program komputer, seorang *programmer* akan menggunakan bahasa yang dimengerti oleh komputer, bahasa-bahasa ini disebut dengan bahasa pemrograman (Wang, 2011)

Pada dasarnya, komputer hanya mengetahui bahasa mesin yang terdiri dari nilai biner (0 atau 1). Menulis program dalam bahasa mesin akan memakan waktu yang sangat lama dan membutuhkan ketelitian yang tinggi oleh karenanya diciptakan bahasa pemrograman yang lebih sederhana dan dapat dimengerti manusia yang disebut bahasa *assembly*.

Bahasa *assembly* memberikan kemudahan bagi *programmer* untuk membaca, menulis, maupun menyunting program komputer, karena sintaks atau kosakata yang digunakan dalam bahasa *assembly* sudah menyerupai bahasa manusia. Contoh program komputer menggunakan bahasa assembly terdapat pada gambar 2.1

1 title Hello World

2 ; This program displays “Hello World!” on the screen

3 dosseg

4 .model small

5 .stack 100h

6 .data

7 my\_message db ‘Hello World!’.0dh.0ah.’$’

8 .code

9 main proc

10 mov [ax.@data](mailto:ax.@data)

11 mov ds.ax

12 mov ah.9

13 mov dx.offset my\_message

14 int 21h

15 mov ax.4C00h

16 int 21h

17 main endp

18 end main

Gambar 2.1 Contoh program sederhana menggunakan Bahasa Assembly

Namun bahasa assembly masih dinilai tidak cukup mudah bagi *programmer* untuk membuat program yang menjalankan tugas-tugas rumit dan banyak, oleh karenanya diciptakan berbagai bahasa pemrograman tingkat tinggi seperti COBOL, FORTRAN, BASIC, dan C. Bahasa pemrograman tingkat tinggi semakin memudahkan *programmer* untuk membuat program komputer yang lebih kompleks, selain itu juga memudahkan *programmer* pemula untuk mempelajari pemrograman komputer. Contoh program komputer yang ditulis menggunakan bahasa C terdapat dalam gambar 2.2.

1 main()

2 {

3 printf (“Hello world!\n”);

4 }

Gambar 2.2 Contoh program sederhana menggunakan Bahasa C

* + 1. **Konsep *Control Flow***

Komputer pada umumnya menjalankan sebuah program menurut urutan dan struktur tertentu. Struktur dan urutan ini dikenal dengan istilah *control flow*. *Control flow* untuk tiap-tiap bahasa pemrograman umumnya tidak terlalu berbeda, hal ini ditujukan agar program yang dihasilkan menggunakan tiap bahasa pemrograman dapat lebih terorganisir, mudah untuk dimodifikasi (penambahan maupun pengurangan), dan lebih modular. Beberapa elemen-elemen dalam konsep *control flow*:

1. Variabel, adalah elemen yang menyimpan suatu nilai atau data, variabel dapat menyimpan nilai dasar seperti angka hingga sebuah objek yang terdiri dari banyak nilai
2. Perulangan, adalah elemen yang mengulang sekumpulan perintah, contohnya While loop dan For loop
3. Kondisional, adalah elemen yang menjalankan sekumpulan perintah apabila sebuah kondisi terpenuhi, biasa dituliskan dengan format If Then Else
4. Input/Output, adalah elemen yang berinteraksi dengan entitas diluar program, misalnya mencetak teks di layar, menerima input user, maupun membaca atau menulis file.
5. Subrutin (fungsi dan prosedur), adalah elemen yang digunakan untuk mengelompokkan sekumpulan perintah kedalam sebuah grup, sehingga dapat digunakan lagi dan lagi.
   * 1. **Proses dalam Pemrograman Komputer (*Programming Practices*)**

Dalam pemrograman komputer, seorang *programmer* secara sadar maupun tidak sadar akan melakukan langkah-langkah praktis untuk mendapatkan solusi atas permasalahan yang dihadapinya. Langkah-langkah praktis tersebut antara lain, *requirements analysis* atau mendefinisikan permasalahan yang sedang dihadapinya, *design* atau perencanaan, *coding* atau memprogram, *testing* atau uji coba, dan *debugging* atau memperbaiki bug atau kesalahan logika pada program yang telah dibuat:

* + - 1. *Requirements analysis*, adalah tahap awal dalam proses pemecahan masalah dalam pemrograman komputer. Pada tahap ini, *programmer* menganalisa dan mendefinisikan masalah yang akan dihadapi.
      2. *Design,* pada tahap ini *programmer* akan melakukan perencanaan solusi terhadap permasalahan yang akan dihadapi.
      3. *Coding*, pada tahap ini *programmer* melakukan pengkodean sesuai dengan solusi yang telah dirancang pada tahap *design*.
      4. *Testing*, pada tahap ini programmer melakukan uji coba kode yang telah dihasilkan terhadap hasil yang menjadi ekspektasi.
      5. *Debugging*, apabila kode yang dihasilkan tidak memenuhi ekspektasi, tahap *debugging* dilakukan. Pada tahap ini programmer akan melakukan evaluasi terhadap kode, menjalankan lagi kode yang sudah ada, dan mencoba mencari kesalahan yang terdapat dalam kode.

1. **Serious Game/Game Edukasi**

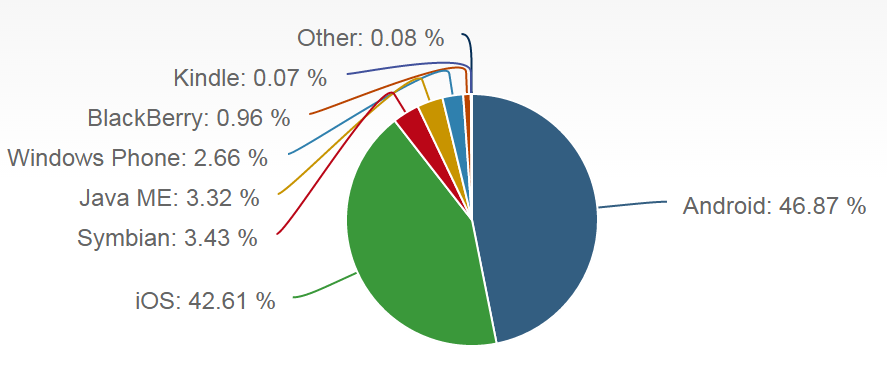
Menurut Zyda (2005), *game* adalah sebuah perlombaan secara fisik maupun psikis yang dijalankan menurut suatu aturan yang spesifik dengan tujuan untuk memberikan kesenangan bagi para pelakunya. Serious *game*, menurut Zyda adalah sebuah *game* yang bertujuan menggunakan kesenangan bermain *game* untuk lebih lanjut memberikan pelatihan, edukasi, maupun manfaat dan tujuan-tujuan lain bagi pelakunya. Menurut Iuppa dan Borst (2012), serious *game* adalah *game* yang didesain untuk menjadi medium untuk:

1. Penyaluran dan penguatan bagi pengetahuan maupun kemampuan praktis
2. Teknik dan konten yang bersifat persuasif yang ditujukan untuk mengubah perilaku sosial maupun personal penggunanya, termasuk *game* yang ditujukan sebagai media promosi, marketing, maupun perekrutan.
3. **Tools dan Library yang Digunakan**

Dalam mengembangkan sebuah *game*, diperlukan *tools* maupun *library* untuk mempermudah proses pengembangan. Demikian juga dalam penelitian ini, berikut tools dan library yang digunakan dalam penelitian ini.

1. **Android**

Android adalah sekumpulan perangkat lunak yang dikembangkan oleh Google untuk perangkat mobile. Android menyediakan aplikasi, sebuah *virtual machine* (Dalvix Virtual Machine), *middleware* (perangkat lunak yang menjadi perantara antara sistem operasi dengan *virtual machine* dan aplikasinya), dan sebuah sistem operasi yang berbasis Linux (Friesen, 2014). Dalam pengembangannya Android memiliki banyak versi, hingga versi terbarunya saat ini yaitu Android 4.5 dengan *codename* Lollipop yang juga dapat digunakan untuk televisi pintar dan jam pintar. Menurut data yang diambil oleh lembaga Net Applications pada Februari 2015, Android adalah sistem operasi untuk perangkat *mobile* dengan jumlah pengguna terbanyak dengan *market share* sebesar 46,87% (Gambar 2.4).



(Sumber : Net Applications, 2015)

Gambar 2.4 Android memiliki jumlah pengguna terbanyak

1. **Sejarah Android**

Sistem Operasi Android awalnya dikembangkan oleh perusahaan Android Inc. yang berbasis di Palo Alto California. Pada awal mula berdirinya, para pendiri perusahaan Android Inc. berniat untuk mengembangkan sistem operasi yang lebih canggih bagi kamera digital. Menyadari kurang besarnya pasar kamera digital, perusahaan ini kemudian memfokuskan diri ke pengembangan sistem operasi bagi ponsel pintar dengan harapan mampu menyaingi sistem operasi yang sudah ada pada waktu itu (Symbian dan Windows Phone). Google kemudian mengakuisi Android Inc. pada 17 Agustus 2005 dan semenjak itu Sistem Operasi Android terus berkembang pesat hingga saat ini.

1. **Versi Android**

Dalam pengembangannya hingga saat ini, Sistem Operasi Android telah memiliki banyak versi:

1. Android versi 1.0 dirilis pada 23 Septermber 2008
2. Android versi 1.1 dirilis pada 9 Februari 2009
3. Android versi 1.5 (Cupcake) dirilis pada 27 April 2009
4. Android versi 1.6 (Donut) dirilis pada 15 September 2009
5. Android versi 2.0 – 2.1 (Eclair) dirilis pada 26 Oktober 2009
6. Android versi 2.2 – 2.2.3 (Froyo/Frozen Yoghurt) dirilis pada 20 Mei 2010
7. Android versi 2.3 – 2.3.7 (Gingerbread) dirilis pada 6 Desember 2010
8. Android versi 3.0 – 3.2 (Honeycomb) dirilis pada 22 Februari 2011
9. Android versi 4.0 – 4.0.4 (Ice Cream Sandwich) dirilis pada 18 Oktober 2011
10. Android versi 4.1 – 4.3 (Jelly Bean) dirilis pada 9 Juni 2012
11. Android versi 4.4 (KitKat) dirilis pada 31 Oktober 2013
12. Android versi 5.0 – 5.1 dirilis pada 12 November 2014
13. **Godot Engine**

Godot Engine adalah sebuah *game engine* yang berorientasi visual dan didesain untuk dapat mengekspor *game* yang dihasilkan ke platform PC, Mobile maupun Web dengan mudah. Godot dapat digunakan untuk membuat *game* 2D maupun 3D. Godot menyediakan berbagai macam komponen dan *tools* yang memudahkan pengembang, sehingga pengembang dapat dengan cepat dan fokus dalam membangun *game-*nya.

1. **Sejarah Godot Engine**

Godot Engine dikembangkan oleh Juan Linietsky dan Ariel Manzur pada 2001 dengan disponsori oleh OKAM Studio. Godot Engine kemudian dirilis untuk publik dengan lisensi MIT pada Februari 2014, dengan lisensi ini, pengembang dimungkinkan untuk menggunakan Godot Engine untuk keperluan komersial. Pada Desember 2014, Godot Engine secara resmi merilis versi *stable* pertamanya (1.0).

1. **Fitur Godot Engine**

Dalam proses pengembangan *game* menggunakan Godot Engine, pengembang sangat terbantu dengan fitur-fitur lengkap yang dimiliki oleh Godot Engine.

1. *Multi-Platform*

Godot Engine sangat fleksibel karena dapat dijalankan pada Windows, OSX, maupun Linux, baik di sistem operasi 32 bits maupun 64 bits. Bagi pengguna sistem operasi Linux, Godot Engine tidak memerlukan *dependency* yang sangat banyak.

Godot Engine juga memungkinkan pengembang untuk men-*deploy* *game* yang dibuatnya dalam berbagai macam platform:

* Platform Mobile: iOS, Android, Windows Phone (*not stable*)
* Platform Desktop: Windows, OSX, Linux
* Platform Web: PNaCL, HTML5 (*not stable*)
* Konsol: Playstation 3 dan Playstation Vita

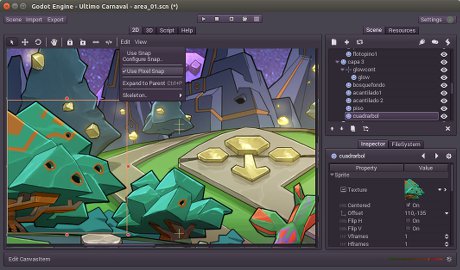


(Sumber : Godot Engine, 2015)

Gambar 2.6 Platform yang di-*support* oleh Godot Engine

1. *Visual Editor*

Godot Engine dilengkapi dengan *visual editor* yang sangat memudahkan pengembang dalam mengembangkan sebuah *game* dan dilengkapi dengan bermacam peralatan, pencari dokumentasi dan kode editor.



(Sumber : Godot Engine, 2015)

Gambar 2.5 Tampilan antarmuka editor Godot Engine

1. Mengembangkan *Game* 2D maupun 3D

Godot Engine menggunakan 2 *engine* terpisah yang dapat digunakan pengembang untuk mengembangkan *game* 2D maupun *game* 3D. Sehingga tidak perlu memalsukan efek 2D di ruang 3D seperti *game* *engine* lain. Engine 2D yang dimiliki oleh Godot Engine bekerja dengan koordinat *pixel* namun *game* yang dihasilkan dapat secara otomatis di resize sesuai resolusi layar. Selain itu *engine* 2D Godot juga mendukung *sprites*, *animated sprites*, *polygons*, *parallax*, *layers*, *particle system*, *engine* untuk *physics* dan *collision* dan lain lain.



(Sumber : Godot Engine, 2015)

Gambar 2.7 *Game* 2D yang dihasilkan menggunakan Godot Engine

Untuk mengembangkan *game* 3D, Godot Engine memiliki *engine* yang didesain sangat mudah untuk mengimpor aset 3D (Model dan animasinya) dari program editor model 3D seperti 3dsMax, Maya, Blender, dan lain-lain. Selain itu Engine 3D yang dimiliki Godot Engine juga mendukung *skeleton deforms*, *blend shapes*, berbagai macam tipe pencahayaan dengan *shadow mapping*, *shader* & *material* yang fleksibel, *rendering* dengan HDR, *anti-aliasing*, dan mode warna linear, Engine 3D Godot juga mendukung *post processing* seperti *fog*, *glow*, *bloom*, *color adjustment*, dan lain-lain.



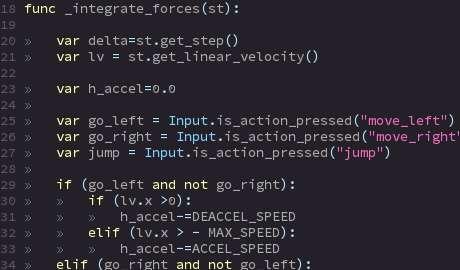
(Sumber : Godot Engine, 2015)

Gambar 2.8 *Game* 3D yang dihasilkan menggunakan Godot Engine

1. *Scripting*

Godot Engine memiliki bahasa *scripting* tersendiri yang disebut GDScript yang didesain berbasis bahasa Python dan dioptimalkan untuk membuat *game*. GDScript didesain secara efisien dalam menangani alokasi memori sehingga meminimalkan terjadinya kebocoran memori. GDScript juga didukung dengan adanya script editor dengan *syntax highlighting*, *real-time parser*, dan *code completion* yang sudah disertakan dalam editor Godot Engine.

Selain itu juga disediakan *Application Programming Interface* (API) untuk C++ untuk melakukan optimasi beberapa bagian dari *game* maupun untuk melakukan kustomisasi dan penambahan fitur untuk Godot Engine sendiri.

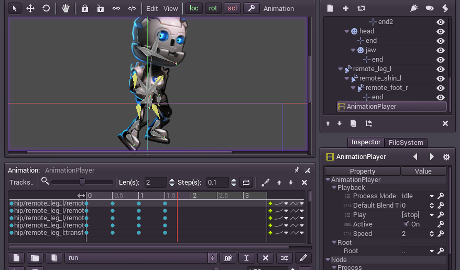


(Sumber : Godot Engine, 2015)

Gambar 2.9 Contoh script menggunakan GDScript

1. Animasi

Setiap properti dari tiap objek pada Godot Engine dapat di animasikan, termasuk pemanggilan fungsi. Selain itu sistem animasi Godot Engine juga mendukung transisi dan *tween*, helper untuk melakukan *rigging* pada animasi 2D dengan *skeletons* maupun IK, *Animation Tree* untuk melakukan animasi karakter yang lebih dinamis, dan lain-lain.



(Sumber : Godot Engine, 2015)

Gambar 2.10 Tampilan editor animasi dalam Godot Engine

1. *Physics Engine*

Godot Engine memiliki *physics engine* yang di desain khusus untuk mengembangkan *game*. Physics *engine* yang dimiliki oleh Godot Engine mendukung beberapa fitur seperti beberapa macam bentuk *collider*, KinematicBody, DynamicCharacter, termasuk beberapa fitur lain seperti simulasi kendaraan.



(Sumber : Godot Engine, 2015)

Gambar 2.11 Contoh simulasi kendaraan pada Godot Engine

1. Kolaborasi

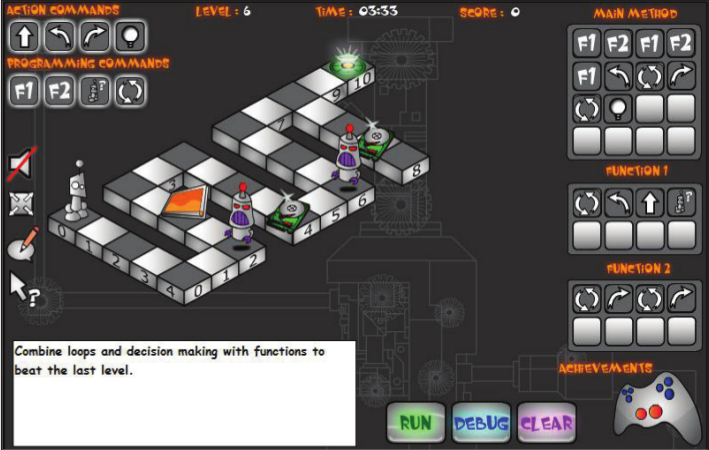
*File system* yang digunakan dalam Godot Engine didesain untuk dapat digunakan dalam proyek dengan banyak kontributor dan sangat mudah digunakan dengan *Version Control System* (VCS) seperti Git, Subversion dan lain-lain.

1. **Penelitian terdahulu**

Muratet dan kawan kawan (2008) dalam penelitiannya yang berjudul “*Learning Programming with an RTS based Serious Games*” mencoba menggabungkan *game* ber-*genre* *Real Time Strategy* (RTS) dengan pembelajaran pemrograman komputer. Dalam penelitiannya Muratet tidak menciptakan sendiri *game* edukasinya dari awal, Muratet menggunakan beberapa *game* RTS yang bersifat *open source* sebagai dasar. *Game* yang digunakan oleh Muratet sebagai dasar dalam penelitiannya adalah *Open Real-Time Strategy* dan *Spring*. Muratet kemudian menciptakan sebuah *dynamic library* yang bertugas untuk mentranslasikan kode-kode yang dituliskan oleh pengguna dalam sintaks C/C++ ke dalam aksi yang akan dilakukan oleh unit dalam *game*. Dalam kesimpulannya, Muratet menjabarkan perlunya sebuah *game* edukasi yang berdiri sendiri lengkap dengan *scenario editor* yang akan sangat berguna bagi guru dalam memberikan pembelajaran yang menyenangkan.

Muratet dan kawan-kawan (2010) melanjutkan penelitiannya pada 2008 dan berhasil mengembangkan sebuah *game* edukasi yang berjudul Prog&Play. *Prog&Play* adalah sebuah *game* RTS yang berbasis pada sebuah *game* RTS *open source* yang berjudul *Kernel Panic*. pengguna dapat berinteraksi dengan *game* menggunakan *Application Programming Interface* (API) *Prog&Play* yang tersedia dalam berbagai bahasa pemrograman seperti C, C++, Java, Ocaml, Ada dan Compalgo. Pengujian kemudian dilakukan dengan memilih 15 dari 196 mahasiswa Institute of Technology in Toulouse yang sedang menjalani studi pada semester awal di jurusan *computer science*. Dalam pemilihan mahasiswa yang akan digunakan sebagai sampel, Muratet menggunakan kuesioner untuk menilai motivasi dari tiap-tiap mahasiswa untuk belajar pemrograman komputer dan motivasi mereka dalam bermain *video game*. Mahasiswa yang diprioritaskan adalah mahasiswa yang tidak termotivasi untuk belajar pemrograman namun sangat termotivasi dalam bermain *video game*. Dalam penelitian ini pembelajaran dibagi dalam 2 fase, pada fase pertama mahasiswa diberikan kesempatan untuk memainkan *game* tanpa menggunakan pemrograman untuk memperkenalkan mekanisme *game* terlebih dahulu, pada fase kedua pengajar mempresentasikan cara kerja API Prog&Play. Pengujian kemudian dilakukan dengan membandingkan nilai ujian tengah semester dan nilai ujian akhir mahasiswa yang menggunakan *Prog&Play* dengan mahasiswa yang tidak menggunakan Prog&Play.

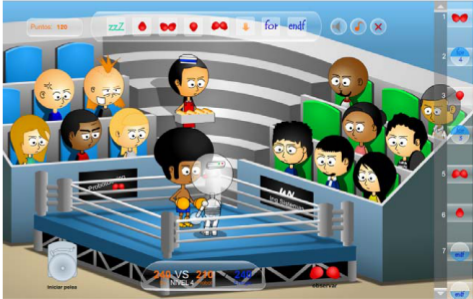
Penelitian yang dilakukan oleh Kazigmolu, dkk (2012) memberikan kesimpulan bahwa penggunaan *serious game* untuk memberikan pengenalan pemrograman komputer kepada peserta didik dengan sedikit atau tanpa pengalaman dalam pemrograman komputer adalah salah satu metode yang sangat efektif. Dalam penelitian yang berjudul “*A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming*” tersebut 25 mahasiswa dari University of Greenwich yang sedang menempuh mata kuliah pengenalan pemrograman komputer diminta memainkan *serious game* yang dibuat oleh Kazigmolu, dkk yang berjudul *Program Your Robot*. *Program Your Robot* adalah sebuah serious *game* yang didesain untuk memperkenalkan pemrograman komputer kepada para pengguna. Tujuan dari *Program Your Robot* adalah membantu sebuah robot untuk meloloskan diri dari rintangan-rintangan dengan menyediakan sebuah algoritma solusi. pengguna dapat membangun algoritma solusinya dengan memberikan perintah-perintah yang akan dijalankan oleh robot. Perintah yang digunakan dapat berupa perintah aksi seperti berjalan, berbelok ke kanan atau ke kiri dan melompat maupun perintah pemrograman seperti perulangan, subrutin, maupun kondisional. *Program Your Robot* mendapat respon yang sangat positif dari 25 mahasiswa tersebut, dan dinilai sebagai solusi yang tepat untuk memperkenalkan pemrograman komputer kepada peserta didik. Beberapa mahasiswa menyarankan penambahan sistem *achievement* dan *scoreboard* agar terdapat elemen kompetitif dalam *serious game* tersebut sehingga peserta didik akan lebih tertantang untuk mencari solusi terbaik bagi tiap-tiap masalah serta perbaikan pada beberapa bagian tampilan antarmuka agar lebih informatif.



(Sumber: Kazigmolu, 2012)

Gambar 2.13 Tampilan game edukasi Program Your Robot

Penelitian yang dilakukan oleh Moreno (2011) juga memberikan kesimpulan yang serupa. Dalam penelitiannya, Moreno menciptakan sebuah *game* edukasi berjudul *ProBot*. *ProBot* adalah sebuah *game* edukasi berbasis Flash yang digunakan untuk mengasah kemampuan pemrograman komputer pengguna. Dalam *ProBot*, pengguna akan bermain sebagai sebuah robot petinju yang bertugas untuk mengalahkan lawannya di arena permainan. Pengguna dapat memberikan instruksi kepada robot petinjunya melalui perintah-perintah seperti menghindar, tinju atas, bertahan, dan lain lain yang kemudian akan dijalankan oleh robot secara sekuensial. Untuk memenangkan pertandingan pengguna harus mempelajari pola gerakan robot lawan dan merancang algoritma yang sesuai. Permainan akan berakhir apabila salah satu petinju kehabisan *Resistance level* akibat terkena terlalu banyak pukulan atau kehabisan *Energy level* akibat melakukan terlalu banyak perintah. *ProBot* kemudian diujikan kepada 123 mahasiswa Universidad Nacional de Colombia yang sedang mengikuti kelas dasar pemrograman, 123 mahasiswa tersebut dibagi menjadi 2 kelompok secara acak untuk membentuk *control group* dan *test group*. Data mengenai nilai kemampuan dasar pemrograman kemudian diambil untuk kedua grup, hasilnya mahasiswa yang tergabung dalam *control group* memiliki nilai rata-rata 3.206 dari 5 dan mahasiswa dalam *test group* memiliki nilai rata-rata 3.315 dari 5. Setelah 4 minggu mengikuti materi pada kelas dasar pemrograman, mahasiswa pada kedua grup diberikan tugas rumah selama 2 minggu. Mahasiswa yang tergabung dalam *test group* juga berinteraksi dengan *ProBot* dalam kurun waktu yang sama. Setelah masa pengujian, mahasiswa kembali diberikan tes untuk mengukur efektifitas penggunaan *ProBot* pada nilai mahasiswa, hasilnya nilai rata-rata *control group* meningkat menjadi 3.471 dan nilai rata-rata test group yang menggunakan *ProBot* meningkat menjadi 3.847. Setelah pengujian selesai Moreno juga memonitor nilai ujian akhir 123 mahasiswa tersebut, hasilnya *control group* memiliki nilai rata-rata ujian akhir sebesar 3.251 sedangkan test group memiliki nilai rata-rata ujian akhir sebesar 3.642.



(Sumber: Moreno, 2011)

Gambar 2.14 Tampilan game edukasi ProBot

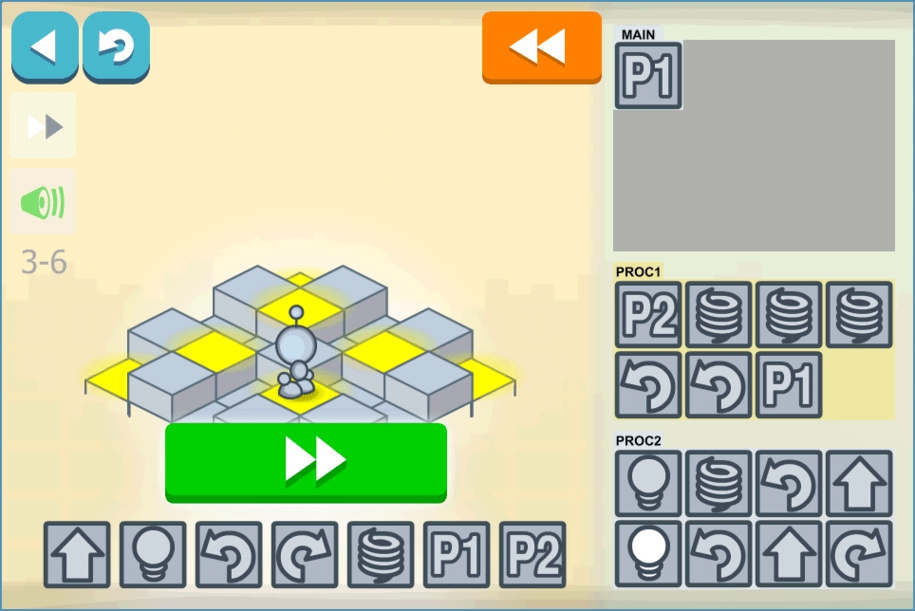
Penelitian yang dilakukan oleh Zhang dan Lu (2014) menghasilkan sebuah *game* edukasi berbasis mobile untuk pembelajaran pemrograman komputer yang berjudul iPlayCode. iPlayCode adalah sebuah *game* edukasi berbasis kuis untuk pengguna dengan sedikit atau tanpa pengalaman dibidang pemrograman komputer. Pada iPlayCode pengguna akan dihadapkan pada sebuah kode dan tombol benar salah. pengguna ditugaskan untuk melakukan analisa pada kode yang muncul di layar, apabila kode yang ditampilkan adalah kode yang *valid* maka pengguna diharuskan menekan tombol benar, dan sebaliknya jika kode tidak *valid* maka *pengguna* diharuskan menekan tombol salah. Dalam menjalankan tugasnya pengguna diberikan waktu tertentu, semakin sedikit waktu yang dibutuhkan oleh pengguna dalam menentukan jawaban maka semakin tinggi poin yang didapat oleh pengguna.



(Sumber: Zhang, 2014)

Gambar 2.15 Tampilan game edukasi iPlayCode

Lightbot adalah salah satu game edukasi yang ditujukan untuk memperkenalkan dasar pemrograman komputer (Yaroslavski, 2014). Lightbot diciptakan oleh Danny Yaroslavski menggunakan *framework* OpenFL dan bahasa pemrograman Haxe. Dalam tiap level pada game edukasi Lightbot, pengguna bertugas untuk membantu sebuah robot untuk menyalakan tiap petak biru yang terdapat pada level tersebut. Untuk melakukan hal tersebut, pengguna harus memprogram robot dengan instruksi-instruksi seperti maju, belok ke kiri, belok ke kanan, dan menyalakan petak. Instruksi-instruksi tersebut dilambangkan sebagai ikon-ikon sederhana. Lightbot sendiri menjadi salah satu game edukasi yang direkomendasikan dalam gerakan global Hour of Code yang diselenggarakan oleh Code.org pada 8-14 Desember 2014. Pada kegiatan yang bertujuan untuk menarik minat kaum awam untuk mempelajari pemrograman komputer dan teknologi informasi tersebut, Lightbot berhasil menarik minat 4.486.944 partisipan untuk mulai belajar dasar pemrograman komputer.



(Sumber: Lightbot, 2014)

Gambar 2.16 Tampilan game edukasi Lightbot

**BAB III**

**ANALISIS DAN PERANCANGAN SISTEM**

1. **Metodologi Penelitian**

Dalam melakukan penelitian, metodologi penelitian penting untuk didefinisikan. Hal ini ditujukan agar kegiatan penelitian lebih sistematis dan terarah. Demikian pula pada penelitian ini, berikut metodologi penelitian yang akan digunakan dalam penelitian ini:

1. Analisis

Pada tahapan analisis pengembang melakukan analisis terhadap kebutuhan yang diperlukan dalam rangka mengembangkan perangkat lunak, sifat dari perangkat lunak, tujuan dari dikembangkan perangkat lunak, dan performa yang diharapkan dari perangkat lunak itu sendiri.

1. Desain

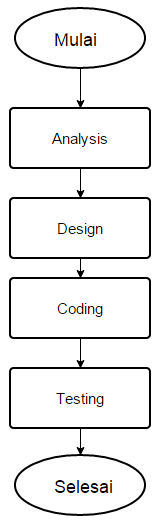
Pada tahapan desain seorang pengembang perangkat lunak mendefinisikan desain perangkat lunak yang akan dibuat. Dalam proses mendesain perangkat lunak terdapat empat atribut yang perlu diperhatikan yakni struktur data yang akan digunakan, arsitektur perangkat lunak yang akan dibuat, desain antarmuka untuk perangkat lunak, dan detail mengenai prosedur dan algoritma yang akan digunakan. Proses desain menerjemahkan hasil analisis yang telah dilakukan menjadi representasi atau gambaran dari perangkat lunak yang akan dibuat sebelum proses pengkodean atau implementasi dimulai.

1. Implementasi

Pada tahap implementasi pengembang perangkat lunak akan menerjemahkan desain yang sudah dibuat kedalam bahasa yang dimengerti mesin. Pada tahap ini desain yang sudah dibuat diubah menjadi sebuah produk jadi yang benar-benar dapat berfungsi.

1. Pengujian

Setelah perangkat lunak selesai dibuat, tahap selanjutnya yang dilakukan adalah tahap pengujian terhadap perangkat lunak. Pada tahap ini pengembang perangkat lunak memastikan setiap aspek dari perangkat lunak berjalan dengan baik dan tidak terjadi *error* pada perangkat lunak.



Gambar 2.12 Model Linear Sequential/ Waterfall menurut Roger Pressman

1. **Analisis**

Dalam sebuah penelitian, Analisis dari kebutuhan maupun permasalahan yang mendasari diadakannya penelitian tentunya harus dilakukan. Analisis merupakan tahapan mengurai permasalahan yang telah teridentifikasi serta menentukan kebutuhan apa saja yang diperlukan dalam proses melakukan penelitian.

1. **Analisis Permasalahan**

Permasalahan yang mendasari diadakannya penelitian ini adalah kurangnya minat peserta didik dalam bidang pemrograman komputer, hal ini disebabkan karena minimnya media pembelajaran dasar pemrograman komputer yang mudah untuk dipahami dan menarik minat peserta didik khususnya peserta didik usia dini. Kurangnya minat peserta didik dalam mendalami pemrograman komputer berimplikasi pada tidak tercukupinya kebutuhan industri akan tenaga terlatih di bidang pemrograman komputer, padahal kebutuhan akan tenaga terlatih di bidang tersebut semakin meningkat dari tahun ke tahun.

1. **Analisis Kebutuhan**

Dalam melakukan penelitian, tentunya terdapat kebutuhan-kebutuhan baik kebutuhan akan data maupun perangkat-perangkat penelitian. Demikian juga pada penelitian ini, dalam menganalisa kebutuhan yang diperlukan, kebutuhan-kebutuhan penelitian dibagi sesuai jenisnya yakni kebutuhan akan perangkat lunak dan kebutuhan akan perangkat keras.

1. **Kebutuhan P erangkat Lunak**

Dalam penelitian ini, diperlukan dukungan beberapa perangkat lunak diantaranya:

1. Godot Engine

Digunakan dalam proses pengembangan *game* edukasi. Godot Engine yang digunakan dalam penelitian ini adalah Godot Engine versi 1.1.

1. GIMP (GNU Image Manipulation Program)

Digunakan dalam proses pembuatan aset gambar raster dalam tahap pengembangan *game* edukasi.

1. Inkscape

Digunakan dalam proses pembuatan aset gambar vector dalam tahap pengembangan *game* edukasi.

1. Android Emulator

Digunakan dalam proses *testing* dan *debugging* *game* sebelum di-*deploy* di perangkat android yang sesungguhnya.

1. **Kebutuhan Perangkat Keras**

Selain perangkat lunak, dibutuhkan juga perangkat keras untuk mendukung terlaksananya penelitian ini:

1. Perangkat Android

Digunakan untuk menjalankan *game* edukasi yang telah selesai dibuat. Perangkat Android yang digunakan minimal menggunakan Sistem Operasi Android versi 4.0.

1. Komputer Desktop

Digunakan dalam proses pengembangan *game* edukasi. Dikarenakan fleksibilitas yang dimiliki Godot Engine, sistem operasi yang digunakan dalam proses pengembangan dapat menggunakan Windows, OSX, maupun Linux baik versi 32 bit maupun 64 bit. Dalam penelitian ini digunakan sistem operasi Linux Ubuntu 14.04 64 bit dengan alasan preferensi pribadi.

1. **Desain**

Dalam proses pengembangan sebuah perangkat lunak diperlukan desain yang matang sebelum memulai proses pengerjaan. Desain yang matang akan membuat tahap implementasi berjalan dengan lebih sistematis.

1. **Konsep Game**

*Ada’s Delivery Service* adalah sebuah *game* edukasi yang didesain untuk memperkenalkan dasar pemrograman komputer kepada peserta didik dengan *gameplay* dan grafis yang menarik. Dalam *game* edukasi ini, pengguna akan berperan sebagai Ada, seorang penyihir wanita cilik yang bertugas mengantarkan permen ke rumah-rumah yang terdapat di arena permainan dengan tepat waktu. Sebelum menjalankan tugasnya, Ada harus merencanakan langkahnya terlebih dahulu dengan menyediakan algoritma solusi untuk sapu ajaibnya sebelum tiap-tiap level permainan dimulai. pengguna dapat membentuk algoritma solusi dengan berbagai macam perintah. Perintah yang dapat digunakan oleh pengguna dibagi menjadi 2 macam perintah yakni perintah aksi dan perintah elemen pemrograman. Perintah aksi adalah perintah yang memerintahkan sapu ajaib untuk melakukan suatu tindakan, misalnya maju ke depan, belok ke kiri atau ke kanan, berinteraksi dengan objek, dan lain-lain, sementara perintah elemen pemrograman adalah perintah yang mengadopsi elemen-elemen pemrograman yakni subrutin dan perulangan.

Dalam setiap level permainan, tampilan antarmuka akan menampilkan area permainan, sebuah *toolbar* yang berisi setiap perintah yang direpresentasikan dengan sebuah bidak atau *piece* dan sebuah papan yang menampung algoritma solusi yang disebut *solution container*. Di dalam papan *solution container* terdapat beberapa papan kecil yang berfungsi untuk menampung fungsi-fungsi yang disebut *function container*, terdapat sebuah papan yang menampung fungsi utama atau fungsi *main*. Di dalam setiap *function container* dan *main function container* terdapat *slot* yang digunakan untuk menampung sebuah bidak perintah yang telah diinputkan oleh pengguna.

Pengguna dapat membentuk algoritma solusi dengan melakukan *drag and drop* bidak-bidak perintah ke slot-slot yang tersedia. Setelah dirasa telah menemukan algoritma solusi yang tepat, pengguna kemudian menekan tombol *Play* agar algoritma solusi yang telah disusun dijalankan oleh sapu ajaib. Sapu ajaib akan menjalankan perintah-perintah yang terdapat di dalam algoritma solusi secara sekuensial atau berurutan dimulai dari fungsi utama dan akan menjalankan perintah-perintah di fungsi lain apabila fungsi tersebut direferensikan pada fungsi utama. Apabila algoritma solusi berhasil memecahkan masalah pada level tersebut maka layar hasil akan muncul dan menampilkan skor pengguna dan level berikutnya akan terbuka, apabila algoritma solusi belum berhasil memecahkan masalah maka pengguna dapat menekan tombol *Rewind* agar sapu ajaib dapat kembali ke posisi semula dan pengguna dapat melakukan koreksi terhadap algoritma solusi yang telah disusun olehnya.

1. **Desain Materi**

Dalam membuat sebuah game yang bertujuan untuk memberikan pengetahuan dan edukasi bagi penggunanya, desain materi yang akan diajarkan penting untuk dirancang secara matang. Dengan desain materi yang matang, diharapkan pengguna dapat dengan lebih mudah menyerap informasi yang diajarkan. Materi yang akan diajarkan pada game edukasi *Ada’s Delivery Service* diadaptasi dari penelitian dan game edukasi serupa yang sudah terlebih dahulu diciptakan. Salah satunya adalah Lightbot oleh Danny Yaroslavski

Game edukasi *Ada’s Delivery Service* memperkenalkan konsep-konsep dasar mengenai pemrograman komputer. Konsep-konsep yang diperkenalkan dapat dibagi menjadi 2 bagian besar, konsep mengenai bagaimana seorang *programmer* menyelesaikan sebuah masalah yang dihadapinya (*Programming Practices*) dan konsep mengenai alur eksekusi dari sebuah program komputer secara umum (konsep *Control Flow*). *Programming practices* penting untuk diajarkan karena dalam menyelesaikan sebuah masalah dalam pemrograman komputer diperlukan kebiasaan ini, dengan terbiasa mengikuti alur *programming practices*, diharapkan pengguna dapat dengan lebih cepat menyelesaikan masalah yang dihadapinya. Sedangkan konsep mengenai *control flow* penting untuk dipahami oleh pengguna, karena untuk memprogram komputer untuk menjalankan instruksi-instruksi dan menghasilkan hasil/*output* sesuai dengan yang diinginkan, seorang *programmer* perlu mengerti terlebih dahulu bagaimana instruksi-instruksi yang dibuatnya akan dijalankan Pengertian ini akan digunakan pada tiap tahap pemrograman komputer, terutama pada tahap *design*.

1. ***Programming Practices***

Dalam game edukasi *Ada’s Deliver Service*, konsep mengenai *programming practices* didapat ketika pengguna sedang mencoba menyelesaikan suatu level dengan menyusun algoritma solusi menggunakan bidak-bidak yang disediakan.

* 1. *Requirement analysis*, pada game edukasi Ada’s Delivery Service, tahapan ini sama dengan ketika pengguna pertama kali menghadapi level, pengguna akan menganalisa dan mempelajari terlebih dahulu level yang sedang ditampilkan, pola-pola masalah yang dihadapi pada level tersebut, perintah-perintah yang tersedia, dan berapa banyak fungsi yang diperbolehkan dalam sebuah level tersebut.
  2. *Design*, pada tahapan ini, pengguna merencanakan algoritma solusi yang akan digunakan.
  3. *Coding*, pada tahapan ini, pengguna menyusun perintah yang disimbolkan dengan bidak-bidak ke dalam kontainer fungsi yang diinginkan, penyusunan dilakukan sesuai dengan desain yang telah dibuat pada tahap *design*.
  4. *Testing*, pada tahapan ini, pengguna menekan tombol *Play* dan melakukan uji coba terhadap algoritma solusi yang telah dibuat.
  5. *Debugging*, apabila level tidak terpecahkan setelah tahap *testing*, maka pengguna akan mengevaluasi algoritma solusi yang telah dibuatnya, mencari *bug* atau kesalahan di dalamnya, menekan tombol *Rewind*, kemudian menjalankannya kembali setelah algoritma solusi telah diperbaiki.

1. **Konsep *Control Flow***

Dalam game *Ada’s Delivery Service*, terdapat beberapa konsep dasar dari *control flow* sebuah program kmputer yang diajarkan, yakni konsep bahwa instruksi dijalankan secara sekuensial atau berurutan, konsep mengenai subrutin, dan konsep mengenai perulangan.

* 1. Konsep bahwa setiap instruksi dijalankan secara sekuensial, pada game edukasi *Ada’s Delivery Service*, setiap bidak-bidak perintah pada kontainer fungsi dijalankan secara sekuensial,
  2. Konsep subrutin, pada game edukasi *Ada’s Delivery Service*, pengguna diajarkan untuk menggunakan fungsi apabila slot perintah pada kontainer fungsi *Main* tidak mencukupi untuk menyelesaikan solusi, pengguna juga diajarkan mengenai bagaimana menggunakan ulang fungsi yang telah terdefinisikan dan bagaimana mengenali pola pada masalah yang dihadapi kemudian menggunakan subrutin untuk menyelesaikan pola masalah tersebut. Ketika penunjuk perintah yang sedang dijalankan menunjuk referensi pada fungsi, setiap perintah pada fungsi yang ditunjuk akan dijalankan secara berurutan kemudian setelah semua perintah dijalankan, penunjuk akan menjalankan perintah setelah referensi fungsi tersebut. Hal ini sama persis dengan cara kerja bahasa-bahasa pemrograman pada umumnya.
  3. Konsep perulangan, pada game edukasi *Ada’s Delivery Service*, baik pendefinisian maupun mekanisme perulangan dibuat sama persis dengan konsep perulangan yang sesungguhnya. Terdapat 2 macam perulangan yang dapat dilakukan pada game edukasi *Ada’s Delivery Service*. Perulangan *recursion*(*infinite looping*) dilakukan dengan cara memanggil sebuah fungsi setelah fungsi tersebut selesai dijalankan. Sedangkan *count-controlled looping*, dapat dilakukan dengan memanggil sebuah fungsi beberapa kali sesuai dengan *count* atau jumlah pemanggilan fungsi yang diinginkan oleh pengguna.

1. **Desain Sistem**

Desain sistem meliputi desain diagram alur atau *Flowchart* dan desain antarmuka *game* edukasi *Ada’s Delivery Service*.

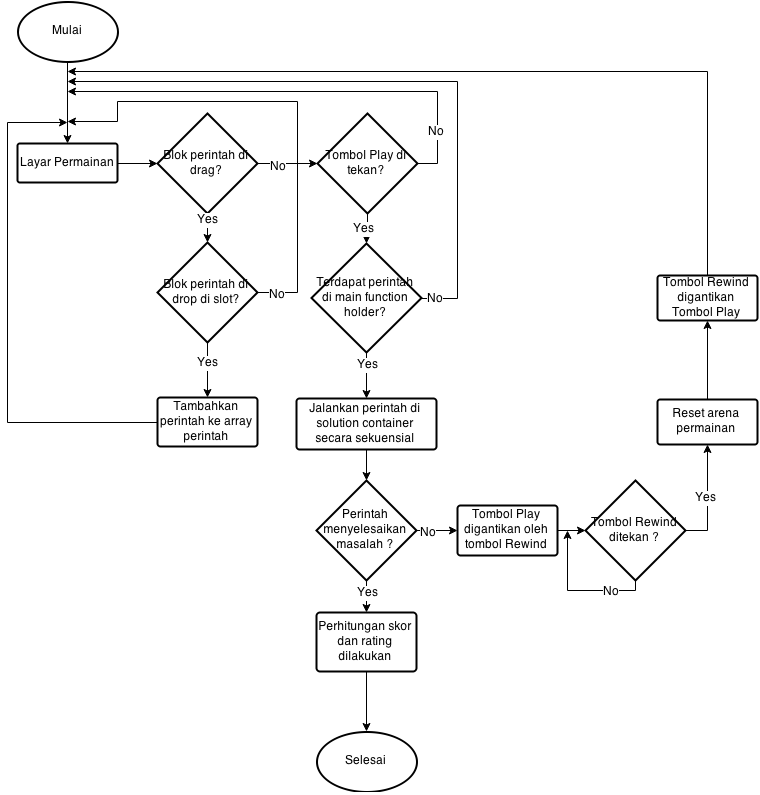
1. **Desain Diagram Alur**

Diagram alur adalah representasi dari alur keseluruhan jalannya aplikasi yang akan dibuat. Desain diagram alur sebelum mengembangkan sebuah aplikasi penting dilakukan agar sistem yang dibuat terstruktur dengan baik dan dapat memenuhi kebutuhan penggunanya. Diagram alur *game* edukasi *Ada’s Delivery Service* secara keseluruhan adalah sebagai berikut:



Gambar 3.1 Diagram Alur keseluruhan *game* edukasi *Ada’s Delivery Service*

Diagram alur keseluruhan game edukasi *Ada’s Delivery Service* (gambar 3.1) menjelaskan mengenai desain alur layar-layar yang akan ditampilkan dimulai dari awal aplikasi dijalankan hingga aplikasi dihentikan. Sedangkan diagram alur permainan atau *gameplay* dari game edukasi *Ada’s Delivery Service* (gambar 3.2) akan menjelaskan mengenai alur permainan dari game edukasi yang akan dibuat.

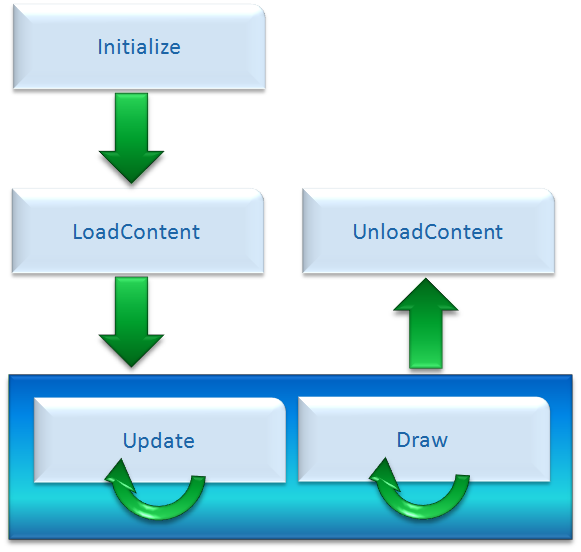


Gambar 3.2 Diagram alur *gameplay* dari *game* edukasi *Ada’s Delivery Service*

1. **Daur Hidup Game**

Dalam setiap perangkat lunak, selalu terdapat daur hidup yang dijalankan dimulai dari awal dijalankannya perangkat lunak hingga perangkat lunak dihentukan. Daur hidup dari game adalah sebagai berikut:

* 1. Tahap Initialize, tahapan ini dilakukan ketika game dijalankan pada tahap ini sistem melakukan inisialisasi *resource* yang dibutuhkan oleh sistem *shell* *game*.
  2. Tahap LoadContent, pada tahapan ini sistem melakukan pemanggilan *resource* yang dibutuhkan oleh *game* itu sendiri misalnya *resource* gambar dan *resource* musik dan *sound effect*.
  3. Tahap Update, pada tahapan ini sistem melakukan *update* pada logika dan variabel dalam game, menerima input dari pengguna, melakukan update animasi, melakukan update animasi, dan lain-lain. Tahapan update bersama dengan tahap Draw akan selalu dijalankan selama *game* masih berjalan atau berada dalam Main Loop.
  4. Tahap Draw, pada tahapan ini sistem me-*render* grafis pada layar permainan sesuai dengan kondisi terkini dari *game*. Tahap Draw bersamaan dengan tahap update akan selalu dijalankan apabila *game* masih berada di *Main Loop*.
  5. Tahap UnloadContent, tahapan ini dilakukan apabila *game* dihentikan, pada tahapan ini sistem melepaskan dan menghapus resource yang sudah di-*load* di memori.



Gambar 3.3 Daur hidup *game*

1. **Desain Antarmuka**

Desain antarmuka dilakukan untuk menemukan tata letak tiap komponen antarmuka yang sesuai agar mempermudahkan pada tahap implementasi. Antarmuka yang baik penting untuk dijadikan pertimbangan karena antarmuka adalah media berinteraksi antara komputer dan manusia. Antarmuka dalam *game* edukasi *Ada’s Delivery Service* terdiri dari beberapa layar diantaranya layar *main menu*, layar pengaturan, layar pemilihan level, dan layar permainan.

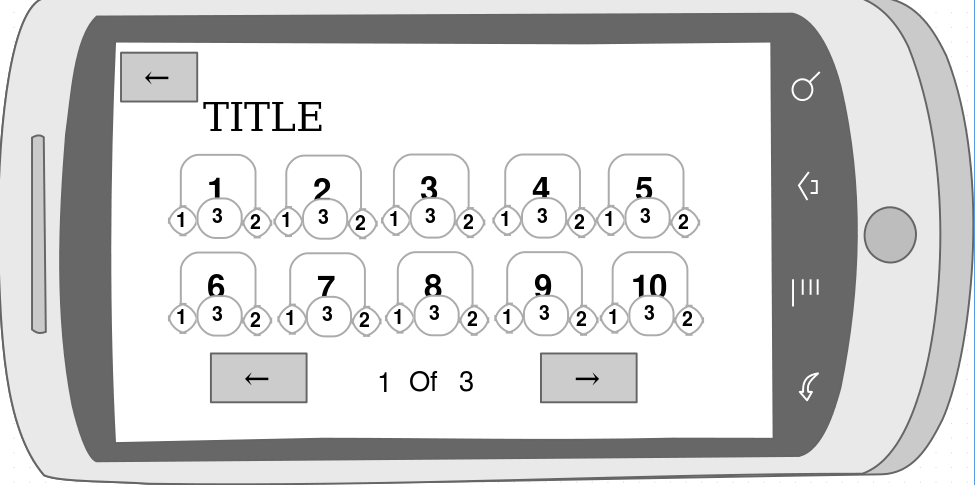
* 1. Layar Menu Utama



Gambar 3.4 *Wireframe* layar *main menu*

Layar menu utama adalah layar awal yang ditemui pengguna ketika pertama kali membuka aplikasi *Ada’s Delivery Service*. Pada layar main menu terdapat logo, tombol *play*, tombol *setting*, tombol *editor*, dan tombol *quit*. Apabila tombol *play* ditekan, fokus akan berpindah pada layar pemilihan level. Apabila tombol *setting* ditekan, fokus akan berpindah pada layar pengaturan. Apabila tombol *editor* ditekan, fokus akan berpindah pada layar editor, sedangkan tombol *quit* apabila ditekan akan menampilkan dialog konfirmasi untuk keluar dari permainan.

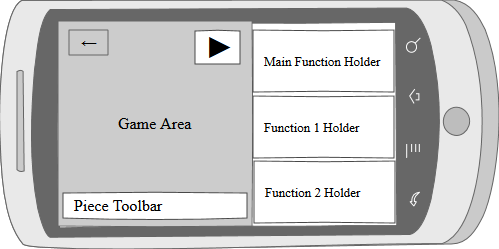
* 1. Layar Pemilihan Level



Gambar 3.5 *Wireframe* layar pemilihan level

Layar pemilihan level menampilkan level-level yang tersedia dalam *game* juga rating yang telah diperoleh oleh pengguna. Pengguna dapat memilih level yang ingin dimainkan dari level-level yang telah terbuka bagi pengguna dengan menekan tombol-tombol pemilihan level. Pada bagian bawah layar terdapat tombol navigasi untuk berpindah *level* *section* beserta indikator yang menandakan *level section* yang sedang aktif. Pada bagian pojok kiri layar terdapat tombol *back* yang apabila ditekan akan memindahkan fokus ke layar menu utama.

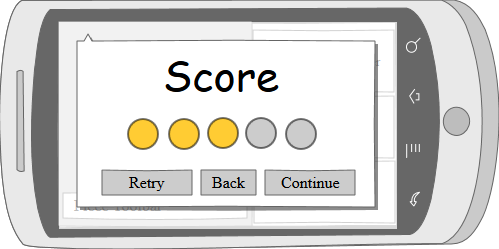
* 1. Layar Permainan



Gambar 3.6 *Wireframe* layar permainan

Layar permainan adalah layar utama *game* *Ada’s Delivery Service*. Pada layar ini terdapat *piece toolbar*, *solution container,* area permainan dan tombol Play.

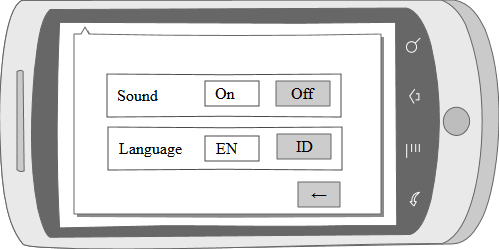
* 1. Layar Permainan (Hasil)



Gambar 3.7 *Wireframe* layar hasil permainan

Setelah pengguna berhasil memecahkan masalah pada suatu level maka pengguna akan dihadapkan pada layar hasil permainan. Pada layar hasil permainan ditampilkan skor dan *rating* yang didapatkan oleh pengguna, tombol retry dan tombol continue. Tombol Retry digunakan untuk mengulangi level yang telah berhasil diselesaikan pengguna apabila pengguna masih kurang puas dengan hasil yang didapatnya sedangkan tombol Continue digunakan apabila pengguna telah puas dengan hasil yang didapat dan ingin melanjutkan ke level berikutnya.

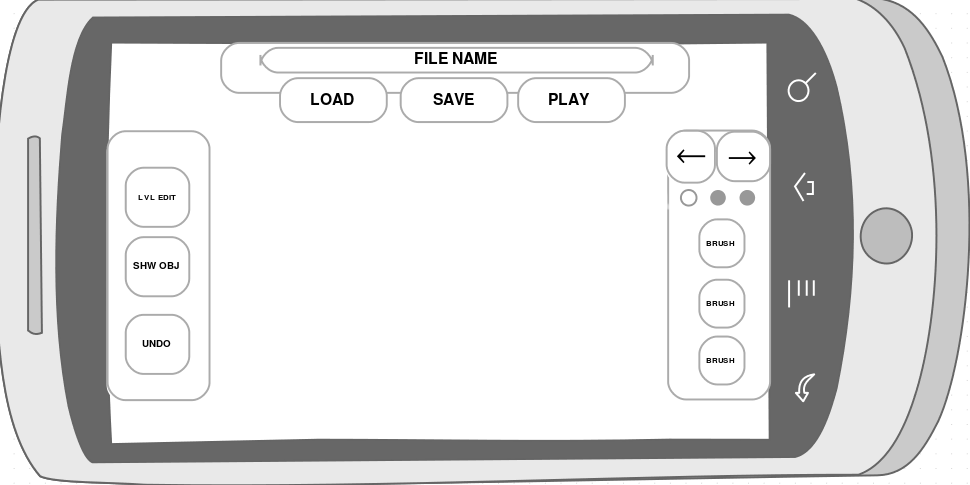
* 1. Layar Pengaturan



Gambar 3.8 *Wireframe* layar hasil permainan

Layar pengaturan ditampilkan apabila pengguna menekan tombol *setting* pada layar main menu, layar pengaturan digunakan untuk mengatur aspek-aspek dalam *game* seperti *sound* dan bahasa.

* 1. Layar Editor



Gambar 3.9 *Wireframe* layar editor

Layar editor digunakan untuk membuat level sendiri, antarmuka pada layar ini, terbagi menjadi 3 bagian utama. *Filebox* terletak pada bagian atas layar, guna dari *filebox* adalah untuk berhubungan dengan *file* dari level yang sedang aktif. Pada bagian kiri layar, terdapat *toolbar*, *toolbar* digunakan untuk menyunting hal-hal yang berhubungan dengan level. Sedangkan pada bagian kanan layar, terdapat *toolbox*, yang berfungsi untuk menampung kuas-kuas yang dapat digunakan untuk mengedit petak-petak di dalam peta dari elvel yang sedang aktif.

1. **Skema Uji Coba**

Dalam proses uji coba game edukasi yang akan dibuat, akan digunakan metode *white box testing* maupun *black box testing*. Metode *white box testing* akan dilakukan ketika proses pengembangan sedang berlangsung. *White box testing* dilakukan dengan cara men-*trace* hasil/*output* dari aplikasi setelah *input* dimasukkan, selain itu dilakukan juga pemeriksaan terhadap kode sumber. Dalam metode ini diharapkan *bug* yang dihasilkan baik karena adanya *syntax error* maupun *logic error* sudah dapat diatasi. Sedangkan metode *black box testing* dilakukan dengan bantuan pihak-pihak eksternal, misalnya pengguna aplikasi. Dengan menggunakan metode *white box testing* maupun *black box testing*, maka skema uji coba yang akan dilakukan dibagi menjadi tiga bagian utama:

* 1. Uji coba fitur-fitur dalam game edukasi

Uji coba ini akan berlangsung mulai dari proses pengembangan hingga proses uji coba yang dilakukan oleh pengguna berjalan. Pada uji coba ini fitur-fitur dalam game edukasi akan dijalankan dan akan dicari *error/bug* yang terdapat dalam fitur-fitur yang diuji coba.

* 1. Uji coba kemenarikan game edukasi menurut pengguna

Untuk mengukur tingkat keberhasilan aplikasi dari segi tingkat kemenarikan menurut pengguna, akan digunakan angket yang akan diberikan setelah pengguna mencoba game edukasi. Diharapkan tidak kurang dari 10 orang pengguna dapat berpartisipasi pada uji coba ini. Hal-hal yang akan diperhatikan dalam pengujian ini adalah kemudahan dalam mengakses tiap fitur game edukasi serta seberapa menarik desain antarmuka, karakter, suara dan musik, maupun *gameplay* yang ditawarkan dalam game edukasi.

* 1. Uji coba manfaat yang didapat oleh pengguna

Pengujian manfaat yang didapat oleh pengguna setelah memainkan game edukasi dititikberatkan pada peningkatan pemahaman pengguna terhadap mekanisme yang diberikan oleh game edukasi yang secara tidak langsung melambangkan konsep-konsep dasar pemrograman komputer. Untuk menguji hal tersebut, akan dilakukan pengujian terhadap peningkatan kecepatan pengguna dalam menyelesaikan masalah yang disediakan oleh game edukasi. Pada pengujian ini, mula-mula akan dipiih 10 orang responden berumur 8-12 tahun. Para responden kemudian akan diberikan versi awal dari game edukasi, pada versi awal tersebut terdapat 3 level yang berfungsi sebagai pengenalan mekanisme dalam game dan konsep-konsep dasar pemrograman komputer yang diajarkan dalam game, yakni pengenalan mengenai perintah-perintah yang tersedia, mekanisme subrutin, dan mekanisme perulangan. Lalu akan diberikan level-level tambahan yang baik dari segi tingkat kesulitan maupun mekanisme tidak jauh berbeda. Pada tiap level yang diselesaikan akan dilakukan pengukuran seberapa cepat tiap responden menyelesaikan level yang diberikan. Hasil kecepatan responden dalam menyelesaikan tiga level pengenalan akan diberlakukan sebagai hasil *pre-test*, sedangkan kecepatan untuk menyelesaikan level-level berikutnya akan diberlakukan sebagai hasil *post-test*

Game edukasi akan dinilai berhasil apabila, pertama, tidak terdapat *major* *error/bug* pada fitur-fitur yang ditawarkan, *major* *error* yang dimaksud ialah *error* yang sifatnya sangat mengganggu pengguna atau bahkan menyebabkan fitur yang ditawarkan tidak dapat digunakan. Kedua, tingkat kemenarikan game edukasi baik dari segi kenyamanan pengguna, desain antarmuka, desain karakter, suara maupun musik, dan dari segi *gameplay* mendapatkan nilai diatas 50% dari minimal 70% responden. Dan ketiga, terdapat peningkatan kecepatan pada 70% responden dalam menyelesaikan tiap-tiap level yang diberikan setelah tiga level pengenalan yang telah diberikan terlebih dahulu.

**BAB IV**

**HASIL DAN PEMBAHASAN**

**4.1 Game Edukasi Ada’s Delivery Service**

Pada game edukasi Ada’s Delivery Service, antarmuka yang digunakan terbagi menjadi beberapa layar-layar dan dialog-dialog pendukung. Layar-layar yang terdapat pada game edukasi ini antara lain: Layar menu utama, layar menu level, layar permainan, dan layar editor peta.

**4.1.1. Layar Menu Utama**

Layar menu utama adalah layar awal yang akan dijumpai ketika aplikasi dibuka. Pada layar ini terdapat 3 tombol utama, yaitu tombol play, tombol setting, dan tombol editor. Tombol play digunakan untuk menuju ke layar menu level, tombol setting digunakan untuk menampilkan dialog pengaturan, sedangkan tombol editor digunakan untuk memulai editor peta. Tampilan layar menu utama dapat dilihat pada gambar 4.main berikut.



Gambar 4.main Tampilan layar menu utama

Apabila tombol setting ditekan maka akan memunculkan dialog setting seperti pada gambar 4.setting, pada dialog setting pengguna dapat menyalakan atau mematikan suara dan mengubah bahasa aplikasi. Untuk mengaplikasikan perubahan bahasa yang digunakan, diperlukan *restart* aplikasi. Oleh karenanya setelah setting bahasa diubah, aplikasi akan menampilkan dialog konfirmasi *restart* aplikasi agar perubahan dapat diaplikasikan segera (gambar 4.restart).



Gambar 4.setting Tampilan dialog setting



Gambar 4.restart Tampilan dialog konfirmasi restart aplikasi

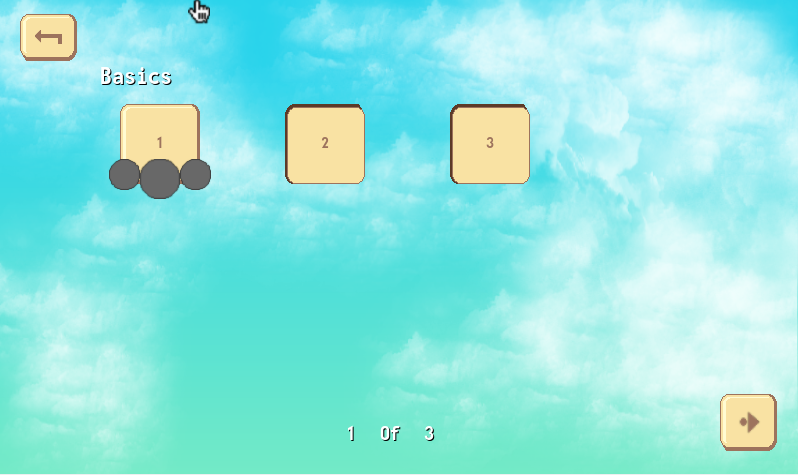
Apabila tombol back pada pojok kiri atas ditekan maka aplikasi akan menampilkan dialog konfirmasi keluar dari aplikasi seperti pada gambar 4.quit



Gambar 4.quit Tampilan dialog konfirmasi keluar dari aplikasi

**4.1.2 Layar Menu Level**

Layar menu level berfungsi untuk memilih level yang akan dimainkan. Pada layar ini terdapat tombol-tombol pemilihan level dibagian atas dan tombol next section dan tombol previous section di bagian bawah layar. Tombol-tombol pemilihan level selain menampilkan indeks level juga menampilkan pencapaian pengguna dalam level yang dimaksud, dalam permainan ini tingkat keberhasilan pengguna disimbolkan dengan permen dalam skala 1-3. Tombol next section digunakan untuk memindah fokus menu ke *level section* berikutnya, sedangkan tombol previous section berlaku sebaliknya. Apabila tombol back pada pojok kiri atas layar ditekan maka antarmuka akan kembali pada layar menu utama. Setelah pengguna memilih level yang akan dimainkan, pengguna akan dihadapkan pada layar permainan. Tampilan layar menu level dapat dilihat pada gambar 4.level.



Gambar 4.level Tampilan layar menu level

Tombol-tombol pemilihan level merepresentasikan sejauh mana progress dari pengguna dalam game edukasi ini, oleh karenanya tombol-tombol pemilihan level harus di-*update* saat aplikasi dimulai dan ketika pengguna menyelesaikan sebuah level. Untuk melakukan proses update tombol pemilihan level, digunakan fungsi updateLevelSelector seperti pada gambar 4.updateLevelSelector

01 func updateLevelSelector():

02 var tempContainer

03 var tempLevel

04 var maxSectionIndex = 0

05 var mapReferences =

06 references["rootNode"].getUserdata().mapReferences

07 for container in mapReferences:

08 if currents["levelSelectorIndex"][container]["container"] == null:

09 tempContainer = prefabs["levelContainer"].instance()

10 references["levelBtnContainer"].add\_child(tempContainer)

11 tempContainer.hide()

12 tempContainer.init(container, mapReferences[container].title)

13 currents["levelSelectorIndex"][container]["container"] =

14 tempContainer

15 currents["levelSelectorIndex"][container]

16 ["container"].init(container,mapReferences[container].title)

17 for level in mapReferences[container]["levels"]:

18 var temp = mapReferences[container]["levels"][level]

19 if currents["levelSelectorIndex"][container][level] == null:

20 tempLevel = prefabs["levelBtn"].instance()

21 tempContainer.references["container"].add\_child(tempLevel)

22 tempLevel.init({"containerIndex":container,"levelIndex":level},

23 temp["unlocked"],temp["bestCandy"])

24 currents["levelSelectorIndex"][container][level] = tempLevel

25

26 currents["levelSelectorIndex"][container][level].init(

27 {"containerIndex":container,"levelIndex":level},temp["unlocked"],

28 temp["bestCandy"])

29

30 maxSectionIndex+=1

31

32 currents["maxSectionIndex"] = maxSectionIndex -1

33

34 self.updateNavigationBtn()

Gambar 4.updateLevelSelector Kode sumber fungsi updateLevelSelector pada menu level

Fungsi updateLevelSelector akan terlebih dahulu melakukan pengambilan data pengguna dari node akar untuk mengetahui progress dari pengguna yakni level yang telah terselesaikan, level yang sudah dapat dimainkan, rating tertinggi yang didapat oleh pengguna, dan skor terbaik yang pernah didapat oleh pengguna. Fungsi tersebut kemudian melakukan pengecekan pada kondisi tombol pemilihan level yang sedang berjalan menggunakan indeks level dari data user yang sudah diambil, apabila tombol dengan indeks yang dimaksud tidak ditemukan maka fungsi akan membuat tombol baru menggunakan prefab yang sudah didefinisikan sebelumnya, tombol yang baru saja dibuat kemudian ditambahkan pada menu. Setelah memastikan tombol yang dimaksud sudah ditambahkan ke dalam menu, fungsi akan mengisi atribut tombol sesuai dengan data pengguna pada indeks yang dimaksud.

**4.1.3 Layar permainan**

Layar permainan adalah layar utama yang digunakan untuk memainkan game edukasi ini. Dalam game edukasi ini pengguna ditantang untuk menyusun algoritma solusi untuk menyelesaikan pengiriman permen ke rumah-rumah di peta yang ditampilkan dengan melakukan *drag and drop* bidak-bidak perintah ke slot-slot yang tersedia. Setelah dirasa telah menemukan algoritma solusi yang tepat, pengguna kemudian menekan tombol *Play* agar algoritma solusi yang telah disusun dijalankan oleh sapu ajaib. Sapu ajaib akan menjalankan perintah-perintah yang terdapat di dalam algoritma solusi secara sekuensial atau berurutan dimulai dari fungsi utama dan akan menjalankan perintah-perintah di fungsi lain apabila fungsi tersebut direferensikan pada fungsi utama. Apabila algoritma solusi berhasil memecahkan masalah pada level tersebut maka layar hasil akan muncul dan menampilkan skor pengguna dan level berikutnya akan terbuka, apabila algoritma solusi belum berhasil memecahkan masalah maka pengguna dapat menekan tombol *Rewind* agar sapu ajaib dapat kembali ke posisi semula dan pengguna dapat melakukan koreksi terhadap algoritma solusi yang telah disusun olehnya. Tampilan layar permainan dapat dilihat pada gambar 4.game.



Gambar 4.game Tampilan layar permainan

Dalam mengelola logika permainan, layar permainan berperan sebagai view yang bertugas untuk menampilkan dan menerima masukan dari pengguna dan meneruskannya ke kontroler game. Kontroler game menerima masukan berupa algoritma solusi yang telah diekstrak menjadi sebuah *dictionary.* Untuk melakukan ekstraksi algoritma solusi dari layar permainan digunakan fungsi extractSolverAlgorithm seperti pada gambar 4.extractSolverAlgorithm.

01 func extractSolverAlgorithm():

02 var tempAlgorithm = {}

03 var tempPieceArray ={}

04 var index = 0

05 for pieceContainerRoot in

06 references["pieceContainers"].get\_children():

07 for piece in

08 pieceContainerRoot.getPieceContainer().get\_children():

09 if piece.getAction() != "":

10 tempPieceArray[str(index)] = piece.getAction()

11 index+=1

12 tempAlgorithm[str(pieceContainerRoot.getLink())] = tempPieceArray

13 tempPieceArray = {}

14 index = 0

15 currents["solverAlgorithm"] = tempAlgorithm

Gambar 4.extractSolverAlgorithm potongan kode sumber untuk fungsi extractSolverAlgorithm pada kelas layar permainan

Fungsi extractSolverAlgorithm melakukan iterasi pada kontainer-kontainer fungsi kemudian mengambil perintah pada tiap-tiap *piece* yang terdapat didalamnya. Perintah yang diambil kemudian ditambahkan ke dalam array *tempPieceArray* secara berurutan. Array perintah tersebut kemudian ditambahkan pada *dictionary* *tempAlgorithm* dan direferensikan menggunakan *link* dari kontainer fungsi yang bersangkutan. Berikut contoh *dictionary* sederhana hasil ekstraksi algoritma solusi:

01 {

02 “M”:{“move”,”turnLeft”,”move”,”turnRight”, “interact”, “F1”},

03 “F1”:{“move”,”turnRight”}

04 }

Gambar 4.solverAlgorithm Contoh d*ictionary* sederhana hasil ekstraksi algoritma solusi dari kontainer fungsi

Setelah *dictionary* algoritma solusi didapat, dictionary tersebut kemudian diteruskan ke kontroler game menggunakan fungsi playSolverAlgorithm seperti pada gambar 4.playSolverAlgorithm, fungsi tersebut terlebih dahulu akan melakukan pengecekan apakah kontroler sedang menjalankan algoritma solusi atau sedang berstatus *idle.* Apabila kontroler sedang berstatus *idle* maka fungsi playSolverAlgorithm akan memanggil fungsi extractSolverAlgorithm untuk melakukan ekstraksi algoritma solusi, mengubah *state* tombol play ke rewind, kemudian meneruskan algoritma solusi ke kontroler game.

01 func playSolverAlgorithm():

02 if not

references["gameController"].currents["HANDLE\_SOLVER\_ALGORITHM"]:

03 self.extractSolverAlgorithm()

04 references["gameController"].playSolverAlgorithm(

currents["solverAlgorithm"])

05 self.setPlayBtn("rewind")

06 if references["tutorialUI"].getCurrentStep() ==

"play\_solver\_algorithm":

07 references["tutorialUI"].goToNextStep()

Gambar 4.playSolverAlgorithm Kode sumber fungsi playSolverAlgorithm

Kontroler game kemudian menerima algoritma solusi menggunakan fungsi receiveSolverAlgorithm() seperti pada gambar 4.receiveSolverAlgorithm. Fungsi tersebut kemudian me-reset peta yang sedang ditampilkan lalu mengubah nilai variabel currents["HANDLE\_SOLVER\_ALGORITHM"] menjadi *true*, variabel tersebut berperan sebagai *flag* untuk menentukan dijalankan atau tidaknya fungsi handleSolverAlgorithm dalam *main loop* aplikasi, apabila fungsi handleSolverAlgorithm dijalankan, fungsi akan mulai menangani algoritma solusi yang telah diterima.

01 func receiveSolverAlgorithm(solverAlgorithm):

02 references["mapRoot"].resetMap()

03 currents["solverAlgorithm"]= solverAlgorithm

04 currents["HANDLE\_SOLVER\_ALGORITHM"] = true

05 currents["actionIndex"] = 0

06 currents["repeatIndex"] = 0

Gambar 4.receiveSolverAlgorithm kode sumber untuk fungsi receiveSolverAlgorithm pada kontroler game.

Fungsi handleSolverAlgorithm akan mengecek status objek *player* terlebih dahulu, apabila objek *player* telah selesai melakukan aksi dan berstatus *idle*,fungsi akan mengambil perintah berikutnya dengan menggunakan fungsi fetchCurrentAction*,* apabila terdapat perintah untuk dijalankan maka perintah akan diteruskan pada objek player untuk selanjutnya dijalankan, apabila perintah yang dimaksud adalah perintah “interact” atau perintah untuk berinteraksi dengan objek di peta, kontroler akan melakukan pengecekan status “IS\_DELIVERED” tiap objek rumah pada peta yang sedang ditampilkan, apabila tidak ditemukan rumah dengan status “IS\_DELIVERED” yang bernilai false, maka game dapat dipastikan telah selesai, dan fungsi gameOver akan dipanggil. Fungsi handleSolverAlgorithm dapat dilihat pada gambar 4.handleSolverAlgorithm

01 func handleSolverAlgorithm(delta):

02 if references["player"].getCurrentActionState() == "idle":

03 var act = self.fetchCurrentAction("M")

04 if act!= null:

05 currents["waitTime"] = 0

06 currents["repeatIndex"] = 0

07 references["player"].doAction(act)

08 if act == "interact":

09 currents["LEVEL\_COMPLETE"] = self.isGameOver()

10 if currents["LEVEL\_COMPLETE"]:

11 self.gameOver()

12 references["mapRoot"].updateWorld()

13 else:

14 currents["HANDLE\_SOLVER\_ALGORITHM"] = false

15 references["gameUI"].setFocusedPiece("null",-1)

Gambar 4.handleSolverAlgorithm Kode sumber dari fungsi handleSolverAlgorithm pada kontroler game

Fungsi fetchCurrentAction digunakan untuk mengambil perintah yang akan dijalankan oleh objek player ketika algoritma solusi ditangani oleh kontroler game. Fungsi ini menerima masukan berupa link dari fungsi yang akan dituju dan mengeluarkan perintah yang harus dijalankan oleh objek player. Fungsi ini melakukan iterasi terhadap salah satu array fungsi pada *dictionary* algoritma solusi yang telah disediakan, link yang menjadi masukan digunakan sebagai *key* untuk menunjuk array fungsi mana yang akan dilakukan iterasi. Dilakukan pengecekan terhadap perintah, apabila perintah berisi nilai “move”, “turnLeft”, “turnRight”, atau “interact”. Kemudian dilakukan pengecekan apakah indeks perulangan telah sesuai dengan indeks perintah yang dimaksud, apabila sesuai maka nilai perintah akan dikembalikan. Apabila perintah tidak dikenali, maka fungsi akan memanggil dirinya sendiri dengan perintah yang tidak dikenali tersebut sebagai link karena perintah yang tidak dikenali tersebut diasumsikan adalah link untuk fungsi lain.

01 func fetchCurrentAction(link):

02 for i in range(0, currents["solverAlgorithm"][link].size()):

03 var action =currents["solverAlgorithm"][link][str(i)]

05 if self.isAction(action):

06 if currents["actionIndex"] == currents["repeatIndex"]:

07 references["gameUI"].setFocusedPiece(link,i)

08 currents["actionIndex"]+=1

09 return action

10 currents["repeatIndex"]+=1

11 else:

12 var act = self.fetchCurrentAction(action)

13 if isAction(act):

14 return act

Gambar 4.fetchCurrentAction kode sumber fungsi fetchCurrentAction pada kontroler game

Apabila tombol back pada pojok kiri atas ditekan maka layar akan menampilkan dialog konfirmasi keluar dari layar permainan seperti pada gambar 4.exitGame. Jika pengguna memilih untuk keluar dari layar permainan, maka aplikasi akan menampilkan layar menu level.



Gambar 4.exitGame Tampilan dialog keluar dari layar permainan

Pada beberapa level tertentu terdapat adegan percakapan para karakter dalam game. Percakapan para karakter menjadi elemen cerita dari game edukasi sekaligus media untuk menyampaikan mekanisme permainan secara lisan. Percakapan akan muncul apabila kondisi yang dibutuhkan oleh percakapan tersebut telah terpenuhi, misalnya saja pada saat permainan dimulai atau level berhasil diselesaikan. Tampilan percakapan dapat dilihat pada gambar 4.conversation



Gambar 4.conversation Tampilan layar percakapan



Gambar 4.focusedItem Tampilan layar percakapan ketika menerangkan mengenai sebuah elemen dalam game

Pada layar percakapan, pengguna dapat melanjutkan percakapan dengan menekan layar dimana saja, atau memilih respon yang diinginkan apabila muncul pilihan respon yang akan diberikan dalam percakapan (Gambar conversation.respond). Pada layar percakapan juga terdapat tombol skip pada pojok kanan atas layar yang digunakan untuk melewati percakapan yang sedang ditampilkan.



Gambar 4.conversation.respond Tampilan pemilihan respon pengguna

Untuk menampilkan percakapan, sebuah level wajib memiliki data percakapan. Data percakapan kemudian akan diseleksi sesuai dengan kondisi yang didefinisikan pada atribut *condition*. Data percakapan kemudian diolah pada antarmuka percakapan lalu ditampilkan ke pengguna. Berikut struktur dasar data percakapan yang akan diolah:

01 {

02 {

03 "condition":{"start":1},

04 "data":{

05 "skipAction":[],

06 0:{"text":"CONVERSATION\_BASIC1\_START0",

07 "name":"Lino",

08 "actor":"lino",

09 "emotion":"normal",

10 "currentAction":[],

11 "nextAction":{1:[["goTo",[1]]]}

12 } },

13 }

Gambar 4.conversationData Struktur dasar data percakapan

Fungsi handleConversationEvent pada kontroler game digunakan untuk menampilkan percakapan pada sebuah level. Fungsi tersebut akan melakukan pengecekan ada tidaknya data percakapan pada data level terlebih dahulu. Apabila ditemukan data percakapan pada data level yang sedang dimainkan, fungsi akan mencocokkan atribut *condition* pada tiap data percakapan dengan kondisi yang menjadi masukan, apabila kondisi sesuai maka fungsi akan menampilkan antarmuka percakapan dan meneruskan data percakapan yang dimaksud sebagai masukan. Apabila tidak ditemukan data percakapan pada data level yang sedang dimasukkan maka fungsi akan mengembalikan nilai null. Kode sumber fungsi handleConversationEvent dapat dilihat pada gambar 4.handleConversationEvent berikut.

01 func handleConversationEvent(condition):

02 if currents["levelData"].has("conversation"):

03 var currentConversation = null

04 for conversation in currents["levelData"].conversation:

05 if currentConversation == null:

06 var isTargetedConversation = true

07 for param in condition:

08 if isTargetedConversation:

09 if conversation.condition.has(param):

10 if not condition[param] ==conversation.condition[param]:

11 isTargetedConversation = false

12 else:

13 isTargetedConversation = false

14 if isTargetedConversation:

15 currentConversation = conversation

16 if currentConversation != null:

17 references["guiRoot"].showConversation(true, self,

18 currentConversation.data)

19 return currentConversation

Gambar 4.handleConversationEvent Kode sumber untuk fungsi handleConversationEvent

Selain percakapan, pada level tertentu juga terdapat tutorial berupa instruksi-instruksi singkat mengenai apa yang harus dilakukan oleh pengguna seperti pada gambar 4.tutorial



Gambar 4.tutorial Tampilan tutorial pada layar permainan

Untuk menampilkan tutorial pada level diperlukan data tutorial yang tersimpan dalam data level tersebut, data tutorial berupa array dari string yang menunjukkan animasi apa yang harus dimainkan oleh antarmuka tutorial dan parameter yang menjadi masukan, parameter didefinisikan setelah karakter titik (.), Contoh data tutorial sederhana dapat dilihat pada gambar 4.dataTutorial berikut.

01 ["grab\_piece.move","drop\_piece.move.M.1","grab\_piece.interact",

02 "drop\_piece.interact.M.2","play\_solver\_algorithm"]

Gambar 4.dataTutorial Contoh data tutorial sederhana

Apabila level telah berhasil diselesaikan, maka aplikasi akan menampilkan dialog hasil permainan, pada dialog ini ditampilkan berapa banyak langkah yang dibutuhkan oleh pengguna untuk menyelesaikan level tersebut dan berapa rating yang didapat oleh pengguna (direpresentasikan sebagai permen). Selain itu pada dialog ini pengguna diberikan pilihan untuk mengulang level yang sedang ditampilkan menggunakan tombol retry, kembali ke menu level menggunakan tombol back atau melanjutkan ke level berikutnya menggunakan tombol continue.



Gambar 4.resultMenu Tampilan dialog hasil permainan

**4.1.4 Layar Editor Peta**

Layar editor peta dapat diakses melalui layar menu utama menggunakan tombol editor. Layar ini digunakan oleh pengguna untuk membuat, menyimpan, maupun memuat peta selain peta yang sudah disediakan secara default oleh aplikasi. Antarmuka di layar ini dibagi menjadi 3 bagian utama, pada bagian atas terdapat Filebox yang digunakan untuk melakukan segala sesuatu yang berhubungan dengan file peta yang aktif. Pada bagian ini terdapat beberapa tombol, tombol load digunakan untuk memuat peta dari file, kemudian tombol save digunakan untuk menyimpan peta ke dalam file, dan tombol play digunakan untuk memainkan peta dalam mode pratinjau. Pada bagian kiri terdapat toolbar yang berisi tombol editor atribut peta yang digunakan untuk menyunting atribut level seperti posisi player, jumlah fungsi yang diperbolehkan, dan lain-lain, tombol tampilkan atau sembunyikan objek pada peta digunakan untuk menyembunyikan sementara objek atau menampilkannya kembali, fitur ini dibuat agar ketika pengguna membuat peta, pandangan tidak terhalang oleh objek-objek pada peta, dan tombol undo yang digunakan untuk mengulangi perintah yang baru saja dilakukan. Pada bagian kanan layar terdapat toolbox yang menampung berbagai macam kuas yang digunakan dalam proses penyuntingan peta, seperti kuas untuk masing-masing petak, kuas untuk menghapus petak dari peta, dan kuas untuk menyunting atribut dari sebuah objek pada peta. Tampilan layar editor peta dapat dilihat pada gambar 4.editor.



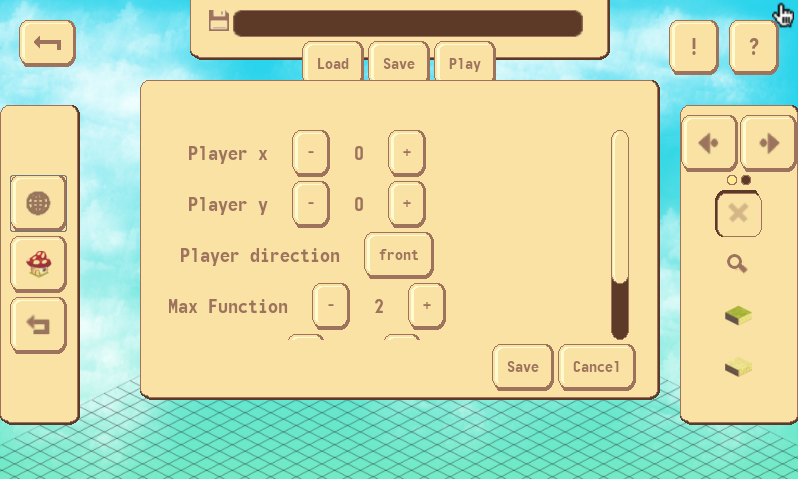
Gambar 4.editor Tampilan layar editor peta

Tombol help pada pojok kanan atas berfungsi untuk menampilkan tutorial singkat mengenai elemen-elemen antarmuka pada layar editor peta. Tampilan tutorial singkat tersebut dapat dilihat pada gambar 4.editorTutorial.



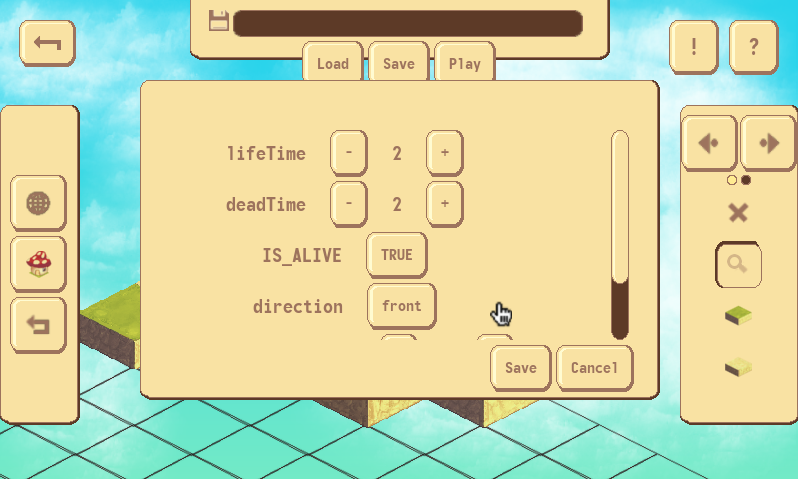
Gambar 4. editorTutorial Tampilan tutorial pada layar editor

Untuk membantu pengguna dalam melakukan proses penyuntingan atribut level, maka dibuat dialog khusus yang akan muncul apabila tombol edit atribut peta ditekan. Pada dialog ini, terdapat segala atribut level seperti posisi awal pengguna, jumlah fungsi maksimal yang dapat digunakan pada level tersebut, berapa jumlah langkah maksimal untuk mendapatkan rating tertentu. Tampilan dialog ini dapat dilihat pada gambar 4.levelEdit.



Gambar 4.levelEdit Tampilan editor atribut level

Dialog untuk mengedit atribut objek juga sengaja dibuat untuk memudahkan pengguna mengedit atribut sebuah objek di dalam peta. Dialog ini akan muncul apabila pengguna menggunakan kuas dengan mode edit atribut objek dan pengguna melakukan klik pada sebuah objek di dalam peta yang sedang aktif. Dialog edit atribut objek dapat dilihat pada gambar 4.



Gambar 4.objectEdit Tampilan editor untuk atribut objek Teleporter

Untuk mengatur operasi penyuntingan peta, operasi penyuntingan data level, maupun operasi terhadap file peta, digunakan kontroler editor terpisah. Kontroler editor akan menerima perintah yang dimasukkan oleh pengguna melalui antarmuka editor lalu meneruskannya ke template peta yang sedang disunting kemudian template peta akan melakukan operasi sesuai perintah dari kontroler editor tersebut.

Untuk menyunting peta menggunakan kuas yang tersedia di *toolbox*, kontroler editor meneruskan nilai posisi yang akan diedit, nilai petak yang aktif, dan atribut objek ke fungsi paint pada template peta. Fungsi paint pada template map dapat dilihat pada gambar 4.paint.

01 func paint(mapPos, objectType, objectAttribute):

02 var cell = {"x": mapPos.x, "y": mapPos.y, "z":0, "groundType":

objectType,"objectAttribute":objectAttribute}

03 if cell.x >= 0 and cell.x < MAP\_MAX\_X and cell.y >= 0 and cell.y <

MAP\_MAX\_Y:

04 if references["ground"].get\_cell(cell.x,cell.y) != objectType:

05 var previousCell = self.getMapDataCell(mapPos.x,mapPos.y,0)

06 if previousCell == null:

07 previousCell = {"x": mapPos.x, "y": mapPos.y, "z": 0,

"groundType": -1, “objectAttribute":{}}

08 references["ground"].set\_cell(cell.x,cell.y,objectType)

09 self.removeObject(cell)

10 self.spawnObject(cell)

11 self.setMapDataCell(cell)

12 return previousCell

13 return null

Gambar 4.paint Kode sumber fungsi paint pada template peta

Untuk memuat data peta dari file, kontroler membaca variabel dari file, variabel kemudian digunakan sebagai acuan untuk membangun peta dan level pada template peta menggunakan fungsi loadMapData pada template peta.

01 func loadMapData():

02 self.clearMap()

03 for cell in currents["map"].mapData:

04 references["ground"].set\_cell(cell.x,cell.y,cell.groundType)

05 if cell.x+1 > currents["mapSize"].x:

06 currents["mapSize"].x = cell.x+1

07 if cell.y+1 > currents["mapSize"].y:

08 currents["mapSize"].y = cell.y+1

09 self.spawnObject(cell)

10 if currents["mapSize"] == Vector2(0,0) or self.isEditorMode():

11 currents["mapSize"] = Vector2(MAP\_MAX\_X,MAP\_MAX\_Y)

11 references["playerCam"].setMaxPos(self.getMapSize(),

12 references["ground"].get\_cell\_size())

13 references["editorCam"].setMaxPos(self.getMapSize(),

14 references["ground"].get\_cell\_size())

15

16 references["player"].setPosition(currents["map"].playerData

, false)

Gambar 4.loadMapData Kode sumber fungsi loadMapData pada template peta

**4.2 Struktur Data Level**

Data level adalah data yang digunakan sebagai acuan ketika aplikasi membangun sebuah level untuk kemudian dapat diedit maupun dimainkan oleh pengguna. Mekanisme penyimpanan data level menjadi penting untuk diperhatikan karena pada game edukasi ini pengguna selain dapat memainkan level yang disediakan secara default oleh aplikasi, pengguna juga dapat membuat sendiri level sesuai dengan keinginan, menyimpan, kemudian membagikannya ke pengguna lain.

Dalam menentukan jenis struktur data yang akan digunakan untuk menyimpan data level, dua faktor utama yang menjadi pertimbangan adalah kemudahan bagi pengguna dalam membaca data ketika data tersebut ditampilkan pada fitur editor dan kemudahan pengaksesan data dari sisi kode. Dengan mempertimbangkan faktor diatas, dipilihlah struktur data *dictionary*. Struktur data *dictionary* dipilih karena mendukung asosiasi *key* dan *value*, dengan demikian struktur data ini lebih mudah dibaca oleh manusia(*human readable*) dan memudahkan pengaksesan dari sisi kode karena *value* yang diinginkan dapat dipanggil menggunakan *key tertentu*.

Data level terbagi menjadi 2 bagian, yakni levelData dan mapData, pada bagian levelData disimpan data-data global dari sebuah level, yakni posisi awal pengguna (playerInitialPosition) , jumlah langkah maksimal yang dibutuhkan untuk mendapatkan rating (candy) tertentu pada level tersebut (candy), berapa banyak fungsi yang dapat digunakan (maxFunction), langkah-langkah tutorial yang akan ditampilkan (tutorial), dan percakapan apa saja yang mungkin terjadi pada level tersebut (conversation). Sedangkan mapData terdiri dari kumpulan array yang berfungsi untuk menyimpan peta dari level tersebut beserta atribut dari tiap-tiap objek yang terdapat didalamnya.

01 {

02 “levelData”:{

03 “playerInitialPosition”:{

04 “x”:0,

05 “y”:0,

06 “z”:0,

07 “direction”:”front”

08 },

09 "candy":{

10 "1": 0,

11 "2": 0,

12 "3": 0

13 },

14 "maxFunction":1,

15 tutorial:["create\_new\_function","grab\_piece.move",

16 "drop\_piece.move.M.1"],

17 conversation : [{

18 "condition":{"start":1},

19 "data":{

20 "skipAction":[],

21 0:{"text":"CONVERSATION\_LEVEL1\_START0",

22 “name":"Lino",

23 "actor":"lino",

24 "emotion":"normal",

25 "currentAction":[],

26 "nextAction":{1:[["goTo",[1]]]}

27 },

28 1:{

29 "text":"CONVERSATION\_LEVEL1\_START1",

30 "name":"Ada",

31 "actor":"ada",

32 "emotion":"normal",

33 "currentAction":[],

34 "nextAction":{"CONVERSATION\_LEVEL1\_START1\_1":[["goTo",[0]]],

35 "CONVERSATION\_LEVEL1\_START1\_2":[["goTo",[2]]]}

36 },

37 2:{

38 "text":"CONVERSATION\_LEVEL1\_START2",

39 "name":"Lino",

40 "actor":"lino",

41 "emotion":"normal",

42 "currentAction":[],

43 "nextAction":{1:[["endConversation",""]]}

44 }

45 }

46 },

47 {

48 "condition": {"gameover":1},

49 "data":{

50 "skipAction":[["showResultMenu",[""]]],

51 0:{"text":"CONVERSATION\_LEVEL1\_END0",

52 "name":"Ada",

53 "actor":"ada",

54 "emotion":"happy",

55 "currentAction":[],

56 "nextAction":{1:[["endConversation",""],

57 ["showResultMenu",[""]]]}

58 }

59 }

60 ],

61 mapData:[

62 {x=0,y=2,z=0,groundType= 1,objectAttribute={}},

63 {x=0,y=1,z=0,groundType= 2,objectAttribute={"houseValue" : 6,

64 "direction": "left"}},

64 {x=1,y=1,z=0,groundType= 0,objectAttribute={}},

65 {x=2,y=1,z=0,groundType= 1,objectAttribute={}},]

66 }

67 }

Gambar 4.dataLevel Contoh struktur data level

**Daftar Pustaka**

Bebbington, S., 2014, *What is Programming*, <http://yearofcodes.tumblr.com/what-is-programming> (diakses pada 25 Februari 2015)

Bureau of Labor Statistics, U.S. Department of Labor, *Occupational Outlook Handbook*, 2014-15 Edition, Software Developers, <http://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm> (diakses pada 25 Februari 2015)

Friesen, J., 2014, *Learn Java for Android Development*, Apress, New York

Iuppa, N. & Borst, T., 2012, *End-to-End Game Development: Creating Independent Serious Games and Simulations from Start to Finish*, Taylor & Francis, New York

Kazigmolu, C., *et al*, 2012, *A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming*, Procedia - Social and Behavioral Sciences 47(2012), hal 1991-1999

Kingsley-Hughes, A. & Kingsley-Hughes, K. 2005, *Beginning Programming*, John Wiley & Sons, New York

Moreno, J., 2012, *Digital Competition Game to Improve Programming Skills*, Educational Technology & Society, 15(3), hal 288-297

Muratet, M., Torguet, P., dan Jessel, J.P., 2008, *Learning Programming with an RTS based Serious Games*, Serious Games on the Move 08

Muratet, M., *et al*, 2010, *Experimental Feedback on Prog&Play, a Serious Game for Programming Practice*, The Eurographics 2010

Net Applications, 2015, Mobile ablet Operating System Market Share, <http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1&qptimeframe=M> (diakses pada 25 Februari 2015)

Pressman, R.S., 1997, *Software Engineering : A Practioner's Approach.*, 4th., McGrawHill., New York

Shiffman, D., 2009, *Learning Processing: A Beginner’s Guide to Programming Images, Animation, and Interaction*, Morgan Kaufmann, Massachussets

Wang, W., 2011, *Beginning Programming For Dummies*, John Wiley & Sons, New York.

Zhang, J. & Lu, J., 2014, *Using Mobile Serious Games for Learning Programming*, INFOCOMP 2014 : The Fourth International Conference on Advanced Communications and Computation

Zyda, M., 2005, *From Visual Simulation to Virtual Reality to Games*, IEEE Computer Society (2005), hal. 25-32