Bab IV. PEMBAHASAN

4.1 Riset dan Penyusunan Konsep *Game "Wana Warrior"*

"Wana Warrior" adalah *game* bercerita tentang perjuangan para satwa-satwa hutan untuk mempertahankan tempat tinggal mereka dari manusia-manusia yang ingin mengalihfungsikan hutan untuk keperluan industri. *"Wana Warrior"* sendiri bermakna para ksatria dari hutan yang ingin melindungi tempat hidup mereka dari serangan manusia. Melalui *game* ini diharapkan pemain dapat mengerti akibat dari alih fungsi hutan secara massal terhadap keberlangsungan satwa-satwa yang hidup dengan mengandalkan hutan sebagai tempat untuk hidup. Dalam produksi *game "Wana Warrior"* dibutuhkan perancangan tema, *gameplay*, cerita, alur logika, program, dan perlengkapan lain untuk membuat *game* seperti *sprite*, *sound*, dan lainnya.

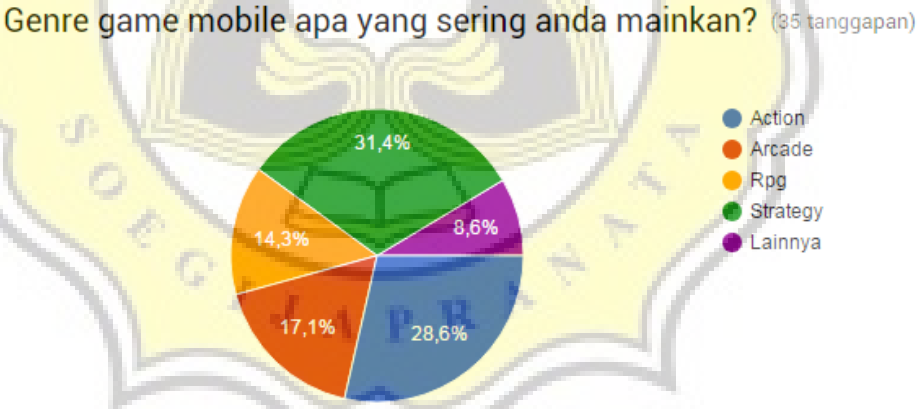
Untuk merumuskan desain *game* diperlukan data yang dapat menunjang desain *game*. Untuk mendapatkan data dilakukan survey dengan metode kuesioner. Kuesioner dibagikan kepada 35 responden dengan rentang usia 15-25 tahun. Jumlah responden laki-laki dan perempuan terbagi seimbang dengan jumlah 18 laki-laki dan 17 perempuan.

Pada pre-test, responden diberi beberapa pertanyaan yang menyangkut kebiasaan responden dalam bermain mobile *game*, pengujian pengeahuan mengenai satwa dan keanekaragaman hayati di Indonesia, dan ketertarikan mereka mengenai *game* sebagai media informasi mengenai penyelamatan satwa.

Pada gambar 4.1 menunjukkan data sebagian besar dari responden bukan merupakan pemain *game mobile* yang aktif. Hanya sejumlah kecil saja yang memainkan *game mobile* di atas 2 jam.

Gambar 4.1 Diagram lingkaran durasi responden memainkan game mobile

Meskipun mayoritas merupakan bukan pemain *game mobile* aktif, tapi *game mobile* yang responden mainkan cukup beragam. Pada gambar 4.2 menunjukkan data yang bisa ditarik kesimpulan bahwa genre *game* strategi dan *action* mendapatkan cukup banyak penggemar.



Gambar 4.2 Diagram lingkaran genre game mobile yang dimainkan responden

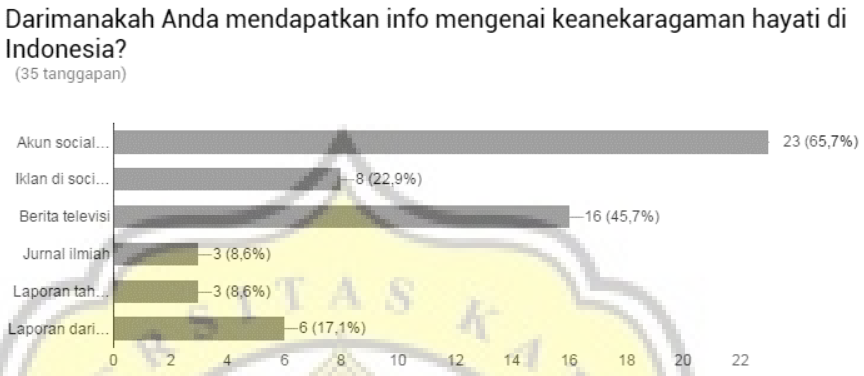
Dari gambar diagram dan beberapa *game* yang disebutkan, yang ditunjukkan pada gambar 4.3, semua responden belum pernah memainkan *game* dengan tema penyelamatan lingkungan. Terlihat dari judul *game* yang disebutkan tidak relevan dengan *game* yang sudah ada mengenai penyelamatan satwa.

Pada bagian pengujian pengetahuan responden mengenai penyelamatan satwa, responden diberikan 10 pertanyaan mengenai satwa yang dilindungi dan kondisi alam yang mendukung keberadaan satwa yang dilindungi. Pada tabel analisa hasil tes dapat dilihat jumlah responden yang memiliki nilai pre-test di atas 6 hanya sebesar 15 orang dari 30 orang. Hal ini menunjukkan bahwa masih kurangnya perhatian dan pengetahuan akan kondisi satwa dan alam di Indonesia.

Gambar 4.3 Diagram lingkaran dan keterangan mengenai game yang tentang penyelamatan alam yang pernah dimainkan responden

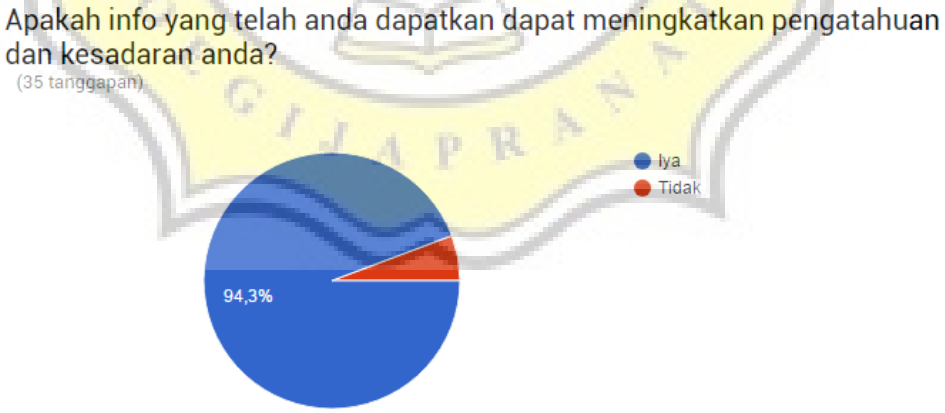
Pada diagram batang yang ditunjukkan pada gambar 4.4, menunjukkan bahwa media sosial masih menjadi ujung tombak penyampaian informasi dan kampanye mengenai penyelaamatan satwa di Indonesia. Media televisi juga menjadi media yang penting sebagai kampanye penyelamatan satwa tetapi masih di

bawah media social. Sedangkan media yang menyajikan data yang lebih mendetil dan rinci kurang diminati. Hal ini mungkin terjadi karena penggunaannya yang ditujukan bukan untuk konsumsi masyarakat awam yang lebih menyenangi media yang singkat dan menyenangkan untuk dibaca.



Gambar 4.4 Diagram batang media dimana responden mendapatkan data mengenai penyelamatan satwa

Hasil pada kuesioner yang ditunjukkan pada gambar 4.5 menunjukkan hampir semua responden merasa terdeukasi dengan info yang sudah disediakan terdapat secara umum. Hanya sebagian kecil yang merasa belum cukup teredukasi dengan adanya info yang sudah tersedia.



Gambar 4.5 Diagram lingkaran pendapat responden mengenai media umum dengan materi penyelamatan satwa

Diagram pada gambar 4.6 menunjukkan antusiasme yang tinggi dari responden apabila *game* menjadi media iformasi penyelamatan satwa di Indonesia.

Hanya satu responden yang memiliki ketertarikan kecil dan 4 responden dengan ketertarikan yang sedang.

Gambar 4.6 Diagram batang ketertarikan responden terhadap game tentang penyelamatan satwa

Data yang masih dengan rasio yang kurang lebih sama, pada gambar 4.7, menunjukkan bahwa banyak yang seutuju apabila *game* menjadi media informasi penyelamatan satwa di Indonesia.

Gambar 4.7 Diagram batang pendapat responden apabila game menjadi media informasi penyelamatan satwa

Data yang telah didapatkan didapatkan kesimpulan bahwa pengethuan masyarakat mengenai keanekaragaman hayati satwa di Indonesia masih perlu ditingkatkan. Selain itu, media yang berupa visual lebih banyak menarik minat

masyarakat daripada laporan dengan teks yang detail. Game juga dianggap sebagai media yang akan menarik mereka untuk lebih mengetahui informasi mengenai keanekaragaman hayati satwa yang dilindungi. Oleh karena itu perlu adanya perumusan desain game yang menarik serta mengedukasi masyarakat mengenai keanekaragaman hayati satwa di Indonesia.

4.2 Desain gameplay "Wana Warrior"

Dengan data yang telah didapat, maka rumusan *Game* "*"Wana Warrior"*" dapat dibuat. Pada data di atas dapat diambil kesimpulan *Game* "*"Wana Warrior"*" dibuat dengan genre strategi dengan menampilkan info-info singkat dengan grafis yang menarik layaknya info yang tampil di media sosial. *Game* "*"Wana Warrior"*" dibuat agar menarik pemain untuk terus menyelesaikan misi. Dengan memberikan info yang singkat pada tiap loading screen, cerita pada tiap level, dan penjelasn cerita pada biografi karakter akan membuat pemain dapat bermain sambil belajar tanpa merasa belajar. Semakin sering pemain memainkan *Game "Wana Warrior*", maka semakin banyak ilmu yang didapat oleh pemain.

Game "Wana Warrior" merupakan *game* dengan genre strategi yang memiliki sentuhan aksi dua dimensi yang melibatkan satu player. Pada halaman awal hanya aka nada dua tombol. Tombol main untuk masuk ke menu level dan tombol suara untuk mengatur suara. Tampilan halaman awal seperti yang ditunjukkan pada gambar 4.8.

Gambar 4.8 Halaman awal game

Pada bagian ke dua, gambar 4.9, setelah halaman awal adalah halaman peta yang menunjukkan level seperti yang ditunjukkan oleh gambar 4.9. Selain itu terdapat juga tombo-tombol yang mebghubungkan ke fitur-fitur *game*. Tombol level ditunjukkan oleh bendera pada peta. Merah padam berarti level belum terbuka dan belum bisa dimainkan, merah menyala berrti level telah terbuka dan dapat dimainkan, dan warna biru berarti level terbuka dan telah dimainkan serta dapat dimainkan kembali.

Gambar 4.9 Halaman misi/peta

Pada *scene* Warrior, pemain bisa mengkostumisasi karakter apa saja yag mau digunakan, mengurutkan karakter sesuai keinginan, dan menaikkan level karakter. Pada *scene* ini, yang ditunjukkan pada gambar 4.10 dan 4.11, juga terdapat

fitur mengenai detil karakter. Pada detil karakter ini pemain bisa mengetahui kemampuan karakter serta cerita dari tiap karakter. Cerita karakter ini juga merupakan fitur utama dalam penyampaian pesan mengenai pelestarian keanekaragaman hayati di Indonesia. Dalam cerita karakter diberikan cerita asal usul karakter dan memberikan rasa pada karakter itu sendiri.

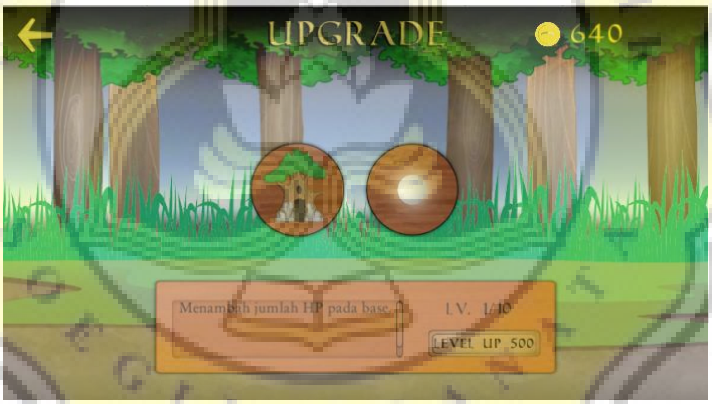
Gambar 4.10 Tampilan halaman Warrior

Gambar 4.11 Tampilan detil karakter

Fitur selanjutnya adalah fitur item shop dan upgrade center. Pada item shop, gambar 4.12, terdapat dua buah item yang bisa dibeli. Item dapat berguna untuk mempermudah permainan. Item petir dapat digunakan untuk menyerang base musuh sebesar 10% sekali serang. Sedangkan item 50% HP *Homebase* berfungsi untuk membuat HP *Homebase* menjadi 50% apapun keadaannya. Pembelian item dibatasi jumlah *Wana gold* yang dimiliki. Fitur *upgrade*, gambar 4.13, sendiri

memiliki dua jenis *upgrade* yang bisa dibeli. Pertama adalah *upgrade* Tambah HP *Homebase* yang berfungsi menambah HP *Homebase* setelah *upgrade* dibeli. Dan Penambah Kecepatan SP yang berfungsi untuk mempercepat penambahan SP dalam *in-game.*

Gambar 4.12 Tampilan Item shop



Gambar 4.13 Tampilan Upgrade center

Gambar 4.14 dan 4.15 adalah tampilan pada *In-game*. *In-game* menggunakan sistem perang dengan beberapa karakter yang dengan tujuan untuk menghancurkan base lawan. Permainan dinyatakan berakhir apabila *hitpoints* salah satu base dari kedua sisi habis. Pemain diharuskan menggunakan satu set kartu karakter yang berisi lima kartu sekali permainan. Pemain juga dapat menggunakan 2 *item* yang masing-masing memiliki fungsi berbeda guna mempermudah pemain untuk menyelesaikan permainan. Sebelum memulai permainan akan disajikan

cerita dari tiap karakter mengenai latar belakang karakter bisa tergabung menjadi *"Wana Warrior"*. Cerita diberikan dalam bentuk teks dan animasi yang terus langsung berlanjur ke *in-game*.

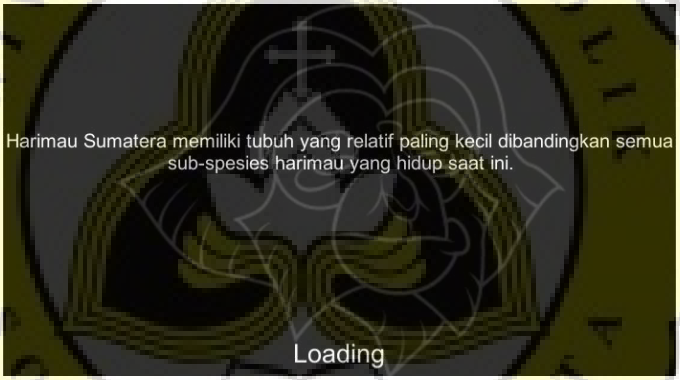
Gambar 4.14 Tampilan In-game (dari homebase pemain)

Gambar 4.15 Tampilan In-game (dari homebase musuh)

Untuk memberikan materi edukasi dalam *game*, penyampaian pesan dilakukan dalam beberapa cara, antara lain melalui *loading screen*, cerita karakter, dan melalui profil singkat pada detil karakter. Pesan yang diberikan dibuat dengan ringan namun memiliki pesan yang jelas serta info yang bermanfaat untuk mendukung pesan. Dengan menggunakan teori *agenda setting*, penggambaran kondisi lingkungan dibuat seakan pemain merasakan bagaimana kondisi satwa di

alam. Pada *game "Wana Warrior",* isu-isu umum diberikan pada *loading screen* dan isu-isu yang lebih tajam diberikan pada cerita karakter.

Loading screen adalah layar yang dibuat sebagai transisi antar *scene. Loading screen* yang ditampilkan akan berisi info singkat mengenai isu-isu keanekaragaman hayati di Indonesia seperti yang ditampilkan pada gambar 4.16. *Loading screen* akan ditampilkan terus pada setiap perpindahan *scene* dengan durasi 4 detik. Durasi 4 detik dipilih karena dalam waktu ini dianggap tidak terlalu lama dan tidak terlalu sebentar. Selain pada *loading screen,* terdapat juga pesan yang diberikan setelah cerita pada karakter. Pesan yang diberikan memiliki hubungan dengan cerita yang ditampilkan sebelumnya.



Gambar 4.16 Tampilan Loading screen

4.2.1 Level permainan

Level permainan dalam *game "Wana Warrior"* terbagi menjadi 6 level. Setiap level akan diberikan cerita dari masing-masing karakter sebagai intro. Cerita terinspirasi dari keadaan nyata di lapangan dengan ditambah konflik drama yang membuat cerita menarik Dan tiap level akan mengikuti cerita yang dijadikan sebagai intro.

Pada level pertama akan menampilkan cerita dari Otan. Otan merupakan Orangutan sumatera yang hidup damai bersama kawanannya. Tapi pada suatu hari

datanglah sekelompok orang yang mengatasnamakan kelompok mereka sebagai "Helm Kuning" yang ingin menebangi hutan dan membuat hutan menjadi lahan perkebunan sawit. Semua orangutan kalangkabut menyingkir dari rumah mereka yang dengan cepat digunduli Helm Kuning. Namun seekor orangutan bernama Otan berani melawan. Namun dalam perlawanannya dia malah ditembak. Dalam keadaan sekarat datan pertolongan dari dewa hutan dan memberika kekuatan ajaib untuk Otan. Jadilah Otan hidup kembali dengan memiliki kekuatan baru. Namun dia diberi tugas oleh dewa hutan untuk menjaga hutan dan diberi kabar kelak dalam perjuangnnya dia akan ditemani oleh beberapa satwa yang akan bersama berjuang bersamanya.

Pada level 1 pemain akan memulai petualangannya dengan satu karakter pemimpin, yaitu Otan. Pada level ini pemain akan melawan musuh dengan menggunakan Otan dan dengan tingkat kesulitan yang mudah. Potongan cerita Otan ditunjukkan pada gambar 4.17 dan level permainan pada gambar 4.18.

Gambar 4.17 Tampilan cerita Otan (level 1)

Gambar 4.18 Tampilan level 1

Level kedua adalah cerita tentang Yongki yang ditunjukkan pada gambar 4.19. Cerita ini terinspirasi oleh tebunuhnya seekor gajah patroli di kawasan Taman Nasional Bukit Barisan. Yongki dalam Wana Warrior sama seperti Yongki dalam kehidupan nyata, seekor gajah patroli. Dalam patrolinya Yongki tidak sengaja memakan racun yang dipasang oleh Helm Kuning yang juga menjalankan bisnis ilegal penjualan gading gajah. Dalam keadaa sekarat dan dati gading sudah dipotong datanglah Otan yang tidak engaja lewat. Dengan kekuatannya, Otan mengusir anggota Helm Kuning dan menyembuhkan Yongki dengan kekuatannya. Yongki bisa hidup dengan baik lagi. Namun gading nya yang patah tidak bisa kembali seperti semua. Lalu dia memutuskan menggunakan gadingnya sebagai senjata.

Pada level 2, gambar 4.20, pemain akan melanjutkan petualangannya dengan karakter sebelumnya dan karakter yang baru. Pada level ini pemain akan melawan musuh dengan menggunakan Otan dan Yongki dan dengan tingkat kesulitan yang lebih sulit dari level sebelumnya.

Gambar 4.19 Tampilan cerita Yongki (level 2)

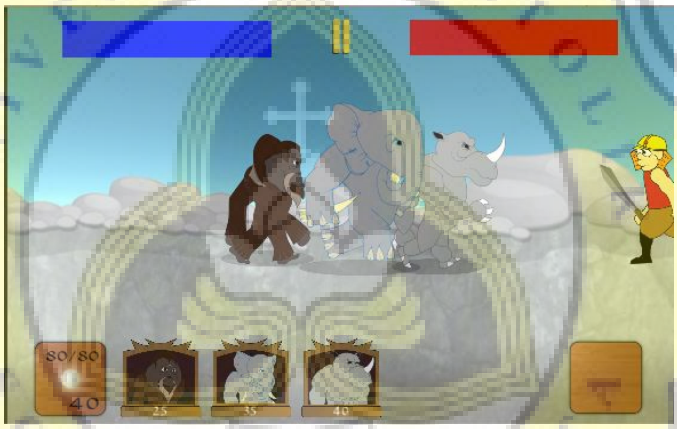
Gambar 4.20 Tampilan level 2

Cerita ketiga adalah cerita mengenai Datu, seekor badak sumatera, yang ditunjukkan pada gambar 4.21. Nama Datu diambil dari nama badak sumatera yang lahir di Taman Nasional Way Kambas pada tahun 2012 yang bernama Andatu. Datu berlatar belakang seekor badak yang hidup soliter. Namun karena tempat hidupnya habis, dia sering mendekat ke kawasan perumahan manusia untuk mengambil makanan. Karena hal itu dia sering dianggap sebagai penganggu. Namun pertemuannya dengan Otan membuat dia dapat menjadi badak yang lebih baik dan tidak mendekat ke lingkungan mnusia kecuali utuk melawan Helm Kuning bersama Wana Warrior.

Pada level 2, yang ditunjukkan pada gambar 4.22, pemain akan melanjutkan petualangannya dengan karakter sebelumnya dan karakter yang baru. Pada level ini

pemain akan melawan musuh dengan menggunakan Otan, Yongki, dan Datu dengan tingkat kesulitan yang lebih sulit dari level sebelumnya.

Gambar 4.21 Tampilan cerita Datu (level 3)



Gambar 4.22 Tampilan level 3

Cerita keempat adalah cerita mengenai Bappu, ditunjukkan pada gambar 4.23. Bappu adalah Owa Jawa yang dibawa manusia untuk hidup dalam penangkaran bersama keluarganya. Namun dalam perjalanan kendaraan yg mereka tumpangi mengalami kecelakaan. Sejak saat itu dia dan keluarganya kabur dan hidup di hutan sumatera. Suatu hari Helm Kuning datang untuk membabat hutan rumahnya. Dalam pelarian, anaknya jatuh dan dibawa oleh seorang anggota Helm Kuning untuk dijual. Dalam pencarian anaknya yang hamper membuatnya stres,

Bappu bertemu dengan Otan dan menceritaka kisahnya. Sejak saat itu dia memutuskan bergabung dengan Wana Warrior.

Pada level 4, ditunjukkan pada gambar 4,24, pemain akan melanjutkan petualangannya dengan karakter sebelumnya dan karakter yang baru. Pada level ini pemain akan melawan musuh dengan menggunakan Otan, Yongki, Datu, dan Bappu dengan tingkat kesulitan yang lebih sulit dari level sebelumnya.

Gambar 4.23 Tampilan cerita Bappu (level 4)

Gambar 4.24 Tampilan level 4

Cerita kelima adalah cerita dari Harimo dan Kijang, ditunjukkan pada gambar 4,25. Harimo adalah seekor harimau sumatera yang sedang mengejar mangsanya, yaitu Kijang. Dalam pengejarannya, Kijang malah berlari memasuki wilayah Helm Kuning. Sontak para anggota Helm kuning kaget dan menembak Harimo. Harimo yang sekarat hampir saja dikuliti. Namun datanglah Otan

menyelamatkan. Otan dan Wana Warrior melawan Helm Kuning dan menyelamatkan hidup Harimo dengan kekuatannya. Harimo bisa hidup dengan kekuatan baru dan bergabung dengan Wana Warrior. Dalam cerita lain pada saat harimo diserang dan diselamatkan, Kijang yang kabur kembali mendekat dan melihat pertempuran. Dia diajak kembali oleh Otan dan berikan kekuatan juga. Tanpa sengaja ada peralatan helm kuning yg tertinggal dan Kijang mengambil dan memakainya.

Pada level 5, ditunjukkan pada gambar 4,26, pemain akan melanjutkan petualangannya dengan karakter sebelumnya dan karakter yang baru. Pada level ini pemain akan melawan musuh dengan menggunakan Otan, Yongki, Datu, Bappu, dan Harimo dengan tingkat kesulitan yang lebih sulit dari level sebelumnya. Karakter Sambar akan disimpan dalam menu Warrior karena dalam satu permainan hanya dapat menggunakan 5 karakter.

Gambar 4.25 Tampilan cerita Harimo dan Sambar (level 5)

Gambar 4.26 Tampilan level 5

Setelah semua cerita terbuka dilanjutkan tidak lagi cerita per karakter, aka nada satu *level* tanpa cerita yang akan membawa pemain pada cerita Wana Warrior yang berjuang bersama dengan lengkap.

4.3 Aset *game*

Aset *game* pada *game "Wana Warrior"* meliputi tombol, karakter, setting lingkungan, dan *GUI* (*Graphic User Interface*) lainnya yang berupa gambar, serta audio yang menujnang suasana *game*. Pembuatan aset yang berua gambar dibuat dengan menggunakan *software* Adobe Illustrator dengan bantuan Adobe Photoshop untuk manipulasi gambar. *Game "Wana Warrior"* menggunakan teknik *sprite sheet* karena *game* menampilkan *action* yang dilakukan karakter.

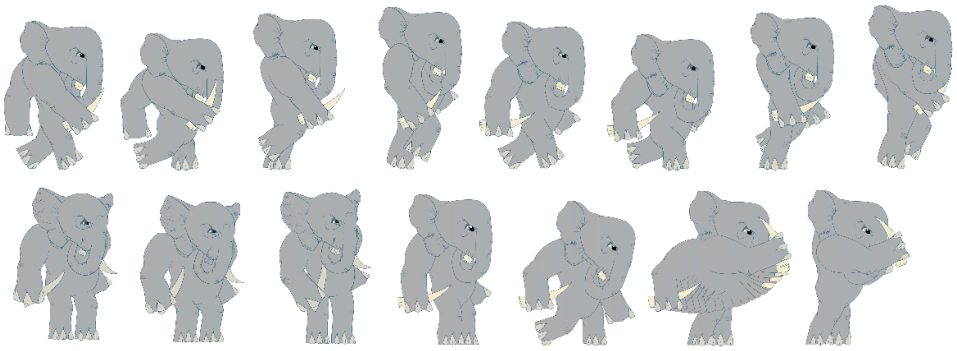
Pembuatan asset gambar menampilkan tema yang disesuaikan dengan *game*. Setting menggunakan karakter hutan, karakter dibuat humanoid untuk menyesuaikan cerita, dan *GUI* yang bertema kayu. Selain itu untuk tombol karakter menggunakan ornamen kebudayaan dari Indonesia sebagai identitas *game*. Yaitu

menggunakan bentuk gunungan pada background-nya serta ditambahkan bentuk atap rumah gadang di atas kotak tombol.

4.3.1 Aset *sprite* karakter

Dalam *game "Wana Warrior"* terdapat 6 karakter utama dan 4 jenis karakter musuh. Karakter utama, pada gambar 4.27, 4.48, 4.29, 4.30, 4,32, merepresentasikan karakter Wana Warrior yang berwujud satwa yang dilindungi yang memiliki kekuatan. Karakter dibuat dengan pola *humanoid* agar dapat memberika efek kekuatan. Sedangkan untuk karakter musuh dibuat dengan 2 pola yang sama dan dengan ciri berhelm kuning sebagai identitas dari geng Helm Kuning. Karakter musuh, pada gambar 4.33, 4.34, 4.35, 4.36, dibuat dengan 4 jenis senjata yaitu senapan laras panjang, pistol, tombak, dan golok. Empat senjata dipilih karena merupakan alat yang biasa masyarakat gunakan untuk masuk ke hutan dan berburu.

Gambar 4.27 Spritesheet karakter Otan



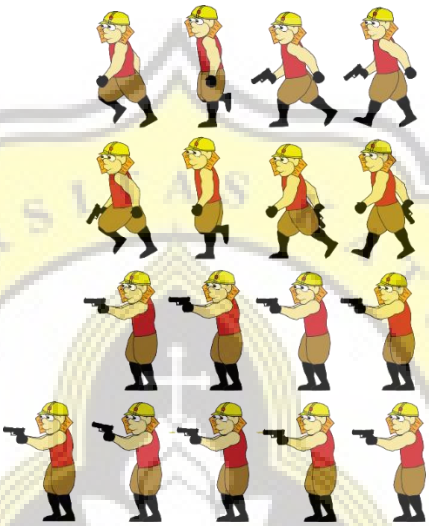
Gambar 4.28 Spritesheet karakter Yongki

Gambar 4.29 Spritesheet karakter Datu

Gambar 4.30 Spritesheet karakter Bappu

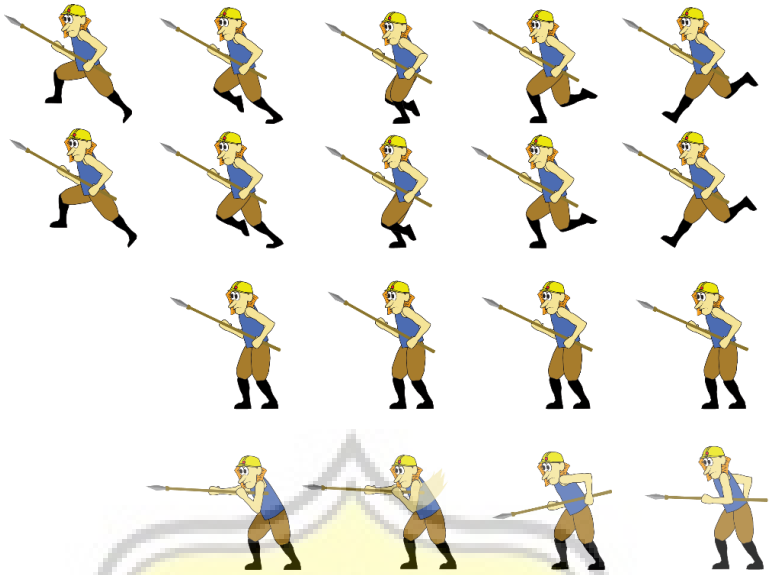
Gambar 4.31 Spritesheet karakter Harimo

Gambar 4.32 Spritesheet karakter Sambar

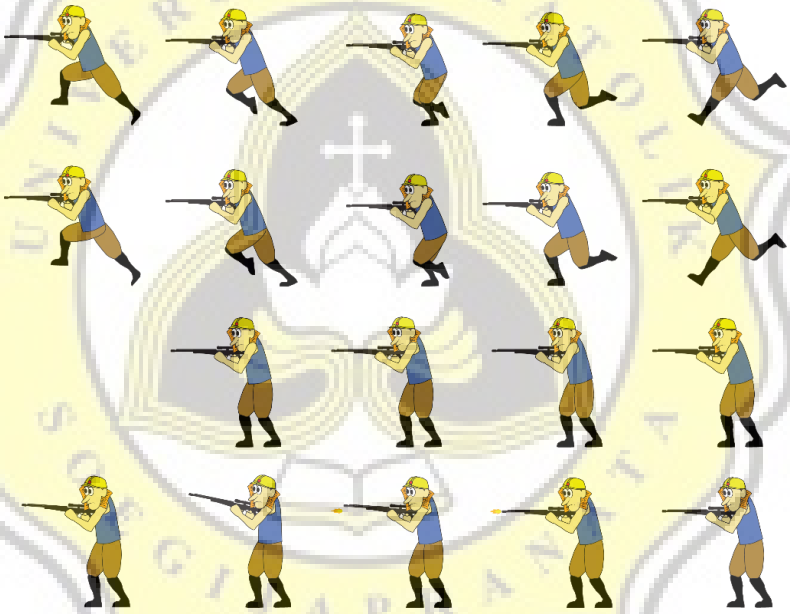


Gambar 4.33 Spritesheet karakter Geng Helm Kuning pistol

Gambar 4.34 Spritesheet karakter Geng Helm Kuning Golok



Gambar 4.35 Spritesheet karakter Geng Helm Kuning Tombak



Gambar 4.36 Spritesheet karakter Geng Helm Kuning Senapan laras panjang

Pada gambar 4.37 menunjukkan GUI tombol karakter. Aset ini digunakan pada saat scene *Warrior setting* dan *in-game.* Asset ini dibuat dengan memadukan unsur-unsur local. Di atas terdapat siluet atap rumah gadang, dan pada *backround* terdapat siluet gunungan wayang dipadukan warna unsur kayu sebagai representasi hutan.

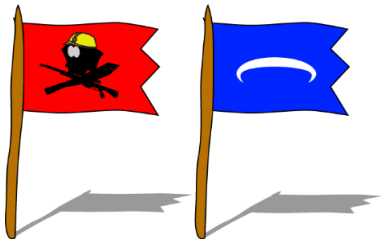
Gambar 4.37 Sprite GUI tombol karakter

4.3.2 Grafis Antarmuka/GUI

Tampilan antarmuka atau *GUI* (Graphic user interface) dalam *game* "Wana Warrior" dibuat dengan tema yang dekat dengan hutan. Dengan warna-warna dan tekstur kayu dalam tampilan antarmuka memberkan kesan yang selaras dengan *game* yang bertema lingkungan dan hewan. Asset ditunjukkan pada gambar 4.38, 4.39, 4.40, dan 4.41.

Gambar 4.38 Sprite GUI tombol-tombol dan koin

Gambar 4.39 Sprite judul scene dan tombol-tombol



Gambar 4.40 Sprite tombol level (merah belum dimenangkan dan biru sudah pernah dimenangkan)

4.3.3 Latar dan setting *game*

Latar dari *game "Wana Warrior"* mengambil tema di Hutan Sumatera. Terdapat beberapa set latar yang merepresentasikan kondisi alam di Sumatera. Yang dipakai adalah tampilan hutan yang tandus, hutan dalam yang lebat, bebatuan dan latar dari pegunungan Bukit Barisan. Gambar 4.41, 4.42, 4.43, 4.44, 4.45, 4.46, dan 4.47 merupakan asset yang digunakan pada game untuk membentuk setting yang diinginkan. Asset ini dapat dipadukan sesuai dengan kebutuhan.

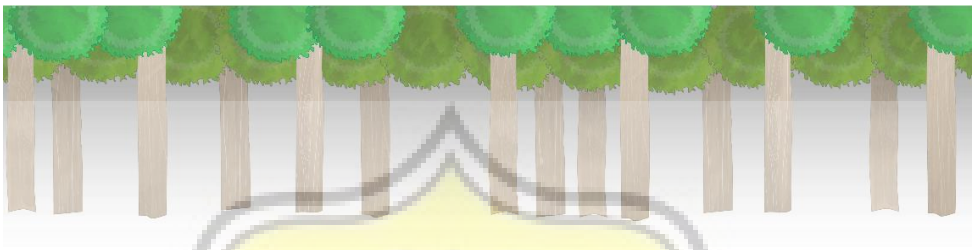
Gambar 4.41 Sprite latar pegunungan Bukit Barisan

Gambar 4.42 Sprite latar tanah berbatuan

Gambar 4.43 Sprite latar tanah tandus

Gambar 4.44 Sprite latar padang rumput

Gambar 4.45 Sprite latar hutan dalam yang rapat



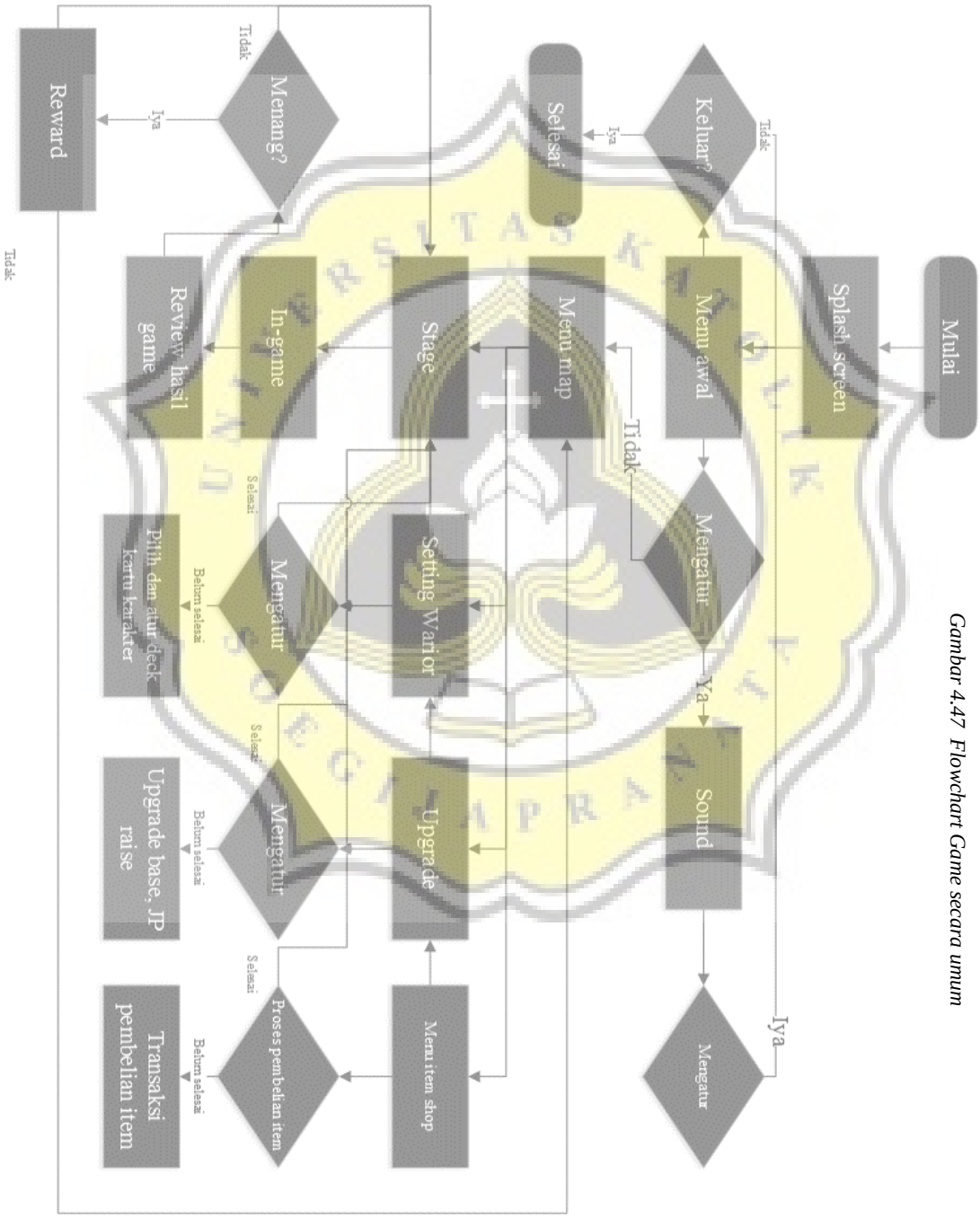
Gambar 4.46 Sprite latar hutan luar yang terbuka

4.4 Pemrograman *game* "Wana Warrior"

Game dibuat dengan menggunakan bantuan *game engine* Unity 2D. Unity 2D merupakan *game engine* yang umum digunakan untuk membuat *game*. Adanya fasilitas gratis serta fitur yang cukup lengkap membuat Unity 2D menjadi pilihan dalam pembuatan *game "Wana Warrior"*. Pada *game "Wana Warrior"* menggunakan bahasa pemrograman C# dengan monodevelop karena bersifat *cross-platform* dan mudah digunakan. Untuk keperluan penyimpanan data pada *game* menggunakan metode penyimpanan json. Json (*JavaScript Object Notation*) adalah format pertukaran data yang ringan dan mudah dibaca dan ditulis. Json didasarkan pada bahasa pemrograman *javascript*. Json yang digunakan adalah Litjson dan JSON.Net. Penyimpanan data menggunakan json dipilih karena lebih mudah untuk dibaca dan dimanipulasi saat pengembangan.

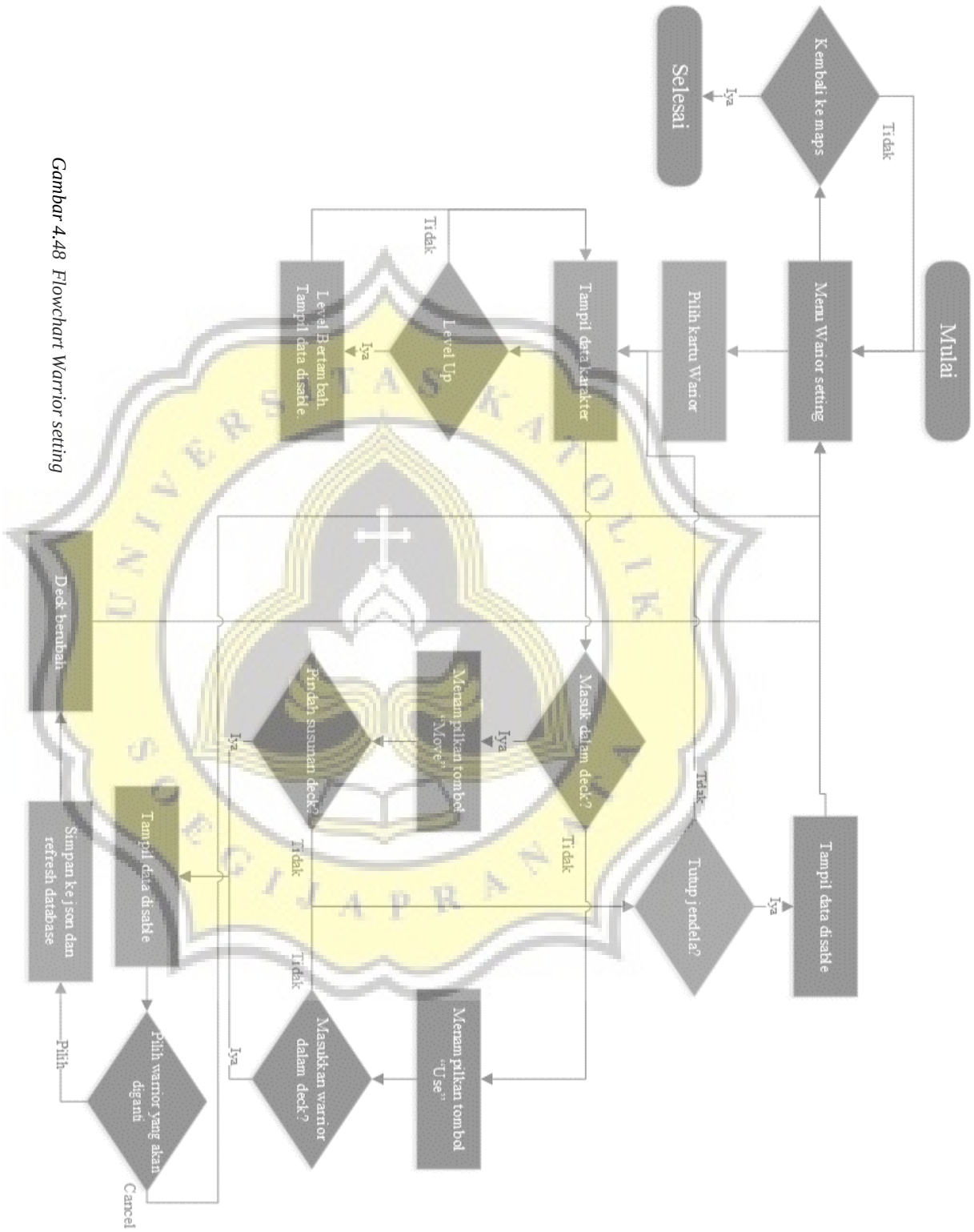
4.4.1 Alur pemrograman game secara umum

Gambar 4.47 merupakan alur proses secara umum dari *game "Wana Warrior"*. Pada gambar tersebut menjelaskan alur pemain dalam memainkan game dan fitur-fitur yang tersedia dalam *gameplay*.



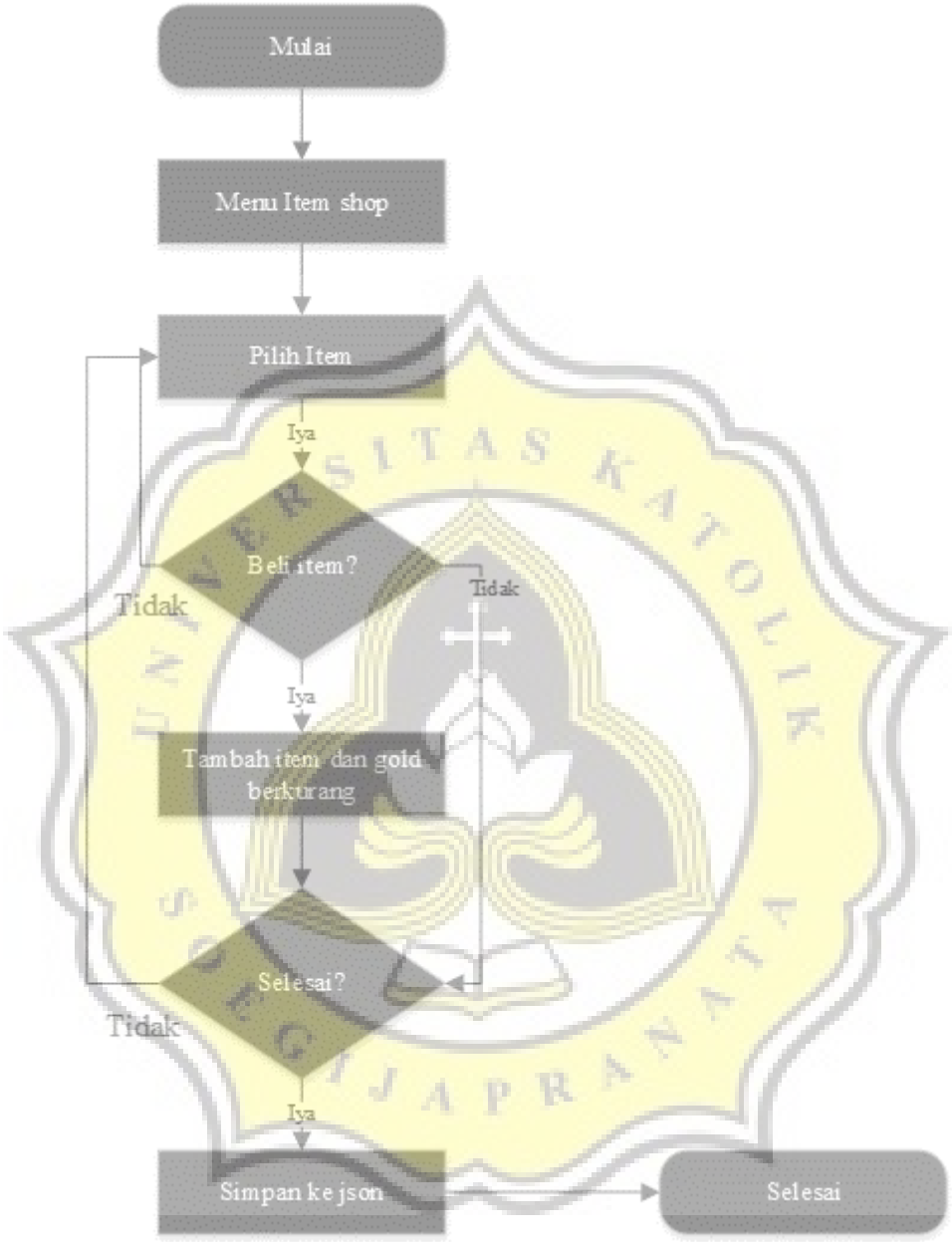
Gambar 4.47 Flowchart Game secara umum

Pada gambar 4.48 menunjukkan proses yang dilakukan program pada saat pemain melakukan kegiatan pada menu *Warrior Setting.*



Gambar 4.48 Flowchart Warrior setting

Pada gambar 4.49 menunjukkan proses yang dilakukan program pada saat pemain melakukan kegiatan pada menu *Item Shop*.

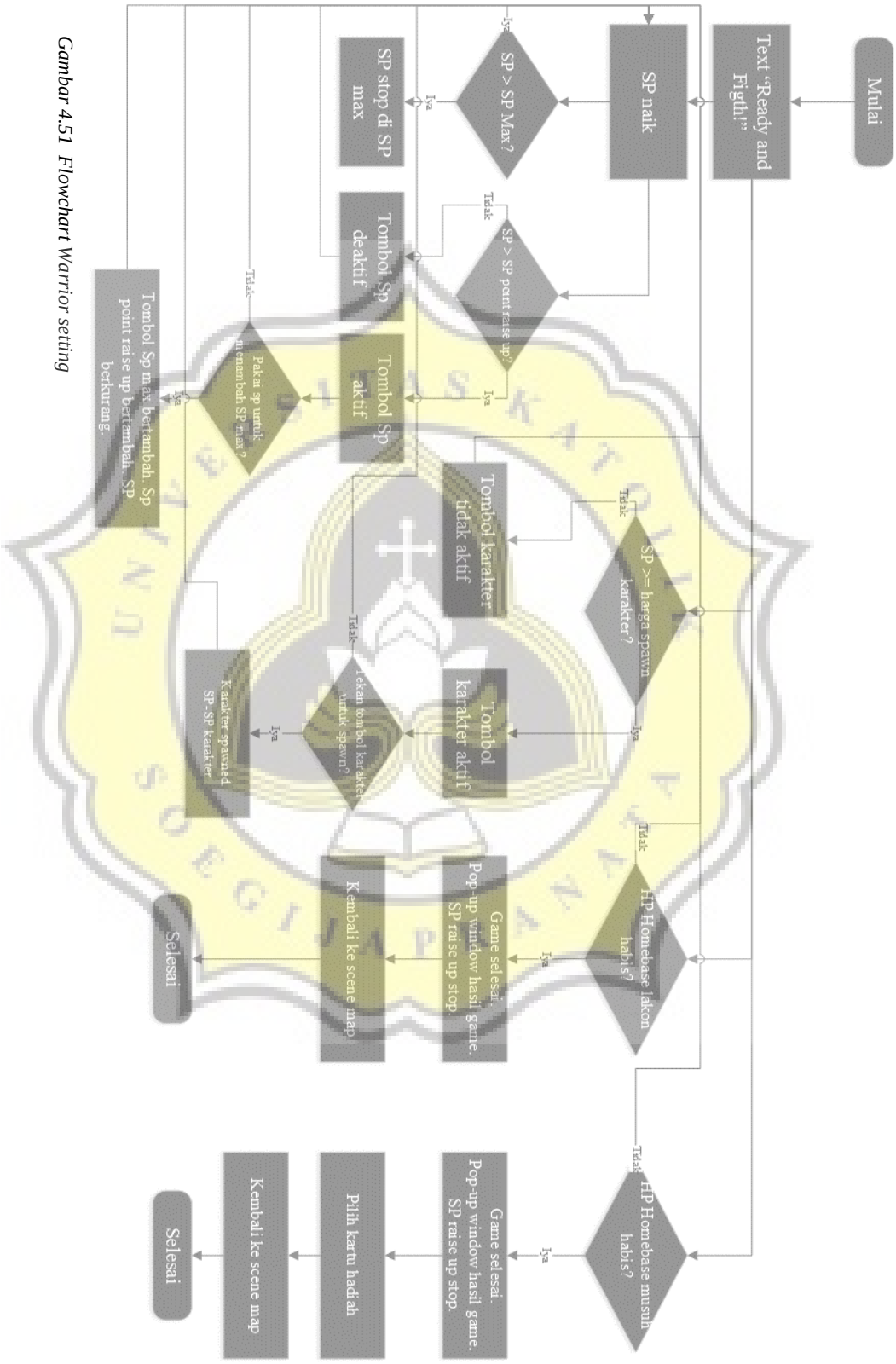


Gambar 4.49 Flowchart Item shop

Pada gambar 4.50 menunjukkan proses yang dilakukan program pada saat pemain melakukan kegiatan pada menu *Upgrade center*.

Gambar 4.50 Flowchart Upgrade center

Pada gambar 4.51 menunjukkan proses yang dilakukan program pada *gameplay* pada game.



Gambar 4.51 Flowchart Warrior setting

4.4.2 Scene halaman awal

Pada scene startpage hanya berisi pengaturan suara hidup dan mati, tombol main, tombol close, dan terdapat *script* untuk menduplikat json ke path yang bisa dimanipulasi. Pada *script* cekdatabaseawal.cs terdapat fungsi Pada fungsi detilkar(), karset(), duititem(), dan koleksi() yang terdapat perintah untuk membaca semua json dan menduplikatnya dari *path Resource* ke *persistentDataPath*. Hal ini dilakukan karena pada path *Resource* bersifat *Read-only* dan *game* butuh untuk memanipulasi json untuk perkembangan *game*.

```
void detilkar(){
    jsonstringDetilkar = Resources.Load<TextAsset> ("db/detilkarakter").text;
    ambildataDetilkar = JObject.Parse (jsonstringDetilkar);
    int i;
    int[] id;           id = new int[5];
    string[] namakar;       namakar = new string[5];
    string[] penjelasan;    penjelasan = new string[5];
    string[] tipe;          tipe = new string[5];
    int[] level;        level = new int[5];
    int[] sp;           sp = new int[5];
    int[] hp;           hp = new int[5];
    int[] atkpoin;      atkpoin = new int[5];
    int[] atkspd;       atkspd = new int[5];
    int[] spd;          spd = new int[5];
    int[] wktprod;      wktprod = new int[5];
    int[] hargalvlup;   hargalvlup = new int[5];
    string[] btnket;    btnket = new string[5];
    string[] btngame;   btngame = new string[5];
    string[] karakter;  karakter = new string[5];
    string[] profpict;  profpict = new string[5];

    for (i = 0; i < 5; i++) {
        id [i] = Convert.ToInt32 (ambildataDetilkar ["karakter"] [i] ["id"]);
        namakar [i] = ambildataDetilkar ["karakter"] [i] ["namakar"].ToString();
        penjelasan[i] = ambildataDetilkar ["karakter"] [i] ["penjelasan"].ToString();
        tipe[i] = ambildataDetilkar ["karakter"] [i] ["tipe"].ToString();
        level[i] = Convert.ToInt32 (ambildataDetilkar ["karakter"] [i] ["level"]);
        sp[i] = Convert.ToInt32 (ambildataDetilkar ["karakter"] [i] ["sp"]);
        hp[i] = Convert.ToInt32 (ambildataDetilkar ["karakter"] [i] ["hp"]);
        atkpoin[i] = Convert.ToInt32 (ambildataDetilkar ["karakter"] [i] ["atkpoin"]);
        atkspd[i] = Convert.ToInt32 (ambildataDetilkar ["karakter"] [i] ["atkspd"]);
        spd[i] = Convert.ToInt32 (ambildataDetilkar ["karakter"] [i] ["spd"]);
        wktprod[i] = Convert.ToInt32 (ambildataDetilkar ["karakter"] [i] ["wktprod"]);
        hargalvlup[i] = Convert.ToInt32 (ambildataDetilkar ["karakter"] [i] ["hargalvlup"]);
        btnket[i] = ambildataDetilkar ["karakter"] [i] ["lokasibtnKeterangan"].ToString();
        btngame[i] = ambildataDetilkar ["karakter"] [i] ["lokasibtnGame"].ToString();
        karakter[i] = ambildataDetilkar ["karakter"] [i] ["lokasiKarakter"].ToString();
        profpict[i] = ambildataDetilkar ["karakter"] [i] ["profpict"].ToString();
    }
    //bikinjson baru dan pasang
    if (!System.IO.File.Exists (Application.persistentDataPath + "/detilkarakter1.json")) {
        if (!System.IO.File.Exists (Application.persistentDataPath)) {
```

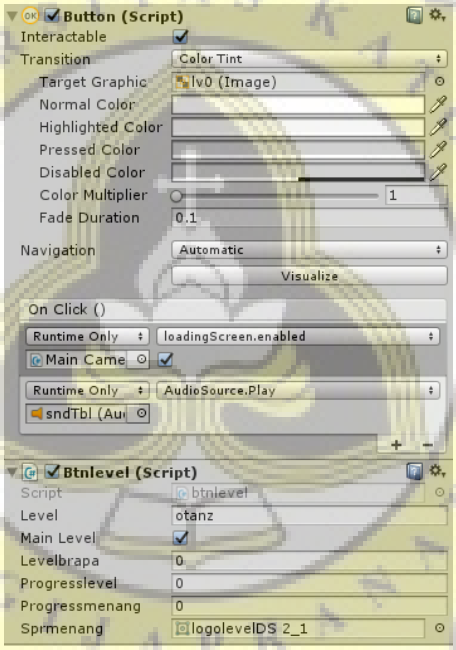
```
            System.IO.Directory.CreateDirectory (Application.persistentDataPath);
        }
    }
    for (i = 0; i < 5; i++) {
        ambildataDetilkar ["karakter"] [i] ["id"] = id [i];
        ambildataDetilkar ["karakter"] [i] ["namakar"] = namakar [i];
        ambildataDetilkar ["karakter"] [i] ["penjelasan"] = penjelasan [i];
        ambildataDetilkar ["karakter"] [i] ["tipe"] = tipe [i];
        ambildataDetilkar ["karakter"] [i] ["level"] = level [i];
        ambildataDetilkar ["karakter"] [i] ["sp"] = sp [i];
        ambildataDetilkar ["karakter"] [i] ["hp"] = hp [i];
        ambildataDetilkar ["karakter"] [i] ["atkpoin"] = atkpoin [i];
        ambildataDetilkar ["karakter"] [i] ["atkspd"] = atkspd [i];
        ambildataDetilkar ["karakter"] [i] ["spd"] = spd [i];
        ambildataDetilkar ["karakter"] [i] ["wktprod"] = wktprod [i];
        ambildataDetilkar ["karakter"] [i] ["hargalvlup"] = hargalvlup [i];
        ambildataDetilkar ["karakter"] [i] ["lokasibtnKeterangan"] = btnket [i];
        ambildataDetilkar ["karakter"] [i] ["lokasibtnGame"] = btngame [i];
        ambildataDetilkar ["karakter"] [i] ["lokasiKarakter"] = karakter [i];
        ambildataDetilkar ["karakter"] [i] ["profpict"] = profpict [i];
    }
    string tulis = Newtonsoft.Json.JsonConvert.SerializeObject (ambildataDetilkar, Newtonsoft.Jso
n.Formatting.Indented);
    File.WriteAllText (Application.persistentDataPath + "/detilkarakter1.json", tulis);

}

void karset(){
    jsonstringKarset = Resources.Load<TextAsset> ("db/karakterset").text;
    ambildataKarset = JObject.Parse (jsonstringKarset);
    int[] idslot;    idslot = new int[5];
    int[] idkar;    idkar = new int[5];
    int i;
    //ambil data dari resource
    for (i = 0; i < 5; i++) {
        idslot [i] = Convert.ToInt32 (ambildataKarset ["karakterset"] [i] ["idslot"]);
        idkar [i] = Convert.ToInt32 (ambildataKarset ["karakterset"] [i] ["idkar"]);
    }
    //bikin json baru dan pasang
    if (!System.IO.File.Exists (Application.persistentDataPath + "/karakterset1.json")) {
        if (!System.IO.File.Exists (Application.persistentDataPath)) {
            System.IO.Directory.CreateDirectory (Application.persistentDataPath);
        }
    }
    GameObject.Find ("TextSampleDuit").GetComponent<Text> ().text = "exist karakterset1";

    for (i = 0; i < 5; i++) {
        ambildataKarset ["karakterset"] [i] ["idslot"] = idslot [i];
        ambildataKarset ["karakterset"] [i] ["idkar"] = idkar [i];
    }
    string tulis = Newtonsoft.Json.JsonConvert.SerializeObject (ambildataKarset, Newtonsoft.Json
.Formatting.Indented);
    File.WriteAllText (Application.persistentDataPath + "/karakterset1.json", tulis);
}

void duititem(){
    //Ambil json dari resource dulu
    jsonstringDuititem = Resources.Load<TextAsset> ("db/duititems").text;
    ambildataDuititem = JObject.Parse (jsonstringDuititem);
    int jmlpetir = Convert.ToInt32 (ambildataDuititem ["items"] [0] ["dimiliki"]);
    int jmljam = Convert.ToInt32 (ambildataDuititem ["items"] [1] ["dimiliki"]);
    int jmlbase = Convert.ToInt32 (ambildataDuititem ["items"] [2] ["dimiliki"]);
    int jmlduit = Convert.ToInt32 (ambildataDuititem ["duit"]);
    int levelterbuka = Convert.ToInt32 (ambildataDuititem ["progresslevel"][0] ["levelterbuka"]);
```

```
        int statusmenang = Convert.ToInt32 (ambildataDuititem ["progresslevel"][0] ["statusmenang"]);
        //bentuk path baru dan cek apakah exist atau ga
        if (!System.IO.File.Exists (Application.persistentDataPath + "/duititem1.json")) {
            if (!System.IO.File.Exists (Application.persistentDataPath)) {
                System.IO.Directory.CreateDirectory (Application.persistentDataPath);
            }
        }
        //susun variable dari yg diambil tadi dan tulis
        ambildataDuititem ["items"] [0] ["dimiliki"] = jmlpetir;
        ambildataDuititem ["items"] [1] ["dimiliki"] = jmljam;
        ambildataDuititem ["items"] [2] ["dimiliki"] = jmlbase;
        ambildataDuititem ["duit"] = jmlduit;
        ambildataDuititem ["progresslevel"][0] ["levelterbuka"] = levelterbuka;
        ambildataDuititem ["progresslevel"][0] ["statusmenang"] = statusmenang;
        string tulis = Newtonsoft.Json.JsonConvert.SerializeObject (ambildataDuititem, Newtonsoft.Js
on.Formatting.Indented);
        File.WriteAllText (Application.persistentDataPath + "/duititem1.json", tulis);
    }

    void koleksi(){
        jsonstringKoleksi = Resources.Load<TextAsset> ("db/koleksikar").text;
        ambildataKoleksi = JObject.Parse (jsonstringKoleksi);
        int[] idkol;    idkol = new int[5];
        int[] idkar;    idkar = new int[5];
        int i;
        //ambil data dari resource
        for (i = 0; i < 5; i++) {
            idkol [i] = Convert.ToInt32 (ambildataKoleksi ["koleksi"] [i] ["idkol"]);
            idkar [i] = Convert.ToInt32 (ambildataKoleksi ["koleksi"] [i] ["idkar"]);
        }
        //bikin json baru dan pasang
        if (!System.IO.File.Exists (Application.persistentDataPath + "/koleksikar1.json")) {
            if (!System.IO.File.Exists (Application.persistentDataPath)) {
                System.IO.Directory.CreateDirectory (Application.persistentDataPath);
            }
        }
        GameObject.Find ("TextSampleDuit").GetComponent<Text> ().text = "exist koleksikar1";

        for (i = 0; i < 5; i++) {
            ambildataKoleksi ["koleksi"] [i] ["idkol"] = idkol [i];
            ambildataKoleksi ["koleksi"] [i] ["idkar"] = idkar [i];
        }
        string tulis = Newtonsoft.Json.JsonConvert.SerializeObject (jsonstringKoleksi, Newtonsoft.Jso
n.Formatting.Indented);
        File.WriteAllText (Application.persistentDataPath + "/koleksikar1.json", tulis);
    }

    void setexist(){
        PlayerPrefs.SetInt ("exist", 1);
    }
}
```

Selanjutnya untuk perpindahan tiap scene terdapat satu *script* yang dapat digunakan oleh semua tombol yang berfungsi untuk perpindahan *scene*. Pada *scene* ini terdapat tombol main yang menggunakan *script* btnlvl.cs. Pada btnlvl.cs dibuat untuk membedakan mana yang merupakan tombol yang menuju ke suatu level *game* maupun tidak. Jadi digunakan variabel *public Boolean mainLevel* yang dapat

diisi menggunakan pemrograman visual dari Unity Editor seperti pada gambar 4.52. Terdapat juga variabel publik *string* yang dapat diisi dengan tujuan scene berikutnya.

```
// Update is called once per frame
void Update () {
    if (ceksudahkah < 2) {
        if (mainLevel) {
            progresslevel = GameObject.Find ("generalsetmap").GetComponent<generalsetmap> ().levelbuka;
            progressmenang = GameObject.Find ("generalsetmap").GetComponent<generalsetmap> ().levelmenang;
            statuslvl (levelbrapa, progressmenang);
            ceksudahkah++;
        }
    }
}
```



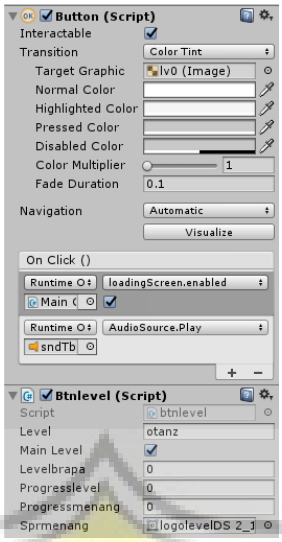
Gambar 4.52. Pemrograman visual tombol

4.4.3 Scene Misi

Pada scene misi, terdapat beberapa *script* yang mengatur jalannya *scene*. Pertama terdapat *script* generalsetmap.cs yang memiliki tugas untuk menampung variabel yang akan digunakan oleh objek lain. Variabel tersebut antara lain adalah variabel *levelbuka, levelmenang, LS, dan swarapindah.* Variabel levelbuka dan levelmenang mendapatkan nilai dari duititem1.json.

```
public class generalsetmap : MonoBehaviour {
    public int levelbuka;
    public int levelmenang;
    public GameObject LS;
    public AudioSource swarapindah;

    // Use this for initialization
    void Start () {
        string jsonstring = System.IO.File.ReadAllText (Application.persistentDataPath + "/duititem1.js
on");
        JObject ambildata = JObject.Parse (jsonstring);
        levelbuka = Convert.ToInt32 (ambildata ["progresslevel"] [0] ["levelterbuka"]);
        levelmenang = Convert.ToInt32 (ambildata ["progresslevel"] [0] ["statusmenang"]);
    }

    // Update is called once per frame
    void Update () {
        btnesc ();
    }

    void btnesc(){
        if (Input.GetKey ("escape")) {
            LS.GetComponent<loadingScreen> ().enabled = true;
            swarapindah.Play ();
            LS.GetComponent<loadingScreen> ().level = "start_page";
        }
    }
}
```

Pada scene ini terdapat objek-objek tombol yang menuju ke misi. Tombol-tombol ini akan memanfaatkan pemrograman visual pada Unity Editor dan untuk pembanding nilainya mengambil nilai-nilai dari variabel-variable publik pada generalsetmap.cs. Pada pemrograman visual Unity Editor seperti yang ditunjukkan pada gambar 4.53, pada btnlevel beri tanda cek pada variabel *mainLevel* dan diisi sesuai level dari tombol. Dengan *script* dan setting seperti gambar akan mendeteksi tombol-tombol level aktif sesuai perkembangan level yang telah dimenangkan.

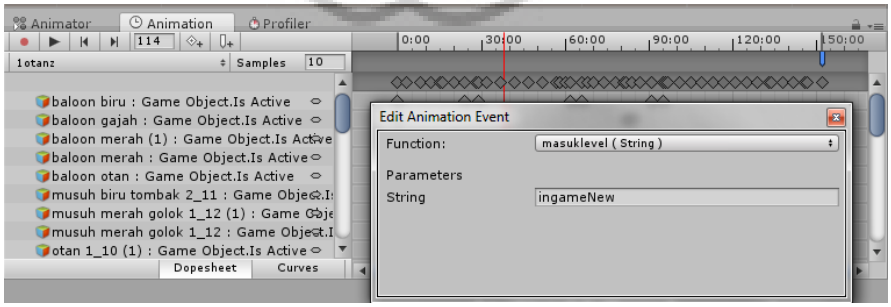


Gambar 4.53 Pemrograman visual tombol level.

Pada scene ini juga terdapat tombol tutorial yang akan mengaktifkan paneltutorial yang akan memberikan tutorial singkat penggunaan umum tombol pada scene misi.

4.4.4 Scene film dan level in-*game*

Setelah memilih level yang aktif, pemain akan masuk pada scene film yang berisi cerita. Pada scene ini ditampilkan film berdurasi pendek menceritakan asal muasal bagaimana para satwa bergabung dengan Wana Warrior. Pada scene ini hanya ada satu *script* yang berisi perintah untuk masuk ke scene level yang dituju. Perintah tersebut disisipkan pada animation event pada Animation seperti yang ditunjukkan gambar 4.54..



Gambar 4.54 Animation pada scene cerita

Selanjutnya pemain akan masuk pada scene in-*game* yang merupakan *gameplay* dari *"Wana Warrior"*. Pada scene ini secara umum diatur oleh *script* generalsetI*game*.cs. *Script* generalsetI*game*.cs berisi seluruh pengaturan dalam *game*.

Pada rutin Update() terdapat beberapa program yang terus berjalan pada tiap skala waktu. Variabel startcount digunakan untuk memberi jeda di awal untuk animasi pada awal *game*. Setelah efek selesai maka rutin-rutin umum dijalankan. Rutin-rutin tersebut antara lain adalah untuk menambah dan menampilkan varabel sp yang akan digunakan untuk Summon Point pada, untuk menghitung waktu *game*, serta untuk mengecek apakah *game* dinyatakan selesai atau belum. Pada variabel waktugame ditambahkan dengan *Time.deltaTime*. *DeltaTime* adalah jarak waktu antara berjalannya fungsi Update(). Terdapat juga *Time.timeScale* yang memiliki fungsi mengontrol waktu berjalannya rutin-rutin pada *game*.

```
   void Update () {
      startcount = startcount - Time.deltaTime;
      if (startcount <= 3.5 && startcount > 1.8) {    //efek getsetgo
         GameObject.Find ("TextReadysetgo").GetComponent<Text> ().text = "Ready";
      } else if (startcount <= 1.8 && startcount > 0.3) {
         GameObject.Find ("TextReadysetgo").GetComponent<Text> ().text = "Fight";
      } else if (startcount <= 0.3 && startcount > 0.05) {
         Destroy (GameObject.Find ("PanelStart"));
         GameObject.Find ("Main Camera").GetComponent<camdrags> ().enabled = true;
      } else if (startcount <= 0) {
//baru mulai rutin utamanya
          textSP = GameObject.Find ("TextSP").GetComponent<Text> ();
         textkembangSP = GameObject.Find ("TextKembangSP").GetComponent<Text> ();
         if (Time.timeScale <= 0 || sp >= spmax || gamefinished == true)
            sp = sp + 0;
         else if (Time.timeScale > 0 && sp <= spmax)
            sp = sp + perkembanganspdsp;

         textspp = String.Format ("{0:#}", sp);
         textSP.text = (textspp + "/" + spmax.ToString ());
         textkembangSP.text = untuknaikinspdsp.ToString ();
         if(!gamefinished)
            waktugame = waktugame + Time.deltaTime;
         if (isGamefinished) {
            waktugame = waktugame + 0;
            finished (waktugame, apakalakon);
            GameObject.Find ("Main Camera").GetComponent<camdrags> ().enabled = false;
```

```
        }
    }
}
```

Selanjutnya terdapat fungsi publik paused() yang memiliki tugas untuk mengatur timeScale dan *camera drag* sehingga *game* berhenti saat dipause. Fungsi paused() diaktifkan dengan pemrograman visual, gambar 4.55, pada tombol pause yang juga mengaktifkan panel pause yang akan mengaktifkan dan menon-aktifkan suara pada *game*. Tombol resume yang berfungsi menjalankan kembali *game* dan menonaktifkan panel pause.

```
public void pause(bool pausecondition){
    if (pausecondition) {
        Time.timeScale = 0;
        GameObject.Find ("Main Camera").GetComponent<camdrags> ().enabled = false;
    } else if (!pausecondition) {
        Time.timeScale = 1;
        GameObject.Find ("Main Camera").GetComponent<camdrags> ().enabled = true;
    }
}
```

Gambar 4.55 Pemrograman visual pada tombol pause

Selanjutnya terdapat fungsi publik finished() yang menyatakan *game* telah selesai dan memberikan laporan berapa lama pemain menyelesaikan permainannya. Permainan dinyatakan selesai apabila salah satu base kehabisan *hitpoint*. Pada objek base akan mengirimkan variabel boolean is*Game*finished dan apakalakon yang akan memicu if pada fungsi Update(). Update() akan mengirim sinyal pada fungsi

finished() yang akan mengaktifkan objek panel memroses variabel yang dikirimkan dan dijadikan tampilan berupa teks. Pada fungsi finished() dilakukan kalibrasi nilai dari nilai dari variabel waktu*game* yang berguna memberikan tampilan menit dan detik yang sesuai. Penggunaan string format digunakan untuk penulisan string yang sesuai dengan waktu yang biasa tercatat.

```
public void finished (float waktumain, bool musuhlakon){
    //berhentiin broadcast dari update
    isGamefinished = false;
    gamefinished = true;
    panelfinish.SetActive(true);
    Destroy (GameObject.Find ("spawnermusuh"));
    float detik = waktumain % 60;//sisa bagi angka buat dapetin detik - mod
    float menit = waktumain / 60;//hasil bagi buat dapetin menit - div

    if (menit < 1)//penyesuaian angka apabila dibawah satu menit
        menit = 0;
    string dtk = String.Format ("{00:#}", detik);
    string men = String.Format ("{00:#}", menit);
    //penyesuaian digit
    if (detik <  9.5)
        panelfinish.transform.Find ("TextHasilWaktu").GetChild (0).GetComponent<Text> ().text = "
0" + dtk;
    else
        panelfinish.transform.Find ("TextHasilWaktu").GetChild (0).GetComponent<Text> ().text = d
tk;
    if (dtk == "" || detik > 59.5)
        panelfinish.transform.Find ("TextHasilWaktu").GetChild (0).GetComponent<Text> ().text = "
00";

    if (menit < 9.5)
        panelfinish.transform.Find ("TextHasilWaktu").GetChild (1).GetComponent<Text> ().text = "
0" + men;
    else
        panelfinish.transform.Find ("TextHasilWaktu").GetChild (1).GetComponent<Text> ().text =
men;
    if (men == "" || men == "60")
        panelfinish.transform.Find ("TextHasilWaktu").GetChild (1).GetComponent<Text> ().text = "
00";

    PlayerPrefs.SetInt ("jmlkoinMin", jmlkoinMin);
    PlayerPrefs.SetInt ("jmlkoinMax", jmlkoinMax);

    if (musuhlakon) {
        panelfinish.transform.Find ("ButtonMenang").gameObject.SetActive (true);
        panelfinish.transform.Find ("ButtonKalah").gameObject.SetActive (false);
        nambahprogress ();
    } else {
        panelfinish.transform.Find ("ButtonMenang").gameObject.SetActive (false);
        panelfinish.transform.Find ("ButtonKalah").gameObject.SetActive (true);
    }
}

public void kurangispuntukspawn(int spkar){
    sp -= spkar;
```

```
}
```

Pada scene selanjutnya, pemain akan diberikan hadiah berupa koin atau item. Pada tiap level akan diberikan jumlah hadiah koin yang berbeda. Untuk itu, dilakukan penyesuaian tiap level agar berbeda hadiah yang diperoleh di setiap level yang dimenangkan. Agar perintah untuk penyesuaian hadiah dapat dilakukan pada scene selanjutnya, maka variabel jmlkoinMin dan jmlkoinMax disimpan pada playerprefs.

Selanjutnya apabila pemain dinyatakan menang atau kalah dilakukan dengan mengaktifkan tombol ButtonMenang dan menonaktifkan tombol ButtonKalah. Hal ini berlaku sebaliknya apabila pemain dinyatakan kalah dalam permainan. Namun apabila pemain memenangkan permainan akan dimasukkan pada fungsi nambahprogress() yang akan mengecek apakah progress akan ditambah atau tidak sesuai level yang dimainkan oleh pemain.

Pada fungsi nambahprogress() dilakukan pengecekan pada data json apakah pemain sudah pernah memenangkan level yang dimainkan atau belum. Apabila pemain belum pernah menyelesaikan level yang dimainkan maka dilakukan manipulasi data pada json. Data json akan diambil untuk menentukan jumlah variabel pada progresslevel dan progressmenang. Setelah diambil, dilakukakn penambahan satu angka pada json dan data json disimpan pada json yang diambil sebelumnya. Hal ini akan berdampak pada jumlah terbukanya level yang ada pada scene misi. Selanjutnya apabila pemain belum memenangkan level sebelumnya akan masuk juga ke fungsi nambah koleksi.

```
void nambahprogress(){
    int progresslevel = Convert.ToInt32 (ambildata ["progresslevel"][0] ["levelterbuka"]);
    int progressmenang = Convert.ToInt32 (ambildata ["progresslevel"][0] ["statusmenang"]);
    if (progressmenang <= inilevelberapa) {
        //nambah untuk progress
        progresslevel++;
```

```
            progressmenang++;
            ambildata ["progresslevel"] [0] ["levelterbuka"] = progresslevel;
            ambildata ["progresslevel"] [0] ["statusmenang"] = progressmenang;
            string tulis = Newtonsoft.Json.JsonConvert.SerializeObject (ambildata, Newtonsoft.Json.For
matting.Indented);
            File.WriteAllText (Application.persistentDataPath + "/duititem1.json", tulis);
            //nambah karakter apabila sebelumnya belum memenangkan misi.
            nambahkoleksi (inilevelberapa);
        }
    }
```

Pada fungsi nambahkoleksi() dimaksudkan agar pemain mendapatkan karakter baru saat menjalankan misi selanjutnya apabila memenangkan level sekarang pada saat ini dan belum memenangkan level yang sekarang sebelumnya. Pada nambah koleksi akan menggunakan data dari dua file json, yaitu koleksikar1.json dan karakterset1.json. Sebelum memroses data json, panjang variabel array yang akan digunnakan disesuaikan dahulu agar panjang array sesuai dengan jumlah data yang akan dimanipilasi. Setelah itu dilakukan looping susuai panjang data pada json untuk mengambil data pada json. Setelah itu data langsung dimasukkan pada variabel json masing-masing lagi dan ditambahkan 1 pasang variabel json di masing-masing json. Perbedaan terletak apabila pemain memenangkan permainan pada level 5 untuk pertama kali. Kalau level sebelumnya hanya diberi satu karakter, namun pada level 5 akan langsung diberikan 2 karakter. Setelah semua variabel json terisi, dilakukan penyimpanan data pada koleksikar1.json dan karakterset1.json.

```
    void nambahkoleksi(int id){
        string jsonstrkol = System.IO.File.ReadAllText (Application.persistentDataPath + "/koleksikar1.
json");
        JObject ambildatakol = JObject.Parse (jsonstrkol);
        string jsonstrkarset = System.IO.File.ReadAllText (Application.persistentDataPath + "/karakter
set1.json");
        JObject ambildatakarset = JObject.Parse (jsonstrkarset);
        idka = new int[id + 1];
        idko = new int[id + 1];
        idks = new int[id + 1];
        idkr = new int[id + 1];
        for (int i = 0; i < id + 1; i++) {
            idko [i] = Convert.ToInt32 (ambildatakol ["koleksi"] [i] ["idkol"].ToString ());
            idka [i] = Convert.ToInt32 (ambildatakol ["koleksi"] [i] ["idkar"].ToString ());
            idks [i] = Convert.ToInt32 (ambildatakarset ["karakterset"] [i] ["idslot"].ToString ());
            idkr [i] = Convert.ToInt32 (ambildatakarset ["karakterset"] [i] ["idkar"].ToString ());
```

```
        }

        //simpan baru tambah 1 data pada level 1-5
        int j;
        for (j = 0; j < id + 1; j++) {
            ambildatakol ["koleksi"] [j] ["idkol"] = idko [j];
            ambildatakol ["koleksi"] [j] ["idkar"] = idka [j];
            ambildatakarset ["karakterset"] [j] ["idslot"] = idks [j];
            ambildatakarset ["karakterset"] [j] ["idkar"] = idkr [j];
        }
        ambildatakol ["koleksi"] [j]["idkol"] = idko [j-1]+1;
        ambildatakol ["koleksi"] [j]["idkar"] = idka [j-1]+1;
        ambildatakarset ["karakterset"] [j] ["idslot"] = idks [j - 1] + 1;
        ambildatakarset ["karakterset"] [j] ["idkar"] = idkr [j - 1] + 1;
        if (id == 4) {
            ambildatakol ["koleksi"] [j + 1] ["idkol"] = idko [j - 1] + 2;
            ambildatakol ["koleksi"] [j + 1] ["idkar"] = idka [j - 1] + 2;
        }
        string tulis1 = Newtonsoft.Json.JsonConvert.SerializeObject (ambildatakol, Newtonsoft.Json.F
ormatting.Indented);
        string tulis2 = Newtonsoft.Json.JsonConvert.SerializeObject (ambildatakarset, Newtonsoft.Jso
n.Formatting.Indented);
        File.WriteAllText (Application.persistentDataPath + "/koleksikar1.json", tulis1);
        File.WriteAllText (Application.persistentDataPath + "/karakterset1.json", tulis2);
    }
    void quitkeluar(int a){
        Time.timeScale = a;
    }
}
```

Terdapat 2 fungsi lain yang memiliki satu perintah namun juga penting, yaitu fungsi kurangispuntukspawn() dan quitkeluar(). Pada fungsi kurangispuntukspawn() berguna untuk mengurangi jumlah variabel sp. Fungsi ini dipicu oleh tombol karakter pada *game*. Fungsi quitkeluar() berfungsi untuk mengembalikan timescale setela tombol keluar pada panel pause ditekan agar timescale kembali normal dan tombol keluar dapat menjalankan tugasnya.

Untuk mengatur karakter terdapat *script* yang identik antara karakter *"Wana Warrior"* dan karakter Helm Kuning. *Script* tersebut adalah *script* untukLakon.cs dan untukMusuh.cs. Kedua *script* tersebut digunakan untuk mengatur objek-objek yang merupakan bagian dari karakter lawan maupun musuh. Terdapat objek-objek yang memiliki properti collider2d yang digunakan sebagai pembatas maupun sebagai pemicu saja.

Pada gambar 4.56 merupakan karakter Otan dan terdapat objek-objek yang dimiliki. Semua karakter “Wana Warrior” memiliki objek-objek yang sama. Yang membedakan antara karakter yang menyerang pada jarak dekat dan jarak jauh adalah objek serangan yang bergerak maupun tidak pada animasi.

Gambar 4.56 Properties pada objek karakter Otan

Pada *script* untukLakon.cs ,variabel dan berisi fungsi-fungsi yang yang digunakan sebagai perintah untuk karakter “Wana Warrior”. Untuk pendeklarasian dari beberapa variabel tidak dimasukkan pada fungsi Start(), namun dimasukkan pada fungsi Awake(). Fungsi Awake() digunakan karena fungsi ini akan diakses paling awal. Sehingga pada saat program memulai rutin pada Update(), semua variabel sudah terisi dan dapat digunakan.

```
void Awake(){
    jarakSerang = transform.Find ("triggerSerang");
    anim = GetComponent<Animator>();
    jsonstr=System.IO.File.ReadAllText(Application.persistentDataPath+"/detilkarakter1.json");
    ambildata = JObject.Parse (jsonstr);
    sp = Convert.ToInt32 (ambildata ["karakter"] [idkar] ["sp"].ToString ());
    hp = Convert.ToInt32 (ambildata ["karakter"] [idkar] ["hp"].ToString ());
    atkpoin = Convert.ToInt32 (ambildata ["karakter"] [idkar] ["atkpoin"].ToString ());
    animatkspd = Convert.ToInt32 (ambildata ["karakter"] [idkar] ["atkspd"].ToString ());
}
```

Selanjutnya pada Update() hanya terdapat perintah untuk pemicu apakah akan menyerang atau tidak. Variabel boolean serang akan mendapatkan nilai dari Physics2d.Linecast. Physics2d.Linecast adalah bagian dari class Physics2d yang berfungsi sebagai sinyal yang sebagai penanda apakah objek menyentuh sesuatu yang diinginkan atau tidak. Jarak pada Linecast ditentukan dengan jarak objek

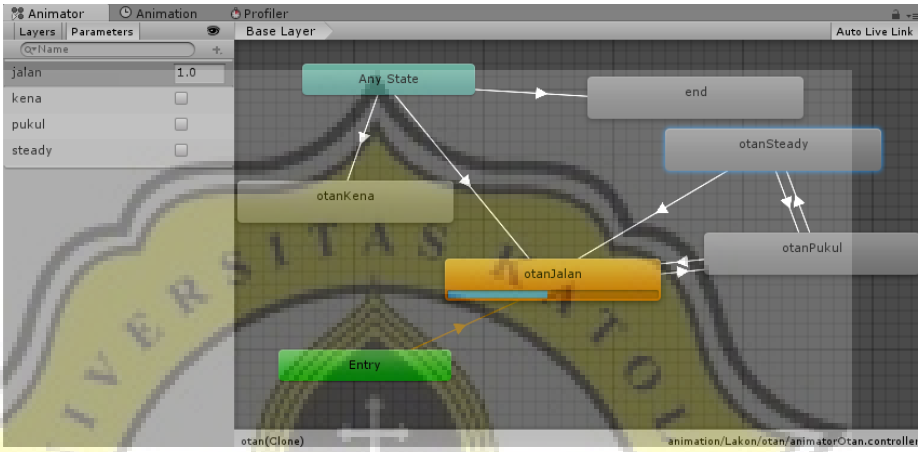
dengan objek triggerserangan. Pada jarak tersebut hanya akan mengecek apakah diantara jarak tersebut bersentuhan dengan objek yang ada pada layer musyuh. Selain itu terdapat if yang akan mengecek apakah permainan dinyatakan selesai atau tidak.

```
void Update () {
    //ngecekin linecastnya nyentuh layer musyuh apa ngga
    serang = Physics2D.Linecast (transform.position, jarakSerang.position, 1 << LayerMask.Name
ToLayer ("musyuh"));
    if (GameObject.Find ("generalSetting").GetComponent<generalsetIngame> ().gamefinished =
= true)
        gamefinished();
    //setting moveForce di editor speednya
}
```

Fungsi FixedUpdate() pada *script* ini digunakan untuk mengecek apakah karakter dalam posisi menyerang atau tidak dan memberikan perintah pada animator, gambar 4.57, untuk menyesuaikan animasi yang digunakan serta memberikan gaya gerak pada karakter dengan posisi yang telah ditentukan. Pada kondisi menyerang, kondisi animasi dan transisinya akan dimanipulasi sehingga status animasi yang ditampilkan sesuai dan karakter berada dalam kondisi tidak bergerak maju. Selanjutnya pada saat karakter pada posisi yang tidak menyerang, karakter akan bergerak maju dengan status animasi yang sesuai. Selain itu pada fungsi *game*finished() juga memiliki perintah yang hamper sama yaitu karakter akan berada dalam kondisi idle dan tidak memiliki gaya gerak maju sehingga karakter tetap pada tempat. Fungsi ini mendapat perintah dari Update() yang mengecek pada objek generalSetting.

```
void FixedUpdate(){
    //dan pabila serang menjadi true
    if (serang) {
        //setting paramater animator;
        anim.SetFloat ("jalan", 0);
        anim.SetBool ("pukul", true);
        GetComponent<Rigidbody2D> ().velocity = new Vector2 (0, 0);
    } else {
            anim.SetFloat ("jalan", 1);
            anim.SetBool ("pukul", false);
            anim.SetBool ("steady", false);
```

```
        //memberikan gaya gerak
        if (statustops == 0)
            GetComponent<Rigidbody2D> ().AddForce (Vector2.right * moveForce);
        else if (statustops == 1)
            GetComponent<Rigidbody2D> ().velocity = new Vector2 (0, 0);
            //kecepatan maksimal
            if (GetComponent<Rigidbody2D> ().velocity.x > maxSpeed)
                GetComponent<Rigidbody2D> ().velocity = new Vector2 (maxSpeed, 0);
    }
    speedo = GetComponent<Rigidbody2D> ().velocity.x;
}
```



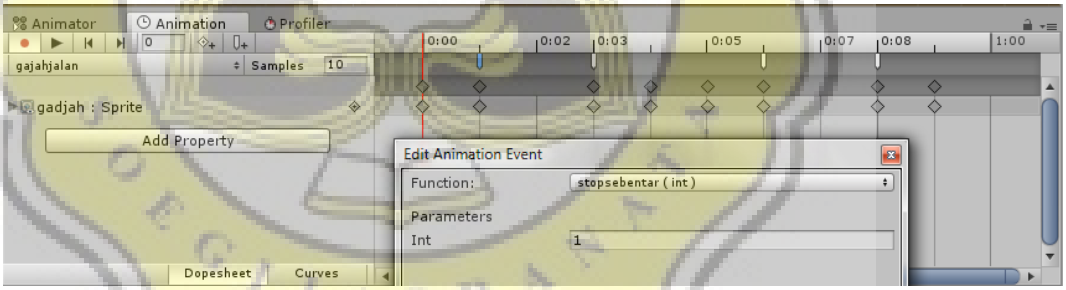
Gambar 4.57 Pemrogaman visual animator pada karakter Otan

Pada fungsi OnTriggerEnter2D() merupakan fungsi dari Unity yang berfungsi untuk pemicu apakah collider2d menyentuh siatu objek. Pada *script* ini, collider2d karakter akan mengecek apakah disentuh oleh collider serangan musuh atau tidak. Apabila menyentuh objek serangan akan dilakukan pengambilan variabel attack dari serangan yang menyentuh collider2d pada karakter dan selanjutnya akan masuk pada fungsi kenaserang(). Fungsi kenaserang() akan menghitung pengurangan variabel hp dengan variabel attack dari objek serangan musuh yang menyentuh collider2d karakter. Apabila hp dalam posisi kurang dari 0 makan karakter akan hancur.

```
public void gamefinished(){
    anim.SetFloat ("jalan", 0);
    anim.SetBool ("pukul", false);
    anim.SetBool ("steady", true);
    moveForce=0f;
}
```

```
    void OnTriggerEnter2D(Collider2D coll){
        if (coll.tag == "seranganMusuh") {
            int atk = coll.gameObject.GetComponent<untukseranganMusuh> ().attack;
            kenaserang (atk);
        }
    }

    void kenaserang(int atkmusuh){
        hp -= atkmusuh;
        if (hp <= 0)
            Destroy (gameObject);
    }

    public void stopsebentar(int statstp){
        statustops = statstp;
    }

    public void suaratt(){
        suaraaatk.Play ();
    }
}
```

Untuk memberikan efek pada beberapa karakter, maka diberikan fungsi publik stopsebentar(). Variabel ini membuat karakter tidak mendapatkan gaya gerak sesaat. Fungsi ini dipicu oleh animation event yang diberikan pada animaton, pada gambar 4.58. Selanjutnya ada fungsi suaraatt() yang berfungsi memunculkan suata saat karakter menyerang.



Gambar 4.58 Animation pada karakter Otan

Untuk mengatur karakter Helm Kuning memiliki konfigurasi yang identik dengan karakter "Wana Warrior". Namun terdapat dua perbedaan, yaitu untuk variabel-variabel yang berhubungan dengan skill diisi melalui Unity editor dengan pemrograman visual dan perbedaan pada fungsi bangbang(). Fungsi bangbang() berfungsi untuk spawn objek peluru untuk karakter yang serangannya menggunakan senapan. Fungsi ini dipicu oleh animation event serangan seperti karakter-karakter lainnya.

```
    void bangbang(){
        float spd=30f;
        bool posision = false;
        if (posision) {
            Rigidbody2D pelur = Instantiate (pelor, gameObject.transform.FindChild("spawnerpeluru").tr
ansform.position, Quaternion.Euler (new Vector3 (0, 0, 180)))as Rigidbody2D;
            pelur.velocity = new Vector2 (spd, 0);
            pelur.transform.parent = gameObject.transform;
        } else {
            Rigidbody2D pelur = Instantiate (pelor, gameObject.transform.FindChild("spawnerpeluru").tr
ansform.position, Quaternion.Euler (new Vector3 (0, 0, 0)))as Rigidbody2D;
            pelur.velocity = new Vector2 (-spd, 0);
            pelur.transform.parent = gameObject.transform;
        }
    }

    public void suaratt(){
        suaraaatk.Play ();
    }
}
```

Selanjutnya adalah objek btnSP yang memiliki *script* untukbtnsp.cs yang memiliki tugas mengatur kondisi pada summon point dan tampilan pada objek btnSP. Pada fungsi ini, data jumlah perkembangan dari summon point akan didapatkan dari json duititem1.json. Untuk mempercepat bertambahnya summon point, maka pemain harus menekan tombol btnSP saat tombol dinyatakan aktif. Terdapat aturan yang diberlakukan, apabila pemain ingin mempercepat bertambahnya summon point, maka harus menggunakan sejumlah summon point. Apabila tombol terpicu, maka pada fungsi kliked() akan memberikan perintah pada objek generalSetting untuk melakukan aksi pada *script* genneralsetin*game*().

```
    public void kliked(){
        GameObject.Find ("generalSetting").GetComponent<generalsetIngame> ().sp -
= GameObject.Find ("generalSetting").GetComponent<generalsetIngame> ().untuknaikinspdsp;
        GameObject.Find ("generalSetting").GetComponent<generalsetIngame> ().perkembanganspd
sp += nilaipengembanganspdSP;
        GameObject.Find ("generalSetting").GetComponent<generalsetIngame> ().untuknaikinspdsp
+= nilaikembangmax;
        GameObject.Find ("generalSetting").GetComponent<generalsetIngame> ().spmax += nilaispm
ax;
    }
```

Untuk mengakomodir tombol-tombol karakter yang akan memicu keluarnya karakter digunakan *script* buatbuttoning*game*1.cs yang akan menata tombol pada *PanelKarset*. Tombol-tombol karakter yang dimuncukan adalah tombol-tombol dari karakter yang masuk dalam karakter set. Pengisian objek pada

panel dilakukan dengan penyesuaian-penyesuaian ukuran agar besarnya tombol bisa selalu sama pada semua ukuran layar *device.* Prosesnya adalah dengan menyiapkan *template* untuk pemasangan tombol. Besaran template dibuat menyesuaikan ketinggian dari panel yang dibuat. Setelah itu lebar tombol akan menyesuaikan dari rasio tinggi tombol dengan panel.

```
    void Start () {
        RectTransform rowRectTransform = btnkotak.GetComponent<RectTransform>();
        RectTransform containerRectTransform = gameObject.GetComponent<RectTransform>();

        float height = containerRectTransform.rect.height;
        float ratio = height / rowRectTransform.rect.height;//tinggi menjadi acuan rasio karena tinggi en
titas yg tidak diubah script
        float width = rowRectTransform.rect.width * ratio;

        float scrollWidth = width * 5;
        containerRectTransform.offsetMin = new Vector2 (-
scrollWidth / 2, containerRectTransform.offsetMin.y);
        containerRectTransform.offsetMax = new Vector2 (scrollWidth / 2, containerRectTransform.off
setMax.y);

        jsonstr1=System.IO.File.ReadAllText(Application.persistentDataPath+"/karakterset1.json");
        ambildata1 = JObject.Parse (jsonstr1);
        jsonstr2=System.IO.File.ReadAllText(Application.persistentDataPath+"/detilkarakter1.json");
        ambildata2 = JObject.Parse (jsonstr2);

        for (int i = 0; i < 5; i++)
        {
            int aidi = Convert.ToInt32(ambildata1 ["karakterset"] [i] ["idkar"].ToString());
            //create new items
            itemPrefab[i]=Resources.Load(ambildata2 ["karakter"] [aidi] ["lokasibtnGame"].ToString())as
 GameObject;
            GameObject newItem = Instantiate(itemPrefab[i]) as GameObject;
            newItem.transform.parent = gameObject.transform;

            //move and size the new item
            RectTransform rectTransform = newItem.GetComponent<RectTransform>();

            float x = -containerRectTransform.rect.width / 2 + width * i-5;
            float y = containerRectTransform.rect.height / 2 - height-5;
            rectTransform.offsetMin = new Vector2(x, y);

            x = rectTransform.offsetMin.x + width+10;
            y = rectTransform.offsetMin.y + height+10;
            rectTransform.offsetMax = new Vector2(x, y);
        }
    }

}
```

Terdapat satu *script* yang digunakan untuk mengatur objek baseHPLakon dan baseHPMusuh yang menjadi visual dari status berapa jumlah hitpoint pada tiap homebase. Dengan membandingkan jumlah *hitpoint* di awal dan hitpoin sekarang

akan mendapatkan panjang visual yang sesuai. Sebelumnya, panjangnya *helathbar* diatur pasti skalanya agar selalu stabil pada semua layar *device*. Seperti proses pembuatan tombol tadi, pembuatan panjang dan lebar awal objek ini juga menggunakan metode yang sama. Namun panjang dari objek ini dibuat dinamis agar dapat menampilkan visual yang sesuai.

```
    void Start () {
        //ambil rect transform dari healthbar
        healthbarRT = gameObject.GetComponent<RectTransform> ();
        skala = Screen.width / 640.0f;
        Debug.Log ("Skala = "+skala);
        panjanghealthbar = healthbarRT.rect.width * skala;
        lebarhealthbar = healthbarRT.rect.height;

        //inisiasi nilai awal untuk health base
        inisialhomebase=homebase.GetComponent<HPhomebase>().hpbase;
    }

    // Update is called once per frame
    void Update () {
        temphp = hphomebase;
        hphomebase = homebase.GetComponent<HPhomebase> ().hpbase;
        if (temphp != hphomebase) {
            persenhp = hphomebase / inisialhomebase;
            x = panjanghealthbar * persenhp;

            if (lakon) {
                healthbarRT.offsetMin = new Vector2 (healthbarRT.offsetMin.x, healthbarRT.offsetMin.y);
                healthbarRT.offsetMax = new Vector2 (x, healthbarRT.offsetMax.y);
            } else if (!lakon) {
                healthbarRT.offsetMin = new Vector2 (-x, healthbarRT.offsetMin.y);
                healthbarRT.offsetMax = new Vector2 (healthbarRT.offsetMax.x, healthbarRT.offsetMax.
y);
            }
        }
    }
}
```

4.4.5 Scene Warrior

Scene Warrior menrupakan scene yang berisi pengaturan pada karakterset, melihatt karakter yang telah didapat, melihat skill karakter serta levelup, dan melihat biografi karakter. Program utama scene ini terdapat 3 objek, yaitu generalset, panel karset dan panel karater. Masing-masing objek memiliki *script* yang memiliki progam untuk tiap objek.

Pada objek generalset terdapat script yang bernama generalsettingPilihkar.cs Pada *script* ini terdapat perintah-perintah yang mengurusi

data junmlah koin sebgai transaksi untuk menaikkan level karakter. Data jumlah koin diambil dari file json duititem1.json. Selanjutnya data jumlah koin disimpan pada variabel jmlduit yang nantinya bisa dimanipilasi oleh script pada objek lain. Tampilan jumlah koin yang dimiliki ditampilkan pada objek TextDuit. *Script* melakukan *broadcast* nilai melalui method FixedUpdate() agar jika ada perubahan dapat dengan cepat ditampilkan. Fungsi btnesc() dibuat untuk memberikan perintah kembali ke scene misi apabila tombol back atau escape ditekan. Pada fungsi gantijsonduit() akan berfungsi sebagai penyimpan data ke file duititem1.json apabila akan berpindah scene. Fungsi ini akan dipicu oleh fungsi btnesc() dan tombol back pada scene.

```
    void Start(){
        RectTransform rowRectTransform = btnkotak.GetComponent<RectTransform> ();
        containerRectTransform = gameObject.GetComponent<RectTransform> ();

        //menentukan ukuran button dengan acuan kontainer. dasar width dan height diukur dari jarak
titik2 rect pada rect transform
        height = containerRectTransform.rect.height;
        float ratio = height / rowRectTransform.rect.height;//tinggi menjadi acuan rasio karena tinggi en
titas yg tidak diubah script
        width = rowRectTransform.rect.width * ratio;

        float scrollWidth = width * 5;
        containerRectTransform.offsetMin = new Vector2 (-
scrollWidth / 2, containerRectTransform.offsetMin.y);
        containerRectTransform.offsetMax = new Vector2 (scrollWidth / 2, containerRectTransform.off
setMax.y);

        datakart (0);
    }

    void datakart (int adaganti){
        string jsonstrr;
        string jsonstr;
        if (adaganti == 1) {
            jsonstrr = null;
            jsonstr = null;
            ambildata1 = null;
            ambildata2 = null;
        }
        string strkount=System.IO.File.ReadAllText(Application.persistentDataPath+"/karakterset1.jso
n");
        JsonData ambildatacount = JsonMapper.ToObject (strkount);
        int kount = Convert.ToInt32 (ambildatacount ["karakterset"] [0].Count.ToString ());
        //ambil data dr karpakai BELUM MASUK KE DETILKARAKTER.JSON
        jsonstr=System.IO.File.ReadAllText(Application.persistentDataPath+"/detilkarakter1.json");
        ambildata2 = JObject.Parse (jsonstr);
        jsonstrr=System.IO.File.ReadAllText(Application.persistentDataPath+"/karakterset1.json");
        ambildata1 = JObject.Parse (jsonstrr);
```

```
        for (int i = 0; i < 5; i++) {
            if (ambildata1 ["karakterset"] [i] ["idkar"].ToString() != "-1") {
                int aidi = Convert.ToInt32 (ambildata1 ["karakterset"] [i] ["idkar"].ToString ());

                //create a new item, name it, and set the parent
                itemPrefab [i] = Resources.Load (ambildata2 ["karakter"] [aidi] ["lokasibtnKeterangan"].To
String ())as GameObject;
                GameObject newItem = Instantiate (itemPrefab [i]) as GameObject;
                newItem.name = "Karset " + i;
                newItem.transform.parent = gameObject.transform;

                //move and size the new item
                RectTransform rectTransform = newItem.GetComponent<RectTransform> ();

                float x = -containerRectTransform.rect.width / 2 + width * i;
                float y = containerRectTransform.rect.height / 2 - height;
                rectTransform.offsetMin = new Vector2 (x, y);

                x = rectTransform.offsetMin.x + width;
                y = rectTransform.offsetMin.y + height;
                rectTransform.offsetMax = new Vector2 (x, y);
            } else
                i = 5;
        }
    }

    public void usebtnpressed(int kode){
        string jsonstr1 = System.IO.File.ReadAllText(Application.persistentDataPath+"/detilkarakter1.j
son");
        ambildata1 = JObject.Parse (jsonstr1);
        string jsonstr2 = System.IO.File.ReadAllText(Application.persistentDataPath+"/karakterset1.js
on");
        ambildata2 = JObject.Parse (jsonstr2);

        for (int i = 0; i < 5; i++) {
            //create a new item, name it, and set the parent
            GameObject itembaru=Instantiate(Resources.Load("prefabChar/btntry")) as GameObject;
            itembaru.name = "Karset ganti" + i;
            itembaru.transform.parent = gameObject.transform;

            //move and size the new item
            RectTransform rectTransform = itembaru.GetComponent<RectTransform> ();

            float x = -containerRectTransform.rect.width / 2 + width * i;
            float y = containerRectTransform.rect.height / 2 - height;
            rectTransform.offsetMin = new Vector2 (x, y);
            x = rectTransform.offsetMin.x + width;
            y = rectTransform.offsetMin.y + height;
            rectTransform.offsetMax = new Vector2 (x, y);
        }
        Button inibutton;
        inibutton = GameObject.Find("Karset ganti0").GetComponent<Button> ();
        inibutton.onClick.AddListener (() => gantikejson (0, kode));
        inibutton = GameObject.Find("Karset ganti1").GetComponent<Button> ();
        inibutton.onClick.AddListener (() => gantikejson (1, kode));
        inibutton = GameObject.Find("Karset ganti2").GetComponent<Button> ();
        inibutton.onClick.AddListener (() => gantikejson (2, kode));
        inibutton = GameObject.Find("Karset ganti3").GetComponent<Button> ();
        inibutton.onClick.AddListener (() => gantikejson (3, kode));
        inibutton = GameObject.Find("Karset ganti4").GetComponent<Button> ();
        inibutton.onClick.AddListener (() => gantikejson (4, kode));

    }
```

```
    void gantikejson(int kodeblok, int idkarakter){
        //cek apakah ada yg sama pada karset
        string jsonstr=System.IO.File.ReadAllText(Application.persistentDataPath+"/karakterset1.json"
);
        JObject parseJ = JObject.Parse (jsonstr);
        JObject ambildata = JObject.Parse (jsonstr);
        bool sama = false;

        //cocokin idslot di json sama kode dr tombol ganti
        for (int i = 0; i < 5; i++) {
            if (ambildata ["karakterset"] [i] ["idslot"].ToString () == kodeblok.ToString()) {
                var karset=parseJ["karakterset"];
                //cek apakah ada yg sama, apabila sama move posisi
                for (int j = 0; j < 5; j++) {
                    if (idkarakter.ToString () == karset [j] ["idkar"].ToString ()) {
                        sama = true;

                        //simpan karset[i] dulu dalam variabel biar ga ilang
                        int tempidkarkarseti=Convert.ToInt32(karset [i] ["idkar"].ToString());
                        //overwrite data di karset[i] dan karset baru
                        karset [i] ["idkar"] = idkarakter;
                        karset [j] ["idkar"] = tempidkarkarseti;
                    }
                }

                if (sama == false)
                    karset [i] ["idkar"] = idkarakter;
                string tulis = Newtonsoft.Json.JsonConvert.SerializeObject (parseJ, Newtonsoft.Json.For
matting.Indented);
                File.WriteAllText (Application.persistentDataPath + "/karakterset1.json", tulis);
                adapergantian ();
            }
        }
    }

    void adapergantian(){
        int a;
        string jsonstr=System.IO.File.ReadAllText(Application.persistentDataPath+"/detilkarakter1.jso
n");
        JsonData ambildata = JsonMapper.ToObject (jsonstr);
        for (a = 0; a <= 4; a++) {
            GameObject tombol = GameObject.Find ("Karset ganti" + a);
            Destroy (tombol);
            GameObject tombolkarset = GameObject.Find ("Karset " + a);
            Destroy (tombolkarset);
        }

        for (a = 0; a < ambildata [0].Count; a++) {
            GameObject tombolkoleksikar = GameObject.Find ("Koleksikar" + a);
            Destroy (tombolkoleksikar);
        }

        #if UNITY_EDITOR_64
        AssetDatabase.Refresh();
        #endif

        datakart (1);
        listKrakterDidapat gantipreviewkar = GameObject.Find ("panelKarakter").GetComponentInChil
dren<listKrakterDidapat> ();
        gantipreviewkar.display ();
    }
}
```

Pada *script* listKarakterPakai.cs terdapat fungsi-fungsi yang mengurusi komunikasi dengan file json, menampilkan set karakter, dan pergantian set karakter. Pada Start() dilakukan pembagian panelKarset menjadi 5 slot yang akan diisioleh tombol karakter. Agar tombol memiliki ukuran dan rasio yang sama, dilakukan penyesuaian ukuran melalui *script*. Hal ini bertujuan agar besarnya stabil pada semua device. Setelah dibuat slot, maka tombol-tombol karakter dimasukan pada tiap slot yang disediakan. Lokasi tombol karakter didapatkan dengan mengambil data dari file detilkarakter1.json. Setelah data didapat, tombol dipasang pada pada slot-slot yang tersedia melalui fungsi datakart();

Tiga fungsi terakhir dalam *script* listKarakterPakai.cs merupakan fungsi-fungsi yang menangani jika ada pergantian dalam karakterset. Dalam pergantian karakter juga membutuhkan 3 *script* yang berbeda. Fungsi-fungsi ini nantinya akan dibahas di bawah.

```
    // Use this for initialization
    void Start () {
        inibutton = GetComponent<Button> ();
        displaydatakarakter bukapanel = GameObject.Find ("Canvas").GetComponent<displaydatakar
akter> ();

        inibutton = GetComponent<Button> ();
        inibutton.onClick.AddListener (() => bukapanel.displaydata (inisialkode));
        inibutton.onClick.AddListener (() => usecondition ());
        inibutton.onClick.AddListener (() => pencetesc());
    }

    void usecondition (){
            for (int i = 0; i < 5; i++)
                Destroy(GameObject.Find ("Karset ganti" + i));
    }

    void pencetesc(){
        GameObject.Find ("generalset").GetComponent<generalsettingPilihkar> ().datakarakterOn = tr
ue;
    }
}
```

Script di atas merupakan *script* btnDisplaykar.cs. *Script* btnDisplaykar.cs terdapat pada semua prefab tombol-tombol karakter. *Script* ini memiliki pengaturan action tombol untuk memicu fungsi-fungsi lai bekerja. Penambahan pengaturan

dalam *method Button.onClick.AddListener* sebenarnya terdapat pada pemrograman visual Unity Editor. Namun untuk menambahkan action yang tidak tersedia pada pemrograman visual, ditambahkan *method Button.onClick.AddListener*.

Setelah tombol ditampilkan dan memililki action, selanjutnya adalah bagaimana tombol menampilkan fungsinya. Pada scene ini, tombol karakter memiliki fungsi menampilkan data karakter. Data-data karakter ditampilkan pada satu panel yang dinamakan Panel Karakter. Pada Panel Karkater tersaji data-data mengenai kemampuan pemain, biografi, dan untuk menaikkan level karakter. Data-data ini diambil dari file detilkarakter1.json. Pada file ini menyimpan data-data karakter yang dapat digunakan untuk semua kebutuhan karakter.

```
public void displaydata(int idkar){
    //defie buat ambil data json
    string jsonstr = System.IO.File.ReadAllText(Application.persistentDataPath+"/detilkarakter1.js
on");
    ambildata = JsonMapper.ToObject (jsonstr);

    GameObject data = Instantiate (paneldata) as GameObject;
    RectTransform datart = data.GetComponent<RectTransform> ();
    RectTransform cnvsrt = gameObject.GetComponent<RectTransform> ();
    float rasio = cnvsrt.rect.width / 653;
    float tinggi = datart.rect.height * rasio;
    float lebar = datart.rect.width * rasio;

    datart.offsetMin = new Vector2 (-lebar / 2, -tinggi / 2);
    datart.offsetMax = new Vector2 (lebar / 2, tinggi / 2);

    data.transform.parent = gameObject.transform;
    data.name = "Panel Karakter";

    //tengahin. semoua perhitungan hasil kalibrasi
    RectTransform dataRT = data.GetComponent<RectTransform> ();
    tinggi = dataRT.rect.height;
    lebar = dataRT.rect.width;
    RectTransform panelmaskkarrt = GameObject.Find ("panelMaskKarakter").GetComponent<R
ectTransform> ();

    float x = -cnvsrt.rect.width / 2+10*rasio;
    float y = (-cnvsrt.rect.height / 2 + panelmaskkarrt.rect.height);
    dataRT.offsetMin = new Vector2 (x, y);

    x = dataRT.offsetMin.x + lebar;
    y = dataRT.offsetMin.y + tinggi;
    dataRT.offsetMax = new Vector2 (x, y);

    //data karakter
    detil.namakarakter = ambildata ["karakter"] [idkar] ["namakar"].ToString ();
    detil.hp = Convert.ToInt32 (ambildata ["karakter"] [idkar] ["hp"].ToString ());
    string hp = Convert.ToString (detil.hp);
```

```
detil.atkpoin = Convert.ToInt32 (ambildata ["karakter"] [idkar] ["atkpoin"].ToString ());
detil.level = Convert.ToInt32 (ambildata ["karakter"] [idkar] ["level"].ToString ());
string level = Convert.ToString (detil.level);
detil.spd = Convert.ToInt32 (ambildata ["karakter"] [idkar] ["spd"].ToString ());
detil.tipeserangan = ambildata ["karakter"] [idkar] ["tipe"].ToString ();
detil.prodtime = Convert.ToDouble (ambildata ["karakter"] [idkar] ["wktprod"].ToString ());
detil.penjelasan = ambildata ["karakter"] [idkar] ["penjelasan"].ToString ();

//tulis dalam panel(bagianText1)
Text teksnama = data.transform.Find ("Nama").GetComponent<Text> ();
teksnama.text = detil.namakarakter;
//HP
Text nyawa = data.transform.Find ("Text1").GetChild (0).GetComponent<Text> ();
nyawa.text = ": " + hp;
//Damage
if (detil.atkpoin >= 20) {
    Text damage = data.transform.Find ("Text1").GetChild (1).GetComponent<Text> ();
    damage.text = ": Kuat";
} else if (detil.atkpoin >= 10) {
    Text damage = data.transform.Find ("Text1").GetChild (1).GetComponent<Text> ();
    damage.text = ": Normal";
} else if (detil.atkpoin >= 5) {
    Text damage = data.transform.Find ("Text1").GetChild (1).GetComponent<Text> ();
    damage.text = ": Lemah";
} else {
    Text damage = data.transform.Find ("Text1").GetChild (1).GetComponent<Text> ();
    damage.text = ": Sangat lemah";
}
//level
Text levell = data.transform.Find ("Text1").GetChild (2).GetComponent<Text> ();
levell.text = ": " + level;

//Text2
//speed
if (detil.spd >= 20) {
    Text cepat = data.transform.Find ("Text2").GetChild (0).GetComponent<Text> ();
    cepat.text = ": Sangat cepat";
} else if (detil.spd >= 15) {
    Text cepat = data.transform.Find ("Text2").GetChild (0).GetComponent<Text> ();
    cepat.text = ": Cepat";
} else if (detil.spd >= 10) {
    Text cepat = data.transform.Find ("Text2").GetChild (0).GetComponent<Text> ();
    cepat.text = ": Normal";
} else {
    Text cepat = data.transform.Find ("Text2").GetChild (0).GetComponent<Text> ();
    cepat.text = ": Pelan";
}
//tipe serang
Text tipe = data.transform.Find ("Text2").GetChild (1).GetComponent<Text> ();
tipe.text = ": " + detil.tipeserangan;
//production time
if (detil.prodtime >= 8) {
    Text prdtime = data.transform.Find ("Text2").GetChild (2).GetComponent<Text> ();
    prdtime.text = ": Sangat lama";
} else if (detil.prodtime >= 6) {
    Text prdtime = data.transform.Find ("Text2").GetChild (2).GetComponent<Text> ();
    prdtime.text = ": Lama";
} else if (detil.prodtime >= 4) {
    Text prdtime = data.transform.Find ("Text2").GetChild (2).GetComponent<Text> ();
    prdtime.text = ": Normal";
} else {
    Text cepat = data.transform.Find ("Text2").GetChild (2).GetComponent<Text> ();
    cepat.text = ": Cepat";
}
```

```
        Text penjelas = data.transform.Find ("PanelDasaran").GetComponentInChildren<Text> ();
        penjelas.text = detil.penjelasan;

        //jsonmapper untuk ngecek apakah ad    a karakter di karset
        JsonData idkarset = JsonMapper.ToObject (System.IO.File.ReadAllText(Application.persistent
DataPath+"/karakterset1.json"));
        for (int a = 0; a < idkarset [0].Count; a++) {
            if (ambildata ["karakter"] [idkar] ["id"].ToString () == idkarset ["karakterset"] [a] ["idkar"].ToStri
ng ()) {
                data.transform.Find ("btnUse").GetComponent<btnUsescript> ().kodeubah = idkar;
                data.transform.Find ("btnUse").GetComponent<btnUsescript> ().goUse = false;
                a = idkarset [0].Count;
            } else if (ambildata ["karakter"] [idkar] ["id"].ToString () != idkarset ["karakterset"] [a] ["idkar"].
ToString ()) {
                data.transform.Find ("btnUse").GetComponent<btnUsescript> ().kodeubah = idkar;
                data.transform.Find ("btnUse").GetComponent<btnUsescript> ().goUse = true;
            }
        }
        //btnLevelupkar data.transform.Find("btnLevelup")
        data.transform.Find("btnLevelup").GetComponent<btnLevelupkar>().idkar = idkar;

        //ganti pfrofpict
        Sprite[] kotakkar=Resources.LoadAll<Sprite>("sprites/kar/lakon/karakterKotak");
        data.transform.Find ("profpict").GetComponent<Image> ().sprite = (Sprite)kotakkar [idkar];
    }
}
```

Pada *script* ini, data karakter diambil dan ditampilkan dengan *script* displaydatakarakter.cs pada fungsi displaydata(). Fungsi displaydata() diaktifkan oleh *action* dari tombol karakter yang telah diset sebelumnya. Fungsi ini bertugas untuk mengambil data dari file detilkarakter1.json, mengatur dimensi dari panel, dan menampilkan data pada objek yang telah dibuat. Semua data yang didapat dari *file* json dikonversi menjadi tipe data yang diinginkan agar dapat ditampilkan dengan benar. Pada bagian akhir fungsi terdapat pengaturan pada tombol pakai dan pindah. Pengecekan dilakukan dengan mencari tahu apakah tombol masuk dalam json karakterset1.json atau tidak. Apabila tombol tidak masuk dalam daftar karakterset maka tombol akan bertuliskan “Pakai”. Sedangkan apabila tidak masuh karakterset maka tombol akan bertuliskan “Pindah”. Selanjutnya untuk fungsi dari tombol pakai dipasangkan dengan *script* btnUse*script*.cs. *script* ini bertugas memberika action yang tepat pada tombol pakai dengan menambahkan method Button.OnClick.AddListener.

```
    public void use(int kodeubahkarset, bool gouse){
        if (gouse) {
            Text teksbtn = gameObject.transform.Find ("TextBtnUse").GetComponent<Text> ();
            teksbtn.text = "Pakai";
        }else {
            Text teksbtn = gameObject.transform.Find("TextBtnUse").GetComponent<Text> ();
            teksbtn.text = "Pindah";
        }
        Button inibutton;
        //buat send ke listKarakterPakai.cs
        listKarakterPakai editkar = GameObject.Find ("PanelMaskKarset").GetComponentInChildren<l
istKarakterPakai> ();
        inibutton = GetComponent<Button> ();
        inibutton.onClick.AddListener (() => editkar.usebtnpressed (kodeubahkarset));
        inibutton.onClick.AddListener (() => destroypanel ());
    }

    void destroypanel(){
        GameObject displaykark = GameObject.Find ("Panel Karakter");
        Destroy (displaykark);
    }
}
```

Untuk proses pergantian karakterset mengandalkan 3 *script* pada 3 objek yang berbeda. Pertama pada objek tombol dengan *script* btnDisplaykar.cs mengirim kode id dari karakter pada panel karakter. Selanjutnya pada tombol pakai yang ada dalam panel karakter akan mengirimkan id pada panel karakterset. Pada *script* displaydatakarakter.cs yang ada di objek panel karakterset terdapat fungsi usebtnpressed() yang memiliki perintah untuk membuat tombol-tombol Karset ganti 1 sampai 5 yang akan menjadi kode slot dari karakterset. Tombol-tombol ini memiliki action untuk masuk ke fungsi gantikejson() dengan mengirimkan kode id karakter yang akan dipindah dank kode slot tujuan. Fungsi ini akan mengecek dan memindahkan karakter pada file json karakterset1.json dengan susunan yang baru. Selanjutnya masuk pada fungsi adapergantian(). Fungsi ini berfungsi untuk menghancurkan objek-objek pada panel karakter set dan menggantinya menjadi yang baru. Fungsi ini juga memberikan perintah pada panel karakter didapat untuk memberi tampilan dan efek pada tombol.

```
    void inactiveoractive(string kode, GameObject item){
        //string jsonstrPk=Resources.Load<TextAsset> ("db/karakterset").text;
        string jsonstrPk=System.IO.File.ReadAllText(Application.persistentDataPath+"/karakterset1.js
on");
        JsonData idkarset = JsonMapper.ToObject (jsonstrPk);
```

```
    int k;
    for (k = 0; k < 5; k++) {
        if (idkarset ["karakterset"] [k] ["idkar"].ToString () != "-1") {
            if (kode == idkarset ["karakterset"] [k] ["idkar"].ToString ()) {
                Button inibutton = item.GetComponent<Button> ();
                inibutton.interactable = false;
                k = 5;
            }
        }
    }
}
```

Terakhir, pada objek panel karakter didapat, terdapat *script* listKrakterdidapat.cs. *Script* ini berfungsi untuk menampilkan semua karakter yang sudah didapat. Sama seperti pada *script* listKarakterPakai.cs, *script* ini membuat membaca data dari file json dan menampilkan tombol-tombol karakter dalam urutan slot yang baik. Selain itu, pada *script* ini terdapat pengaturan tombol karakter dalam kondisi aktif atau tidak. Pada fungsi inactiveoractive.cs tombol akan dicek apakah masuk dalam karakter set. Apabila masuk dalam karakterset, maka tombol akan tidak aktif.

4.5 Pengujian

Untuk mengetahui hasil dari implementasi dari konsep game, mengatahui *bug*, dan menguji efektifitas *game "Wana Warrior"* sebagai media edukasi penyelamatan satwa maka dilakukan kuesioner pengujian. Kuesioner dilakukan kepada responden sejumlah 35 orang dengan rentang usia antara 15-25 tahun. Rentang usia tersebut dibagi menjadi 3 kategori, yaitu usia 15-17 tahun, usia 18-22 tahun dan usia 23-25 tahun. Jumlah responden laki-laki dan perempuan juga seimbang, dengan jumlah responden laki-laki sebanyak 18 orang dan responden perempuan sebanyak 17 orang. Pengisian kuesioner dibagi menjadi 2 bagian, yaitu pre-test dan post-test, dan diantaranya responden memainkan *game "Wana Warrior"*.

Pada tahap pengisian kuesioner pengujian, responden sudah dalam kondisi memainkan *game "Wana Warrior"*. Kuesioner pengujian berisi pertanyaan yang sama untuk menguji meningkat tidaknya pengetahuan responden mengenai kondisi satwa di Indonesia.

Tabel 4.1 menunjukkan hasil dari kuesioner pengujian. Pengujian menunjukkan hasil yang positif dengan mayoritas responden mengalami peningkatan skor.sejumlah 24 responden mendapatkan skor yang lebih tinggi dari skor kuesioner pre-test yang mereka kerjakan sebelumnya. Dan jumlah responden yang memiliki skor lebih dari 6 juga mengalami peningkatan yang drastis. Dari mulai 15 orang menjadi 26 orang yang mendapatkan skor di atas 6. Hal ini menunjukkan *game "Wana Warrior"* membuat bertambahnya pengetahuan responden mengenai penyelamatan satwa di Indonesia.

Tabel 4.1. Hasil Kuesioner pre-test dibandingkan dengan hasil kuesioner pengujian

Responden	1	2	3	4	5	6	7	8	9	10
Pre-test	6	5	7	5.5	6	6	1.5	8	4.5	6
Post-test	7.5	5.5	8	5.5	6	6	5	8	4	7

Responden	11	12	13	14	15	16	17	18	19	20
Pre-test	5	5	8.5	8.5	2	7	7	7.5	6	4
Post-test	7.5	6.5	7	7	1.5	9	9	7	8	6

Responden	21	22	23	24	25	26	27	28	29	30
Pre-test	4.5	5	5.5	3.5	9	6.5	8.5	8.5	6.5	5
Post-test	8.5	10	9	9	10	10	9	9	10	8

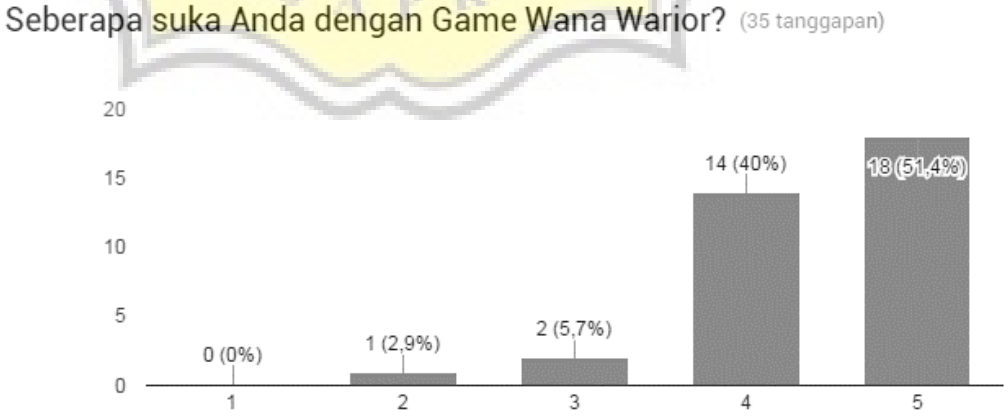
Responden	31	32	33	34	35
Pre-test	6.5	6.5	7	7.5	5.5
Post-test	10	9	7	8	6.5

N	Pr>6	Po>6
24	16	27

N = Jumlah responden yang mengalami kenaikan nilai pre-test dibanding post-test,
Pr=Nilai pre-test,
Po=Nilai Post-test

Setelah dilakukan pengujian pengetahuan responden mengenai keanekaragaman hayati dan penyelamatan satwa, selanjutnya responden diberikan pertanyaaan yang bersifat opini mengenai *game "Wana Warrior"*.

Dari data pada gambar 4.59 menunjukan bahwa sebagian besar responden menyukai *game* "Wana Warrior" secara umum. Hanya 3 responden yang *menganggap game "Wana Warrior"* cukup disukai untuk mereka.

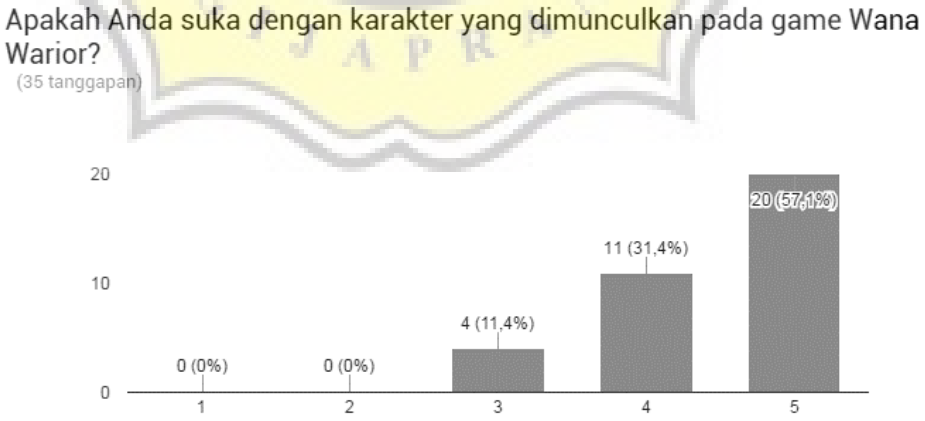


Gambar 4.59 Diagram batang kesukaan responden terhadap game "Wana Warrior"

Gambar 4.60 menunjukkan respon responden terhadap tampilan grafis *game "Wana Warrior".* Tampilan grafis juga menjadi faktor yang cukup membuat responden menyukai *game "Wana Warrior".* Sebanyak 27 orang memberika skor tinggi pada grafis. Hanya satu orang yang memberikan skor 2 untuk grafis.

Gambar 4.60 Diagram batang kesukaan responden terhadap tampilan grafis game "Wana Warrior"

Gambar 4.61 menunjukkan respon responden terhadap karakter *game "Wana Warrior".* Karakter dalam *game "Wana Warrior"* juga mendapatkan nilai yang positif. Tidak ada responden yang memberikan skor yang rendah pada karakter dan sebagian besar memberikan skor 5 untuk karakter.



Gambar 4.61 Diagram batang pendapat responden mengenai karakter game "Wana Warrior"

Gambar 4.62 menunjukkan respon responden terhadap kemudahan memainkan *game "Wana Warrior"*. Untuk kemudahan bermain, *game* "Wana Warrior" mendapatkan skor yang tinggi juga. Dengan menggunakan *gameplay* yang mudah mungkin membuat *game "Wana Warrior"* mendapatkan skor tinggi untuk kemudahan bermain.

Gambar 4.62 Diagram batang kemudahan memainkan game "Wana Warrior" menurut responden

Dari diagram pada gambar 4.63 menunjukkan bahwa seluruh pemain merasa info-info yang diberikan pada *game "Wana Warrior"* berguna bagi mereka untuk memahami mengenai kondisi satwa dan alam Indonesia.

Gambar 4.63 Diagram lingkaran tingkat kegunaan informasi dari game "Wana Warrior" bagi responden