Implementasi Collision Detection Permainan Fly Bee

Menggunakan Game Maker

Tugas Akhir disusun untuk memenuhi syarat

mencapai gelar Kesarjanaan Komputerpada

Program Studi Teknik Informatika

Jenjang Program Strata-1

Oleh :

ZUNIAR FARIH FAHLEVY

12.01.53.0048

14993

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS STIKUBANK (UNISBANK)

SEMARANG

2017

PERNYATAAN KESIAPAN UJIAN TUGAS AKHIR

Saya, Zuniar Farih Fahlevy, dengan ini menyatakan bahwa laporan tugas akhir yang berjudul:

IMPLEMENTASI COLLISION DETECTION PERMAINAN FLY BEE MENGGUNAKAN GAME MAKER

Adalah benar hasil karya saya dan belum pernah diajukan sebagai karya ilmiah, sebagian atau seluruhnya, atas nama saya atau pihak lain.

(Zuniar Farih Fahlevy)

NIM : 12.01.53.0048

Disetujui oleh pembimbing

Kami setuju laporan tersebut diajukan untuk Ujian akhir

Semarang : 26 Januari 2017

(Dr. EDY WINARNO, S.T., M.Eng)

Pembimbing

ii

UNIVERSITAS STIKUBANK "UNISBANK" SEMARANG
FAKULTAS TEKNOLOGI INFORMASI
Rektorat Kampus Mugas
Jl. Tri Lomba Juang No. 1 Semarang 50241
Telp. (024) 8451976, 8311668, 8454746 Fax (024) 8443240
E-mail : info@unisbank.ac.id
Kampus Kendeng :
Jl. Kendeng V Bendan Ngisor Semarang
Telp. (024) 8414970, Fax (024) 8441738
E-mail : fe@unisbank.ac.id

SURAT PERNYATAAN KEASLIAN TUGAS AKHIR/ SKRIPSI

Yang bertanda tangan di bawah ini, saya menyatakan bahwa TUGAS AKHIR / SKRIPSI dengan Judul :

IMPLEMENTASI COLLISION DETECTION PERMAINAN FLY BEE MENGGUNAKAN GAME MAKER

yang telah diuji di depan tim penguji pada tanggal 26 Januari 2017, adalah benar hasil karya saya dan dalam TUGAS AKHIR /SKRIPSI ini tidak terdapat keseluruhan atau sebagian tulisan orang lain yang saya ambil dengan cara menyalin, atau meniru dalam bentuk rangkaian kalimat atau simbol yang saya aku seolah-olah sebagai tulisan saya sendiri dan atau tidak terdapat bagian atau keseluruhan tulisan yang saya salin, tiru atau yang saya ambil dari tulisan orang lain tanpa memberikan pengakuan pada penulis aslinya.

Apabila saya melakukan hal tersebut diatas, baik sengaja maupun tidak, dengan ini saya menyatakan menarik TUGAS AKHIR / SKRIPSI yang saya ajukan sebagai hasil tulisan saya sendiri.

Bila kemudian terbukti bahwa saya ternyata melakukan tindakan menyalin atau meniru tulisan orang lain seolah-olah hasil pemikiran saya sendiri, berarti gelar dan ijazah saya yang telah diberikan oleh Universitas Stikubank (UNISBANK) Semarang batal saya terima.

METERAI TEMPEL E0B1CAEF262329122 6000 ENAM RIBU RUPIAH

.....

Menyatakan

(ZUNIAR FARIH FAHLEVY)

NIM :12.01.53.0048

SAKSI 1
Tim Penguji

(Dr. EDY WINARNO, ST., M.Eng)

SAKSI 2
Tim Penguji

(Dr. Drs. ERI ZULIARSO, M.KOM)

SAKSI 3
Tim Penguji

(EDDY NURRAHARJO, ST, M.Cs)

HALAMAN PENGESAHAN

Telah dipertahankan di depan tim dosen penguji Tugas Akhir fakultas Teknologi Informasi UNIVERSITAS STIKUBANK (UNISBANK) Semarang dan diterima sebagai salah satu syarat guna menyelesaikan Jenjang Program Strata 1, Program studi : Teknik Informatika

Semarang :26 Januari 2017

Ketua

(Dr. Edy Winarno, ST., M.Eng)

Sekertaris

(Dr. Eri Zuliarso, M.Kom)

Anggota

(Eddy Nurraharjo, ST, M.Cs)

MENGETAHUI :

UNIVERSITAS STIKUBANK (UNISBANK) SEMARANG

Fakutas Teknologi Informasi

Dekan

(Dr. Drs. Yohanes Suhari, M.MSI)

MOTTO DAN PERSEMBAHAN

MOTTO

- Hidup sekali huduplah yang berarti
- Bondo bahu pikir lek perlu sak nyawane pisan
- وَمَا الَّذَّةُ أَلاَ بَعْدَ التَّعَبِ

PERSEMBAHAN

- Allah S.W.T yang telah memberikan anugerah hidup, kesehatan dan keselamatan hingga saat ini.
- Kedua orang tua, bapak dan ibu tercinta yang telah melahirkan dan membesarkanku hingga kelak menjadi anak yang berbakti dan berguna
- Adinda khukmailya rohmawati yang selalu sabar memberikan semangat .
- Semua sahabat dan saudaraku yang selalu memberikan dorongan semangat.
- Teman teman angkatan Teknik Informatika Unisbank Semarang

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS STIKUBANK (UNISBANK) SEMARANG

Program Studi: Teknik Informatika

Tugas Akhir Sarjana Komputer

Semester Genap Tahun 2016

IMPLEMENTASI COLLISION DETECTION PERMAINAN FLY BEE MENGGUNAKAN GAME MAKER

Zuniar Farih Fahlevy

12.01.53.0048

Email : zuniarfarih@gmail.com

Abstraksi

Fly bee adalah sebuah aplikasi game yang di tunjukan untuk sistem operasi android dengan genre arcade game, yaitu game dengan berfokus mengejar point dan haighscore, *game fly bee* ini menggunakan karakter lebah sebagai karakter utama didalamnya pemain akan dibawa untuk mengendalikan seekor lebah untuk menghindari musuh seperti pipa yang secara random menghadang bee.

Dalam perjalanan permainan bee peneliti membangun dengan aplikasi game maker, game maker membangun game fly bee dengan sistem drop down dan dengan bahasa pemrograman game maker itu sendiri yaitu GML (game maker language) dengan aplikasi ini juga dapat mengimplementasikan metode *collision detection*.

Penelitian ini mengunakan metode *Collision detection* adalah metode yang digunakan dalam menentukan pergerakan dalam game seperti tubrukan suatu objek. Dengan menggunakan algoritma ini dapat menetukan arah serta tujuan dalam membuat game agar dapat mudah dinikmati, metode ini juga membuat dunia game semakin mirip dengan kenyataan di dunia nyata, dengan metode *collision* dapat membuat game *fly bee* menjadi nyata .

Kata kunci : *collision detection, game, berbasis android, fly bee, Game maker*

Semarang :26 januari 2017
Pembimbing

(Dr. EDY WINARNO, S.T., M.Eng)

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa atas berkat dan rahmat-Nya, penulis dapat menyelesaikan laporan penelitian di Universitas Stikubank (UNISBANK) ini dengan baik. Skripsi yang berjudul "*implementasicollision detection* permaianan *fly bee* menggunakan game maker, aplikasi ini disusun guna memenuhi kewajiban pengambilan mata kuliah skripsi sebagai salah satu syarat yang harus ditempuh dalam kelulusan mahasiswa jenjang strata 1 pada jurusan Teknik Informatika. Selama penulis menyusun laporan ini banyak sekali mendapat bantuan, dukungan serta bimbingan dari berbagai pihak. Untuk itu penulis mengucapkan banyak terima kasih kepada :

1. Allah SWT atas segala rahmat dan karuniaNYA sehingga saya berhasil menyelesaikan skripsi ini.
2. Bapak, ibu, kakak beserta keluarga besar saya yang selalu mendukung baik secara moral dan material.
3. Bapak Dr. Drs. Yohanes Suhari, M.MSI selaku Dekan Fakultas Teknologi Informasi Universitas Stikubank.
4. Bapak Jati Sasongko Wibowo, S.Kom, M.Cs, selaku Ketua Program Studi Teknik Informatika Universitas Stikubank.
5. Bapak Dr. Edy Winarno,S.T.,M.Eng yang memberikan banyak pengarahan, bimbingan, dan masukan yang sangat berguna bagi penyelesaian skripsi ini.
6. Bapak dan Ibu Dosen beserta staff di Universitas Stikubank.

7. Teman-teman kelas yang telah membatu, memberi semangat, dan masukan dalam membuat skripsi penulis.
8. Semua pihak yang telah membantu penulis dalam menyelesaikan penulisan skripsi ini yang tidak dapat disebutkan satu-persatu.

Penulis menyadari bahwa penelitian ini masih jauh dari kata sempurna, untuk itu penulis sangat berharap para pembaca maupun semua pihak yang terkait memberikan kritik dan saran yang bersifat membangun demi kesempurnaan dalam penelitian selanjutnya.

Semarang : 26 Januari 2017

(Zuniar Farih Fahlevy)

DAFTAR ISI

Pernyataan Kesiapan Ujian Tugas Akhir……ii

Halaman Pengesahan iv

Motto Dan Persembahan v

Abstraksi vi

Kata Pengantar vi

Daftar Isi..... ix

Daftar Gambar..... xiii

Daftar Tabel xvi

Daftar Singkatan..... xvii

BAB I 1

PENDAHULUAN..... 1

1.1 Latar Belakang..... 1

1.2 Rumusan Masalah..... 3

1.3 Batasan Masalah 4

1.4 Tujuan Penelitian 4

1.5 Manfaat penilitian 5

1.6 Metodologi Penelitian..... 5

1.7 Sistematika Penulisan 7

BAB II..... 9

TINJAUAN PUSTAKA..... 9

2.1 Pustaka yang terkait dalam penelitian ini 9

2.2 Perbedaan Penelitian yang terdahulu 20

BAB III 21
LANDASAN TEORI 21
3.1 Pengertian Android 21
3.2 Arsitektur Sistem Operasi Android 23
3.3 Lima Struktur Android 24
3.4 Pengertian Game 25
3.5 Elemen Penyusun Game 26
3.5.1 Genre /Jenis Game 27
3.5.2 Karakter Game 29
3.5.3 Background 29
3.5.4 Musik 29
3.6 Collision Detection 29
BAB IV 32
METODE PENELITIAN 32
4.1 Konsep Game 32
4.2 Analisis Kebutuhan 33
4.2.1 Kebutuhan Perangkat Keras 33
4.2.2 Kebutuhan Perangkat Lunak 33
4.3 Implementasi Metode Collision Detection 35
4.4 Rancangan Struktur Menu 41
4.5 Rancangan Algoritma Program 41
4.6 Analisis Fungsi 43
4.6.1 Use Case Diagram 43
4.6.2 Activity Diagram 46
4.7 Rancangan Layout / Tampilan 50

BAB V 54
IMPLEMENTASI 54
5.1 Implementasi 54
5.2 Membangun Dan Menginstal Aplikasi 54
5.2.1 Pembangunan Aplikasi Fly Bee 54
5.2.2 Menginstal Aplikasi 60
5.3 Hasil Instalasi Aplikasi 63
5.4 Tampilan Karakter Pada Game Fly Bee 63
5.5 Skenario Pada Game Fly Bee 64
5.6 Tampilan Menjalankan Game Fly Bee 68
5.6.1 Tampilan Pembuka 68
5.6.2 Tampilan Menu Utama 69
5.6.3 Tampilan Halaman Cara Main 70
5.6.4 Tampilan Tentang 71
5.6.5 Tampilan Memulai Game 72
5.6.6 Tampilan Game Over 73
5.7 Pengujian Sistem 74
5.7.1 Pengujian Pada Smartphone 74
BAB VI 79
HASIL PENELITIAN DAN PEMBAHASAN 79
6.1 Hasil 79
6.2 Pembahasan 82
6.2.1 Game maker 82
6.2.2 Skenario dalam game fly bee 82
6.2.3 Implementasi collision detection 82

BAB VII 91
KESIMPULAN DAN SARAN 91
7.1 Kesimpulan 91
7.2 Saran 91
DAFTAR PUSTAKA 92
LAMPIRAN 94

DAFTAR GAMBAR

Gambar 3.1 Arsitektur Android (Arifan, 2015) 23
Gambar 3.2 Algoritma Collision Bounding Box 30
Gambar 3.3 Bounding Circle Collision 31
Gambar 4.1 Logo Windows 10 33
Gambar 4.2 Logo Game Maker V1.4 34
Gambar 4.3 logo android 6.0 marshmallow 34
Gambar 4.4 logo corel draw x5 35
Gambar 4.5 Logo Photoshop Cs3 35
Gambar 4.6 *Flowchart Collision Detection* Antara Panah dengan Musuh *(Dokkaebi)* 37
Gambar 4.7 Min-max *bounding box* 38
Gambar 4.8 *Bound* bertabrakan 39
Gambar 4.9 *Bound* tidak bertabrakan 40
Gambar 4.10 *Struktur menu* 41
Gambar 4.11 Flowchart *Game Fly Bee* 42
Gambar 4.12 *Use case diagram* 43
Gambar 4.13*Activity diagram* main 46
Gambar 4.14*Activity diagram cara main* 47
Gambar 4.15*Activity diagram tentang* 48
Gambar 4.16*Activity diagram* keluar 49
Gambar 4.17 Rancangan Menu Utama 50
Gambar 4.18 Rancangan tentang 51
Gambar 4.19 Rancangan Cara Main 52
Gambar 4.20 Rancangan Saat Permainan 53
Gambar 5.1 Hasil Instalasi *Game Maker* 55
Gambar 5.2 Membuat *Project* Baru 55
Gambar 5.3 Sprite Yang Sudah Di Masukan 56
Gambar 5.4 Hasil Memasukan *Background* 57
Gambar 5.5 Sound Yang Telah Di Masukan 58

Gambar 5.6 *Font* yang telah dimasukan 59
Gambar 5.7 Tampilan Project Pada Game Maker 60
Gambar 5.8 Save Pada *Creat Application* Pada *Game Maker*..... 61
Gambar 5.9 Penyimpanan *Project File Apk* Pada *Game Maker*..... 62
Gambar 5.10 Instalasi Pada *Smartphone* 62
Gambar 5.11 Hasil Instalasi Aplikasi *Fly Bee* 63
Gambar 5.12 Sprite Flying Bee Dan Falling Bee 65
Gambar 5.13 Sprite Pipa 65
Gambar 5.14 Sprite Background Menu, Tentang, Cara Main, Menu 1..... 66
Gambar 5.15 Sprite Tombol Main, Tentang, Cara Main, Keluar 66
Gambar 5.16 Sprite Land..... 66
Gambar 5.17 Sprite Effect Sentuh 66
Gambar 5.18 Sprite Score Up 67
Gambar 5.19 Sprite Judul..... 67
Gambar 5.20 Sprite Bingkai Dan Game Over 67
Gambar 5.21 Sprite Sentuh Dan Bersiap 67
Gambar 6.22 Sprite Garis Unvisible 68
Gambar 5.23 Tampilan Pembuka..... 68
Gambar 5.24 Menu Utama..... 69
Gambar 5.25 Tampilan Cara Main..... 70
Gambar 5.26 Tampilan Tentang 71
Gambar 5.27 Game Berjalan..... 72
Gambar 5.28*Game Over* 73
Gambar 5.29*Icon fly bee* yang telah terinstal..... 75
Gambar 5.30 Tampilan menu utama saat di jalankan..... 76
Gambar 5.31 Tampilan saat permainan berjalan..... 76
Gambar 5.32 Tampilan cara main..... 77
Gambar 5.33 Tampilan tentang..... 78
Gambar 5.34 Tampilan *game over* serta *score* tertinggi telah dicapai..... 78
Gambar 6.1 (a) Hasil tampilan menu utama 81
Gambar 6.1 (b) Menampilkan main 81

Gambar 6.1(c) Hasil tampilan cara main 81
Gambar 6.1(d) Hasil tampilan tentang 81
Gambar 6.2 Titik kordinat collision detection 84
Gambar 6.3 Tool *collision* pada game maker 85
Gambar 6.4 Tabrakan obj_bee dengan pipa1 86
Gambar 6.5 Tabrukan obj_bee dengan pipa 2 87
Gambar 6.6 Tabrakan obj_bee dengan garis unvisible 88
Gambar 6.7 Tabrakan obj_bee dengan obj_land 90

DAFTAR TABEL

Tabel 2.1 Perbedaan Penelitian Terdahulu..... 18
Tabel 3.1 Versi Android 22
Tabel 4.1 Deskripsi *Use Case Diagram* Mulai Permainan..... 44
Tabel 4.2 Deskripsi *Use Case Diagram* melihat *tentang*..... 44
Tabel 4.3 Deskripsi *Use Case Diagram* melihat *cara main* 45
Tabel 4.4 Deskripsi *Use Case Diagram* keluar permainan..... 45
Tabel 5.1 Karakter *Fly Bee* 64
Tabel 6.1 Hasil Pengujian Pada Aplikasi *Fly Bee*..... 80

DAFTAR SINGKATAN

OS	Operating System
GML	Game Maker Lenguage
AR	Augmented Reality
3D	Three Dimensional
AI	Artificial Intelligence
OHA	Open Handset Alliance
API	application programming interface
SMS	Short Message Service
WiFi	Wireless Fidelity
IPC	Interprocess Communication
BSD	Berkeley Software Distribution
SGL	Seeed Graphic Library
SSl	Server Side Include
Apk	Android Package

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perjalanan waktu telah menujukkan kemajuan zaman dalam bidang teknologi serta informasi, perubahan kemodernisasi begitupun permainan yang semakin canggih dalam pertumbuhannya. Begitu juga perkembangan dunia gadget kini berkembang dengan sangat pesat. Hampir setiap orang memiliki satu buah gadget seperti *featured-phone*, *smartphone,* atau *tablet*. Seiring dengan maraknya perkembangan *gadget*. Dari beberapa *smartphone* atau tablet terdapat berbagai *oprating system* seperti *Windows, Symbian, Android,* dll

Android merupakan OS *Mobile* yang tumbuh di tengah OS lainnya yang berkembang dewasa ini.OS lainnya seperti *Windows Mobile, i-Phone OS, Symbian* dan masih banyak lagi juga menawarkan kekayaan isi dan keoptimalan berjalan di atas perangkat *hardware* yang ada (Purwanto, 2013).

Aplikasi Android saat ini sedang populer dan menjadi salah satu sistem aplikasi yang paling banyak digunakan didunia saat ini,perkembangan aplikasi digital yang dapat dijalankan ada beberapa keperluan dengan bermacam macam kegunaan diantaranya aplikasi yang mencakup bisnis, ekonomi,dan permaianan atau game.

Game atau permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius dengan tujuan *refresing*. Bermain game sudah

dapatdikatakan sebagai *life style* masyarakat dimasa kini. Dimulai dari usia anak-anak hingga orang dewasa pun menyukai *video game*. Itu semua dikarenakan bermain *video game* adalah hal yang menyenangkan (prasetyo,2014)

Adapun permainan di android pun berkembang dengan pesat. Salah satu tipe permainan yang mendapatkan respon yang sangat baik adalah *endless-running game* Hal inilah yang melatarbelakangi pembuatan permainan *fly bee* ini yang bertipe 2D *endless-running game*.

Menurut Arsandi dkk (2012), *collision detection* adalah proses pendeteksian tabrakan antara dua objek. Sebenarnya dalam simulasi penelitian ini tabrakan tidak hanya terjadi antara dua objek, tetapi dapat juga terjadi antara satu objek dengan banyak objek sehingga dibutuhkan *collision detection* yang akurat. *Collision detection* juga berguna untuk menentukan posisi dari satu objek dengan objek yang lain agar tidak ada objek yang saling menembus,sehingga simulasi yang akan dibuat memiliki kesamaan dengan realita yang ada.

Algoritma *collision detection* adalah proses pengecekan apakah beberapa buah objek spasial saling bertumpuk atau tidak. Jika ternyata ada paling sedikit dua buah objek yang bertumpuk, maka kedua objek tersebut dikatakan saling bertumpukkan. Pada ruang spasial dua dimensi. Objek yang bertumpuk berarti objek spasialnya beririsan (Nugraha, 2013).

Deteksi tumbukan pada permainan 2 dimensi dilakukan dengan membuat *bounding-box* pada objek-objek yang akan mengalami tumbukan. Deteksi tumbukan sederhana dapat menggunakan *bounding-box* yang

berbentuk kotak ataupun lingkaran Algoritma *Axis-Aligned Bounding Box* bekerja dengan memastikan tidak adanya jarak Antara kedua buah box tersebut. Jika ada, maka kedua *box* tersebut tidak mengalami tumbukan tumbukan. Algoritma *Circle Collision* akan memeriksa apakah jarak antar 2 titik pusat lingkaran lebih besar dari pada jumlah kedua jari-jari. Jika ya, maka tidak terjadi tumbukan jika tidak maka telah terjadi tumbukan (handoyo,2014)

Putrady (2011) menjelaskan tentang *collision detection* bagaimana cara mengetahui objek-objek apa saja yang bersentuhan dalam bidang koordinat tertentu. Objek-objek ini bisa saja memiliki bentuk yang sangat bervariasi. Dengan demikian bahwa objek-objek pada game memiliki bentuk yang bervariasi, ada yang berbentuk kotak, persegi, sampai bentuk pesawat pemain yang sangat mendetail. Untuk mempercepat proses pada *collision detection*, umumnya objek-objek ini direpresentasikan secara logik dengan bentuk primitif seperti segiempat dan lingkaran (jika pada koordinat dua dimensi), atau kubus dan bola (jika pada koordinat tiga dimensi). Bentuk primitif yang merepresentasikan objek ini biasa disebut sebagai *Bounding Box atau Bounding Circle*.

dengan hal tersebeut peneliti menerapkan pada penelitian ini metode *collision detection* pada permainan *fly bee* pada *game maker*

1.2 Rumusan Masalah

Proses pembuatan aplikasi akan dibahas beberapa masalah pembuatannya diantaranya sebagai berikut :

1. Bagaimana pembuatan game yang menarik serta mudah untuk di buat serta di mainkan
2. Bagaimana metode C*ollision Detection* bisa diterapkan dalam game ini

1.3 Batasan Masalah

Dengan penelitian ini terdapat keperluan yang pembatasan masalah agar peneliti terarah dan lebih mudah pembahasa dan pembangunan aplikasi ini,adalah sebgai berikut :

1. Aplikasi ini dibangun menggunakan *game maker* yaitu *engine* untuk membuat game dengan metode *drop down*
2. Bahasa yang di gunakan adalah GML (*game maker language*) dan *javascrip*
3. Permainan di batasi 1 level saja dimana permaian selesai mendapatkan score
4. Permainan game ini di khusus kan untuk pengguna android
5. Pada game ini menggunakan untuk *standalone* dan tidak untuk *client server* atau *multiplayer* seperti yang di gunakan untuk dua player
6. Mengimplementasikan metode *collision detection* pada game ini untuk mengetahui tabrakan objek

1.4 Tujuan Penelitian

Dengan melihat permasalahan yang telah teruraikan dengan itu maka tujuan yang ingin di dapat adalah sebagai berikut :

1. Membuat aplikasi game yang menarik dengan model *standalone* dengan karakter *bee* atau lebah yang menghindari rintangan, dibuat agar dapat menarik terdapat *score* yang membuat pemaian dapat mencapai

maksimal skore yang telah dilampauinya dan implementasi deteksi tabrak pada objek tersebut. Serta menghasilkkan game yang sesuai dengan keinginan pemain sekarang.

2. Sebagai bahan refrensi untuk penelitian yang akan datang tentang implementasi *collision detection* pada game.

1.5 Manfaat penilitian

1. Bagi pemain /user

   Mengharapkan dapat di gunakan dan dimainkan permaianan dan mengasah pemain.

2. Bagi Unisbank

   Sebagai bahan refrensi bagi mahasiswa unisbank untuk implementasi *collision detection* pada game dan pembuatan game dengan *game maker*.

3. Bagi peneliti

   Sebagai pengembangan ilmu yang telah di dapat dalam perkuliahan dan dan menerapkannya serta mengetahui permasalahan pembuatan game dengan *game maker*.

1.6 Metodologi Penelitian

Metode penelitian yang digunakan adalah metode eksperimental. Metode eksperimental adalah metode penelitian yang didasarkan pada suatu percobaan – percobaan ilmiah yang dilakukan dalam membuat sesuatu yang baru atau mengembangkan sesuatu berdasarkan ilmu-ilmu pengetahuan

1. Tahap Pengumpulan data

   Model pengumpulan data yang digunakan dalam penelitian ini adalah sebagai berikut :

   a. Studi pustaka

      Pengumpulan data dengan cara mengumpulkan bacaan-bacaan yang ada, baik dari buku-buku, karya tulis dan lain sebagainya yang ada kaitannya dengan judul penelitian.

   b. Observasi

      Teknik pengumpulan data dengan mengadakan penelitian yang sudah ada sebelumnya

2. Tahap Pembangunan Aplikasi

   Proses pembangunan aplikasi *game fly bee* ini, digunakan metode pengembangan perangkat lunak secara *Waterfall* yang meliputi:

   a. Analisis (*Analysis*)

      Analisis adalah tahap menganalisis hal-hal yang diperlukan dalam pelaksanaan proyek pembuatan aplikasi.

   b. Perancangan (*Planning*)

      Perancangan adalah tahap perencanaan dari data yang dianalisis kedalam bentuk yang mudah dimengerti oleh pengguna.

   c. Desain (*Design*)

      Desain adalah tahap menggambarkan karakter-karakter *game*, baik itu *karakter* maupun *background* sebagai map yang akan digunakan.

d. Pengkodean (*Coding*)

Pengkodean adalah tahap penerjemahan data atau pemecahan masalah yang telah dirancang ke dalam bahasa pemrograman tertentu dan pemberian tahap alur langkah objek dalam game tersebut.

e. Pengujian (*Testing*)

Pengujian adalah tahap pengujian terhadap game telah jadi dibuat.

f. Perawatan (*Maintenance*)

Perbaikan adalah tahap perawatan dan perbaikan aplikasi yang telah di buat dan di main kan.

1.7 Sistematika Penulisan

Sistematika penulisan laporan ini akan disajikan dalam 7 (tujuh) bab, yang masing-masing bab sebagai berikut :

BAB I : PENDAHULUAN

Bab ini menjelaskan latar belakang masalah serta perumusan masalah,tujuan dan manfaat perumusan masalah, metodologi penelitian dan sistematika penulisan.

BAB II :TINJAUAN PUSTAKA

Bab ini menjelaskan tentang tinjauan pustaka terhadappenelitian-penelitian sebelumnya dan menghubungkan dengan masalah penilitian yang sedang diteliti.

BAB III : LANDASAN TEORI

Bab ini menjelesakan teori-teori dasar permasalahan dalam penelitian dan membahas berbagai konsep dasar, pengertian game, pengertian, pengertian *collision detection*, metode yang digunakan, game maker.

BAB IV : METODE PENELITIAN

Bab ini mejelaskan tentang metode penelitian dan analisa serta perancangan system yang akan di gunakan untuk mendesain program.

BAB V : IMPLEMENTASI SISTEM

Pada bab ini menerangkan hasil atau implementasi dan langkah langkah penggunaan yang telah di buat, serta pembangunan aplikasi.

BAB VI : HASIL PENELITIAN DAN PEMBAHASAN

Pada bab ini menjelaskan hasil pembahasan dari penelitian yang telah di buat.

BAB VII : SARAN DAN KESIMPULAN

Pada bab ini berisi tentang saran dan kesimpulan berdasarkan dari hasil keseluruhan penelitian .

BAB II

TINJAUAN PUSTAKA

Pada bab ini akan dijelaskan tentang tinjauan pustaka yang dipakai pada penelitian ini, pembahasan tentang tinjauan pustaka dari hasil penelitian terdahulu serta perbandingan dengan penelitian sekarang.

2.1 Pustaka yang terkait dalam penelitian ini

Teguh Dkk (2013) melakukan penelitian tentang simulasi animasi tiga dimensi gerombolan ikan dalam akuarium virtual menggunakan Algoritma *Artificial Bee Colonydan Bounding Box Collision Detection* dalam penelitiannya bahwa dalam algoritma *Artificial Bee Colony* (ABC), sangat baik dalam menentukan pergerakan simulasi grombolan ikan dan pada proses awal dari algoritma *Artificial Bee Colony* (ABC) adalah menentukan nilai parameter kontrol yang dimasukkan oleh pengguna, peneliti juga membahas tentang penerapan algoritma *bounding box collision* pada akhir simulasinya dengan demikian penentuan metode menurut peneliti sebagian penentuan arah gerakan pada ikan simulasi tersebut dalam hal ini, Metode *Bounding Box Collision Detection* hanya melakukan proses pendeteksian tabrakan tetapi tidak memberikan solusi agar tidak terjadi tabrakan. Solusi yang digunakan dalam penelitian ini merupakan pengujian dari penulis.

Musfiroh Dkk (2014) melakukan penelitian tentang aplikasi game *Dokkaebi Shooter* berbasis android menerapkan algoritma *collision detection* untuk medeteksi serangan tabrakan dan algoritma *boids* untuk pergerakan musuhnya. Metode yang digunakan dalam penelitian ini adalah metode *prototype*,

yaitu metode yang memberikan ide bagi sistem analis atau pemrogram dalam menyajikan gambaran lengkap sistem dari identifikasi kebutuhan user (pemakai), mengembangkan *prototype*, menentukan *prototype*, apakah *prototype* dapat diterima user dan penggunaan dari *prototype.* Hasil dari penelitian ini yaitu menghasilkan *game mobile* android berupa *game single player* tembakan vertikal *monster dokkaebi* yang dapat melatih ketangkasan pemain dengan tampilan 2D dan terdiri dari 3 level, yang dapat dijalankan pada *smartphone* maupun tablet android, kemudian pengimplementasiannya ke dalam *source code* Implementasi Algoritma *Collision Detection* Antara *Dokkaebi* dan Panah *Player* dan implementasinya dalam antarmuka permaianan, Implementasi Algoritma *Collision Detection* dalam antar muka aplikasi, dalam hal ini juga peneliti berpendapat *game Dokkaebi Shooter* dan dengan demikian dapat disimpulkan bahwa kelebihan dari game ini yaitu game ini dapat mengimplementasikan algoritma *collision detection* dan *boids* dengan baik, *game* ini tidak hanya bergantung pada matinya musuh namun juga tergantung *timer* untuk memenangkan *game* ini, musuh dapat menyerang pemain, dan dapat menyimpan *score*.

Darmawan (2014) melakukan penelitian tentang permainan Super Noseman, pembuatan permainan Super Noseman ini yang bertipe 2D *endless-running game*. Dalam pembuatannya, permainan Super Noseman ini akan menerapkan beberapa hal seperti efek *parallax*, efek *endless-running*, dan deteksi tumbukan. Efek *parallax* adalah efek yang dilihat mata seseorang terhadap perpindahan atau besarnya sebuah benda yang yang disebabkan oleh perbedaan posisi benda tersebut. Dalam hal ini permaian super noseman cara bermain

permainan adalah menyelamatkan penduduk di dalam sebuah gedung dengan cara menangkapnya dan memukul para ninja yang terdapat dalam gedung tersebut. Super Noseman akan terus berlari hingga permainan berakhir. Permainan akan berakhir jika Super Noseman tidak berhasil menyelamatkan atau memukul ninja (terlewatkan) dengan jumlah tertentu. Nilai akan didapatkan tergantung dengan jumlah penduduk yang berhasil diselamatkan dan ninja yang berhasil dipukul . dan untuk pengimplementasian dalam *endless game* agar memiliki *endless running effect,* maka *background* pada permainan ini harus bergerak terus menerus ke arah kiri layar. Oleh karena itu, sisi kiri dan kanan *background* harus dapat digabung tanpa terlihat.gambar *background* yang diperlukan untuk menciptakan efek *endless running* pada permainan Super Noseman. Untuk *efek parallax* dapat ditimbulkan dengan cara memisahkan gambar *background* menjadi beberapa layer.

Susilawati (2014)melakukan penelitian tentang perancangan game *space ship* dengan mengunakan metode *quad tree*. *Spaceship* adalah pesawat luar angkasa. Kebanyakan permainan ini tentang misi, Langkah-langkah yang dilakukan pada permainan ini menggunakan Algoritma *QuadTree* :

1. Tentukan titik awal dan titik tujuan
2. Periksa nilai titik awal dan titik tujuan, jika berbeda proses selesai.
3. Buatlah node dari titik awal dan *set parent*nya dengan nilai null lalu masukan ke dalam *linked list*
4. Periksa apakah node tersebut sama dengan titik tujuan, jika ya maka proses selesai dengan mengembalikan nilai node tersebut.

5. Periksa node tersebut apakah *parentnya* tidak sama dengan null dan tidak sama dengan ruang kosong (tidak bisa dilalui), jika benar maka tahap berikutnya tidak di proses dan dilanjut ke node berikutnya.
6. Periksa apakah node.x > -1atau node.x < lebar papan atau node.y > -1 atau node.y < tinggi papan, jika ya masuk tahap berikutnya, jika tidak dilanjut ke node berikutnya.
7. Periksa apakah jumlah garis pada *parent* tersebut sudah 3 garis, Jika sudah maka dilanjut ke arah node berikutnya.
8. Bangkitkan semua suksesor pada node tersebut dengan cara mengunjungi setiap arah (*left, up, right, down*).
9. Masukan semua suksesor / *child* tersebut kedalam *linked list* dalam bentuk node dimana parentnya adalah node sebelumnya, dengan syarat node tersebut belum pernah dilalui.
10. Ulangi langkah 4 – 9. Berdasarkan contoh yang terdapat pada analisis kasus, maka pemecahan masalah dengan algoritma *QuadTree* untuk pencarian jalan dengan titik awal (1,1) dan titik tujuan (1,3) dalam pohon pencarian.

Dengan menggunakan algoritma *collition detection* dengan model *quad tree* menjalankan *game* ini berjalan dengan baik.

Arsandi Dkk (2012) melakukan penelitian tentang visualisasi gerakan objek 3D pada *augmented reality* dengan deteksi tumbukan berbasis *bounding box* meneliti didalam nya menggunakan AR dalam menerapkan implementasi *collision detection* didalam nya, *Augmented Reality* (AR) merupakan bidang penelitian komputer yang menggabungkan data computer grafis 3D dengan dunia

nyata. Semakin berkembangnya AR membuat teknologi ini banyak dicari, Penelitian pada dasarnya yang diajukan dalam membuat visualisasi gerak 3D pada AR dengan metode *collision detection* berjalan dengan baik. Tekstur yang selama ini menjadi permasalahan dalam *ARToolKit* dapat terselesaikan dengan mengubah maping tekstur pada objek 3D menjadi berformat bmp 24 bit. Agar lebih terlihat meyakinkan, AR harus mampu menghadirkan interaksi secara realistis. Dalam sistem ini, interaksi *keyboard* diterapkan dengan melakukan animasi *sequence* pada objek. Animasi dipandu dengan *script action txt* untuk dapat menjalankan urutan-urutan pose 3D yang nantinya akan beranimasi. Pergerakan objek menerapkan algoritma *Euclidean* berhasil membawa perubahan posisi objek dalam media AR. Sedangkan untuk *collision detection*, metode sederhana *bounding box* diajukan sesuai dengan tujuan untuk mendeteksi tumbukan antar dua buah objek 3D. Dengan demikian pendeteksian AR mengacu pada tubrukan dengan *collision detection* sangat berguna.

Putrady (2011) melakukan penelitian tentang *Optimasi Collision Detection* Menggunakan *Quadtree*, *Collision Detection* adalah suatu topik yang membahas tentang bagaimana cara mengetahui objek-objek apa saja yang bersentuhan satu sama lain dalam bidang koordinat 2 dimensi ataupun 3 dimensi. C*ollision detection* digunakan untuk mengetahui apakah peluru lawan mengenai pemain. *Brute force* adalah suatu bentuk algoritma yang lempang (*straight forward*) dalam memecahkan suatu masalah. *Brute force* biasanya didasarkan pada pernyataan masalah dan pemecahan masalah secara intuitif ataupun sesuai konsep *Collision detection* menggunakan *brute force*, seperti yang telah dibahas oleh peneliti, membutuhkan waktu komputasi yang semakin meningkat seiring dengan

bertambahnya jumlah objek yang akan dicek. Untuk mengurangi jumlah pengecekan, kita harus sebisa mungkin melakukan pengecekan hanya pada objek-objek yang mungkin bersentuhan sehingga objek-objek yang sudah pasti tidak bersentuhan tidak perlu dicek. *Quadtree* adalah salah satu metode yang menggunakan konsep ini dengan ini peneliti membedaan *collision detection* dengan *quadtree* dan *brute force.*

Ganni dkk (2013) melakukan penelitian tentang perancangan game *live for gym, Game* ini menceritakan tentang seseorang yang mampu mewujudkan mimpinya untuk memiliki sebuah tempat *fitness* yang dirintisnya dari nol. Tempat *fitness* ini menyediakan sejumlah alat *fitness* yang dapat digunakan oleh setiap pemain. Game ini dirancang dengan harapan mampu memberikan sedikit gambaran kepada pemain dalam membuka usaha tempat *fitness* dalam hal melayani pelanggan dengan baik, melalui pengaturan waktu menempatan pelanggan dengan cepat namun sempurna agar pelanggan yang datang tidak segan untuk berkunjung kembali ke tempat *fitness*. Simulasi ini mencoba meniru berbagai kegiatan *fitness* layaknya *fitness* di dalam kehidupan nyata. Di mana game ini telah disesuaikan dengan hasil *survey* yang di lakukan di *Celebrityfitness*.dalam game, game tersebut menggunakan *collision detection* dengan metode menjadi dua jenis metode, yaitu menggunakan *hitTestObject* dan *hitTestPoint*, diantaranya adalah :

1. Metode *hitTestObject*, metode ini digunakan untuk memberi perintah ketika movieclip terjadi tabrakan dengan movieclip lain
2. Metode *hitTestpoint*, metode ini digunakan untuk mendeteksi objek dalam ruang suatu gambar. Berbeda dengan *hitTestObject*, metode ini

mendeteksi point yang ada di suatu ruang. Method ini sangat diperlukan dalam pembuatan game

Reynilda Dkk (2013) melakukan penelitian tentang perancangan *game education* kehamilan berbasis multimedia yang didalam nya memberikan informasi bagi ibu hamil maupun orang-orang disekitarnya mengenai tahap kehamilan. Perancangan dibuat dengan menggunakan perangkat lunak *Adobe Flash*. Hasil dari pengujian program *Game Education* Kehamilanku adalah keseluruhan modul dan fungsi pada program telah dapat berjalan dengan baik, tutorial pada program dapat dimengerti dengan mudah oleh pemain. Berdasarkan *beta testing*, informasi mengenai kehamilan telah dapat tersampaikan dengan baik, secara keseluruhan, *game play* dalam permainan sudah cukup baik. Metode yang di gunakan dalam game ini adalah dua *collision detection function*, yaitu *HitTestPoint* dan *HitTestObject. HitTestObject* akan memeriksa apakah kedua obyek tersebut bersinggungan. Sedangkan sedikit lebih sulit dari *HitTestObject, HitTestPoint* memeriksa lokasi sebuah titik untuk menentukan apakah titik tersebut bersinggungan atau tidak. Perancangan *Game Education* kehamilanku berbasis multimedia dengan menggunakan bahasa pemrograman *Adobe Flash CS6*, *ActionScript* 3.0 dan system operasi windows 7 dengan tampilan grafis *Corel Draw* X5. Permainan ini dibagi dalam 40 minggu (karena durasi kehamilan adalah 9 bulan 10 hari), dan pada tiap minggu pemain harus menyelesaikan salah 1 dari 5 *gameplay* yang tersedia, sesuai dengan jadwal pada tabel yang disediakan di dalam game tersebut.

Agus Dkk (2010) melakukan penelitian tentang transformasi linier dalam game animasi untuk pembelajaran persamaan kurva dibahas hasil pembuatan

game 2 dimensi yang dapat menampilkan visualisasi pergerakan objek mengikuti sebuah persamaan kurva dengan menggunakan metode transformasi linier. Game yang dibuat terdiri dari 3 level yaitu level pertama berupa game penentuan persamaan garis lurus, level kedua berupa game penentuan persamaan kurva sinus, dan level ketiga berupa game penentuan persamaan kurva cosinus. Masing-masing tahap mempunyai 10 sublevel yang harus diselesaikan untuk dapat menuju level berikutnya. Dari hasil penerapan metode transformasi linier dalam game ini diketahui bahwa kecekungan persamaan kurva mempengaruhi kecepatan pergerakan animasi objek yang mengikuti persamaan kurva tersebut semakin cekung persamaan kurva, semakin bertambah cepat pergerakan objek tersebut. Game yang dibuat terdiri dari 3 level yaitu level pertama berupa game penentuan persamaan garis lurus, level kedua berupa game penentuan persamaan kurva sinus, dan level ketiga berupa game penentuan persamaan kurva cosinus. Masing-masing tahap mempunyai 10 sublevel yang harus diselesaikan untuk dapat menuju level berikutnya. Pada level satu, dibuat sebuah fungsi yang dapat menerima dua masukan dua paramater m (gradien) dan c (perpotongan dengan sumbu y). Berikutnya dibuat visualisasi garis persamaan fungsi y = mx + c, yang diikuti oleh pergerakan objek-objek (berupa serangga) sesuai garis tersebut. Objek-objek akan bergerak dari kiri ke kanan sampai keluar batas area utama game. Pada level dua, dibuat sebuah fungsi yang dapat menerima tiga masukan parameter a,b, dan c untuk membentuk persamaan y = a + b.sin cx. Berikutnya dibuat visualisasi kurva dari persamaan tersebut. yang diikuti oleh pergerakan objek-objek (berupa serangga) sesuai kurva tersebut. Objek-objek akan bergerak dari kiri ke kanan sampai keluar batas area utama game. Pada level dua, dibuat sebuah fungsi yang

dapat menerima tiga masukan parameter a,b, dan c untuk membentuk persamaan y = a + b.cos cx. Berikutnya dibuat visualisasi kurva dari persamaan tersebut. yang diikuti oleh pergerakan objek-objek (berupa serangga) sesuai kurva tersebut. Objek-objek akan bergerak dari kiri ke kanan sampai keluar batas area utama game didalam nya.

Maulana Dkk (2011) melakukan penelitian tentang penggunaan struktur data *Quad-Tree* dalam *Algoritma Collision Detection* pada *Vertical Shooter Game* dengan mengimplementasi struktur data *quad-tree* pada algoritma *collision detection* menggunakan strategi *divide* and *conquer*. Peneliti menguji mengkhususkan untuk implementasi pada permainan dengan genre *vertical shooter game*. Pengujian dilakukan menggunakan tiga parameter uji, yaitu kedalaman *Quad-tree*, banyaknya objek spasial yang dicek dan ukuran ruang spasial yang dipartisi oleh *Quad-tree*. Penulis berkesimpulan struktur data ini sangat efisien mengindeks objek-objek spasial pada *vertical shooter* game yang memiliki jumlah objek spasial dibawah 1000. Selain itu, struktur data ini juga umum digunakan untuk *Graphic Information Systems* dan aplikasi *layouting* penelitian tersebut mengungkapkan, Pada *vertical shooter game,* pendeteksian yang digunakan adalah *post detection* karena tumbukan baru ditangani setelah tumbukan tersebut terjadi. Namun, jika game tersebut memiliki AI (*Artificial Intelligence*) maka pendeteksian harus berjenis *priori detection* agar AI dapat bereaksi agar terhindar dari tumbukan. Algoritma pendeteksian tumbukan menggunakan *Quadtree* ini adalah *algoritma indexing*. Jadi, sudah jelas kolisi dideteksi setelah objek spasial diindeks. Karena sudah diindeks, artinya objek sudah bertumbukan.

Uraian penelitian diatas sehingga peneliti menyimpulkan ada beberapa perbedaan yang akan di jelaskan pada tabel 2.1 di bawah :

Tabel 2.1 Perbedaan Penelitian Terdahulu

No	Metode / Algoritma	Metode/ Algoritma Tambahan	Penulis	Kelebihan	Kekurangan	Spesifikasi Penelitian
1.	• Bounding box collision detection	• Algoritma Artificial Bee Colony	• Teguh Dkk (2013)	• Pada simulasi game ini metode *bounding box collision detection* berjalan dengan baik • Grombolan simulasi ikan pada game ini sangat lancer serta tanpa hambatan • Dalam kasus pergerakan segerombolan ikan menuju sumber makanan menggunakan algoritma *Artificial Bee Colony* (ABC), hal-hal yang mempengaruhi nilai fitness terbaik adalah jumlah ikan, maksimum iterasi, nilai batas, dan nilai random	• Ikan dalam simulasi tidak leluasa mengikuti makanan yang di berikan	
2.			• Darmawan (2014)	• Pada game ini efek dan gerakan untuk metode *collision detection* sangat baik • permainan dengan jenis *endlessrunning* ini sangat diminati oleh banyak user terutama dari kalangan yang berusia 13 – 40 tahun.	• Gambar untuk *background/layer* terjauh diberikan efek blur agar dapat mensimulasikan efek perbedaan titik fokus mata. • Gerakan terpaku dalam satu jenis *endlles game* • Pemeberian *score* serta mencapaian tidak di temukan pada game ini	
3.			• Arsandi Dkk (2012)	• Metode yang diajukan dalam membuat visualisasi gerak 3D pada AR dengan metode *collision detection* berjalan dengan baik • Pergerakan objek menerapkan *algoritma Euclidean* berhasil membawa perubahan posisi objek dalam media AR • *collision detection*, metode sederhana bounding box diajukan sesuai dengan tujuan untuk mendeteksi tumbukan antar dua buah objek 3D.	• AR di gunakan dengan system operasi windows saja	• Ar toolkit
4.	• Collision Detection	• Boids	• mufiroh Dkk (2014)	• Pada game ini menjalan kan algoritma *collisiondetection* berjalan sangat dinamis dan baik • game ini tidak hanya bergantung pada matinya musuh namun juga tergantung timer untuk	• Tidak dapat dimainkan dengan jaringan internet • Terlalu menentukan kapasitas spesifikasi mobile yang di gunakan mainkan nya	• Pemrogr aman menggunak an java • Android

				memenangkan game ini, musuh dapat menyerang pemain, dan dapat menyimpan *score*		
No	Metode / Algoritma	Metode/ Algoritma Tambahan	Penulis	Kelebihan	Kekurangan	Spesifikasi Penelitian
5.		• *Quad Tree*	• Susilawati (2014)	• Perangkat lunak ini memungkinkan pemakai (*user*) komputer untuk memainkan permainan *spaceship* tanpa harus berhadapan secara langsung. • Perangkat lunak dapat dimainkan oleh beberapa orang yang terkoneksi dalam jaringan dengan masing-masing dua orang saling berhadapan • Perhitungan game dengan *collision detection* berjalan dengan baik.	• Konsep star ship space nya kurang maksimalkan karakter pada game tersebut • Hanya di mainkan di computer dengan system operasi windows	• Visual basic • windows
6.		• *Quad Tree* • *Brute force*	• Putrady (2011)	• Algoritma *brute force* dapat dilakukan untuk menyelesaikan masalah *collision detection* • Optimasi pada *collision detection* dapat dilakukan dengan hanya mengecek objek-objek mana saja yang masih mungkin bersentuhan.	• Algoritma *brute force* memakan waktu komputasi yang besar • Penyelesaian *Quad Tree* terlalu lama untuk komputasi pada objek	
7.		• hitTestObject • hitTestPoin	• Ganni dkk (2012)	• Game ini memiliki tampilan menu utama yang menarik dan penjelasan tutorial yang mudah dimengerti • Game ini memiliki control dan user interface yang mudah digunakan • Penggunaan algoritma *collision detection* berjalan baik.	• Permainan sedikit sulit pada saat melakukan permainan menggunakan typing • Tampilan alat *fitness* yang biasa-biasa saja.	• Action Script 3
8.			• Wellan dkk (2013)	• Modul serta game nya berjalan dengan baik • Informasi yang diberikan dalam *game education* Kehamilanku telah dapat menambah pengetahuan responden seputar hal-hal umum pada proses	• Game tidak berjalan dalam multi player • Penggunaan game hanya pada system oprasi windows	• Adobe Flash CS5.5 • Corel Draw X5
9.			• Nugroho (2011)	• Game dapat mengacak nilai-nilai koefisien dan konstanta dari persamaan garis yang • Game dapat memberikan waktu yang tidak jauh berbeda meskipun persamaan yang ditampilkan berbeda • Implementasi algoritma *collision detection* sangat tepat dan berjalan baik	• Beberapa pergerakan animasi objek terlihat kurang halus terutama jika waktu delay objek dibuat terlalu tingg • Game hanya dapat menentukan waktu pergerakan animasi dengan tingkat akurasi 1 detik	

No	Metode / Algoritma	Metode/ Algoritma Tambahan	Penulis	Kelebihan	Kekurangan	Spesifikasi Penelitian
10.			• Maulana (2010)	• performa algoritma *collision detection* menggunakan *divide* and *conque*r dan struktur data *Quad-Tree* sangat baik • Untuk *game vertical shooter* biasa, jumlah objek spasial rata-rata tidak melebihi seribu buah • Jika pendeteksian 5000 objek dengan ketelitian 1x1 pixel saja bisa dilakukan dengan sangat efisien, maka algoritma ini bisa sangat efisien juga untuk aplikasi *layouting* seperti browser ataupun CAD.	• untuk pengecekan lebih lanjut untuk geometri yang lebih rumit dapat dilakukan lebih lanjut setelah mendapat list objek spasial dalam segmen yang dicek tersebut • implementasi pada game tersebut kurang memadai dalam algoritma	

2.2 Perbedaan Penelitian yang terdahulu

Dengan mengacu penelitian yang sudah ada, peneliti menyimpulkan tentang implementasi *collision detection* pada permasalahan penelitian sekarang ini. Peneliti ingin menerapkan collision detection pada *game fly bee*, game yang berbasis system oprasi android ini di buat dengan aplikasi *game maker studio*, dalam *game fly bee* ini menggunakan karakter *bee* atau lebah didalamnya pemain akan dibawa untuk mengendalikan seekor lebah untuk menghindari musuh seperti pipa, setiap melewati pipa atau penghalang tersebut pemain mendapatkan *score*. Permaianan ini menggunakan system gugur pada saat *bee* atau si lebah menabrak penghalang atau balok pipa yang berjalan. Musuh akan di berposisi secara acak dan akan memberikan celah yang bisa di lewati lebah sedangkan permainan kembali ke awal saat menabrak pepohonan, pemaian akan kembali ke start game dengan score yang telah di capai saat gugur dan terdapat penampilan *score* yang dicapai pemain. Metode *collision detection* pada game ini saat *bee* tersebut menabrak penghalang pipa yang berjalan untuk medeteksi si lebah dan pipa tubrukan.

BAB III

LANDASAN TEORI

3.1 Pengertian Android

Android adalah sistem operasi berbasis*Linux* yang di buat untuk mengaplikasikan perangkat mobile dan tablet. Pertama kali android berawal dan dikembangkan oleh *Android, Inc*., dengan di dukung keuangan oleh *Google*, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007. Ponsel Android pertama mulai dijual pada bulan Oktober 2008. Sebenarnya, versi komersil pertama dari Android adalah Android 1.0, OS ini dirilis pada bulan September 2008 dan versi beta yang dirilis pada November 2007 (*https://id.wikipedia.org/wiki/Android*). Android merupakan pengembangan berkelanjutan oleh Google dan Open Handset Alliance (OHA) dan telah melakukan beberapa kali update sejak sistem operasi ini dirilis.

Sejak diakuisisi *Google* OS Android berkembang dan menduduki OS terlaris yang di pakai saat ini, ada beberapa nama atau kode urutan android dari versi pertama sampai terkini,android di beri kode berdasarkan nama *dessert* (pencuci mulut) dan diurut berdasarkan alfabet, seperti *Cupcake* (Android 1.5), *Donut* (Android 1.6), *Eclair* (Android 2.0 – 2.1), *Froyo* (Android 2.2 – 2.2.3), *Gingerbread* (OS Android 2.3–2.3.7), *Honeycomb* (Android 3.0–3.2.6), *Ice Cream Sandwich* (Android 4.0–4.0.4), *Jelly Bean* (Android 4.1–4.3), *KitKat* (Android 4.4), *Lollipop* (Android 5.0). dan terakhir ini merilis *Marshmallow* (Android 6.0) dan yang akan datang versi N belum di tentukan. perkembangan sistem operasi Android sejak diperkenalkan ke publik pada tanggal 5 November

2007, sampai sekarang OS ini masih teratas yang di sukai *user mobile* saat ini, pada penjelasan di atas akan di jelaskan kelangkapan pada Tabel 3.1 di bawah :

Tabel 3.1 Versi Android

Versi	Nama Kode	Level API
6.0	Marshmallow	23
5.x	Lollipop	21
4.4.x	KitKat	19
4.3.x	Jelly Bean	18
4.2.x	Jelly Bean	17
4.1.x	Jelly Bean	16
4.0.3–4.0.4	Ice Cream Sandwich	15
3.2	Honeycomb	13
3.1	Honeycomb	12
2.3.3–2.3.7	Gingerbread	10
2.3–2.3.2	Gingerbread	9
2.2	Froyo	8
2.0–2.1	Eclair	7
1.6	Donut	4
1.5	Cupcake	3

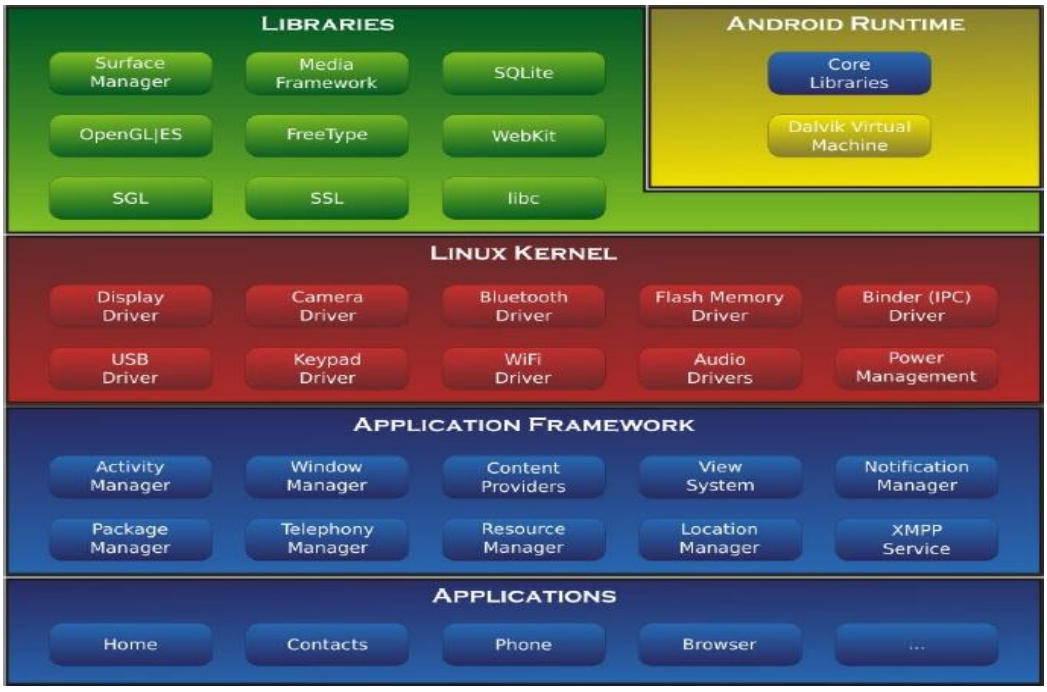
Pada Tabel 3.1 terdapat perbedaan API di setiap pembaruan android terdapat tingkatan API. API (*application programming interface*) atau dalam bahasa Indonesia berarti Antarmuka Pemrograman Aplikasi adalah sekumpulan fungsi, perintah maupun protokol khusus yang dibutuhkan oleh seorang programer untuk membangun sebuah Aplikasi atau Software.

Keamanan dan privasi pada sistem, Aplikasi Android berjalan di *sandbox*, sebuah area terisolasi yang tidak memiliki akses pada sistem, kecuali izin akses yang secara eksplisit diberikan oleh pengguna ketika memasang aplikasi. Sebelum

memasang aplikasi, *Play Store* akan menampilkan semua izin yang diperlukan, misalnya : sebuah permainan perlu mengaktifkan getaran atau menyimpan data pada Kartu SD, tapi tidak perlu izin untuk membaca SMS atau mengakses buku telepon. Setelah meninjau izin tersebut, pengguna dapat memilih untuk menerima atau menolaknya, dan bisa memasang aplikasi hanya jika mereka menerimanya (*https://id.wikipedia.org/wiki/Android*).

3.2 Arsitektur Sistem Operasi Android

Sistem operasi android memiliki beberapa bagan arsitektur,bagan paling bawah pada struktur Android adalah kernel. *Google* menggunakan kernel Linux versi 2.6 untuk membangun Android yang meliputi *memory management*, *security setting, power management* dan beberapa *driver hardware*.*library* yang digunakan adalah library java. Lapisan selanjutnya adalah *application framework* yang mencakup program untuk mengatur fungsi- fungsi dasar smartphone.pada Gambar 3.1 di jelaskan bagan arsitektur pada system operasi android.



Gambar 3.1 Arsitektur Android (Arifan, 2015)

3.3 Lima Struktur Android

a. *Linux kernel*

Android menggunakan kernel yang berbasis *kernel Linux*, Namun secara keseluruhan android bukanlah linux, karena dalam android tidak terdapat paket standar yang dimiliki oleh linux lainnya. Linux merupakan sistem operasi terbuka yang handal dalam manajemen memori dan proses. Oleh karenanya pada android hanya terdapat beberapa servis yang diperlukan seperti keamanan, manajemen memori, manajemen proses, jaringan dan driver. Kernel linux menyediakan *driver* layar, kamera, *keypad, WiFi, Flash Memory, audio*, dan IPC (*Interprocess Communication*) untuk mengatur aplikasi dan celah dalam keamanan.

b. *Library*

Android menggunakan pustaka yang terdapat Bahasa C/C++ dengan standar *Berkeley Software Distribution* (BSD) dan sebagian didalamnya tidak mencakup semua dari yang aslinya untuk tertanam pada kernel Linux. Beberapa pustaka diantaranya:

1) *Media Library* hanya dipergunakan untuk memutar format multimedia
2) *Surface Manager* pengaturan hak akses pada aplikasi didalamnya
3) *Graphic Library* terdapat didalamnya SGL dan *OpenGL SQLite* untuk mengatur relasi database yang digunakan pada aplikasi.
4) *SSl dan WebKit untuk browser dan keamanan internet.*

c. *Android runtime*

Android Runtime merupakan mesin virtual yang membuat aplikasi android menjadi lebih tangguh dengan paket pustaka yang telah ada.

Dalam *Android Runtime* terdapat 2 bagian utama, diantaranya: pustaka inti dan mesin *virtual davisk*.

d. Application framework

Pada *system android* terdapat kerangka aplikasi yang menyediakan kelas-kelas, yang dapat digunakan untuk mengembangkan aplikasi android. Selain itu, juga menyediakan abstraksi generik untuk mengakses perangkat, serta mengatur tampilan *user interface* dan sumber daya aplikasi

e. Application layer

Aplikasi merupakan lapisan yang paling tampak pada pengguna ketika menjalankan program. Pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam Android runtime dengan menggunakan kelas dan *service* yang tersedia pada *framework* aplikasi.

3.4 Pengertian Game

Game merupakan sebuah permainan dimana permain berakhir dengan menang ataupun kalah. Di dunia teknologi sendiri *game* adalah sebuah aplikasi permainan yang terpasang di alat elektronik, perkembangan pesat oleh jumlah pengguna *mobile game* diindonesia yang semakin meningkat setiap tahunnya. Kemajuan teknologi yang ada membuat game menjadi andalan bagi seseorang untuk mengisi keseharian hidup. Game juga media pembelajaran dan peningkatan dalam ketangkasan serta bisa membantu untuk pendidikan dalam pembelajaranpelajaran seperti permaianan catur atau tebak sejarah. Permainan merupakan bagian dari bermain dan bermain juga bagian dari permainan

keduanya saling berhubungan. Permainan adalah kegiatan yang didalamnya terdapat peraturan, play dan budaya. Sebuah permainan adalah sebuah sistem dimana pemain terlibat dalam konflik buatan, disini pemain berinteraksi dengan sistem dan konflik dalam permainan merupakan rekayasa atau buatan, pada permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan. Game bertujuan untuk menghibur, biasanya game banyak disukai oleh anak - anak hingga orang dewasa.

Pada dasarnya Game adalah permainan yang menggunakan media elektronik, merupakan sebuah hiburan berbentuk multimedia yang dibuat semenarik mungkin agar pemain bisa mendapatkan sesuatu sehingga adanya kepuasaan batin. Bermain game merupakan salah satu sarana pembelajaran. Game edukasi dibuat dengan tujuan spesifik sebagai alat pendidikan, untuk belajar mengenal warna, mengenal huruf dan angka, matematika, sampai belajar bahasa asing dan pergkembangan game melalui banyak generasi Game pertama di dunia 1972, yaitu sebuah permainan elektronik yang menampilkan gambar bergerak (video). Sebuah perusahaan bernama *Magnavox* meluncurkan video game pertama, yaitu *Odyssey*. *Magnavox Odyssey*, konsol game pertama di dunia mengoperasikan Pong (*http://en.wikipedia.org/wiki/Game*).

3.5 Elemen Penyusun Game

Pada*game* ada beberapa jenis *game*, karakter game, background, dan musik. Hal tersebut saling mendukung dan sangat penting dalam sebuah *game* dengan tujuan supaya *user* tertarik dan tidak membosankan.sehingga pemain dapat menghilangkan kebosanan dalm memainkannya.

3.5.1 Genre /Jenis Game

Perkembangan game semakin maju sehingga ada berbagai jenis /genre dalam permaian game sebagai penjelasan beberapa yang di ketahui :

a. Shooting

Game ini adalah jenis game yang mememerlukan ketangkasan dalam menembak, serta ketrampilan dalam mengolah tembakan dan akursi bidikan

b. Adventure

Game yang bertema perjalanan serta mengarungi sebuah misteri atau tekateki di dalam perjalanan nya

c. Arcade / Side Scrolling Game

Arcade game adalah genre game yang tidak terfokus pada cerita, melainkan hanya dimainkan untuk kejar-mengejar *point / highscore*.

d. Fighting

Fighting adalah genre game bertarung. Seperti dalam arcadegame ini mengadopsi pertarungan dan nilai score, *Genre fighting* biasanya *one on one* dalam sebuah arena yang sempit.

e. Sport

Sports game adalah salah satu genre game yang di buat dari olahraga yang ada di kehidupan nyata. Dari pemain, wasit, stadion sampai strategi permainan di perhatikan disini. Tokoh dari game ini biasanya benar-benar ada di dunia nyata. Tapi ada juga yang merupakan hasil kreasi pembuat game. Sistem permainan tiap game berbeda tergantung jenis olahraga dan yang membuat.

f. Racing

Game yang memiliki elemen-elemen yang mengikuti perkembangan otomotif didunia, dengan tampilan yang real yang kita dapatkan saat bermain game berjenis *racing*. Pemain akan memainkan layaknya menjadi seorang rider, yaitu melakukan balapan untuk merebut posisi pertama.

g. Simulation

Game dengan penggambaran konsep permainan dengan segala sesuatu yang nyata pada kehidupan dan memperhatikan detail berbagai faktor.

h. RTS (*Real Team Strategy*)

Game yang memerlukan strategi dalam memainkannya, kebanyakan *game* RTS adalah *game* strategi perang.

i. RPG (*Role Playing Game*)

adalah game yang para pemainna memainkan peran tokoh-tokoh khayalan dan berkolaborasi untuk merajut sebuah cerita bersama, didalam game ini biasanya terdapat unsur seperti experience point, atau perkembangan karakter yang kita mainkan sehingga membuat karakter kita naik level dan semakin kuat. Unsur cerita dalam game RPG sangat kental.

j. Puzzle

Pemain harus menyelesaikan teka-teki dalam game tersebut dimana pemain menata ulang yang di acak.

3.5.2 Karakter Game

Pada permaianan ada beberapa karakter yang dimainan dalam sisi ini terbagi menjadi karakter pahlawan dan karakter musuh, setiap karakter mewakili perannya sehingga alur game menjadi menarik untuk dimainkan.

3.5.3 Background

Background adalah tampilan latar pada sebuah permainan yang di buat untuk menghidupkan latar pada game agar tetap menyerupai kenyataan .

3.5.4 Musik

Pada sebuah *game* memiliki bagian musik yang melatarbelakangi untuk mengiringi permaianan saaat di jalankan, music dibagi beberapa tempat pemutaran awal permaianan dan perjalanan permaian atau di tengah permainan.

3.6 Collision Detection

Collision detection adalah metode yang di gunakan dalam menentukan pergerakan dalam game sperti tubrukan suatu objek. Dengan menggunakan algoritma ini dapat menetukan arah serta tujuan dalam membuat game agar dapat mudah dinikmati, metode ini juga membuat dunia game semakin mirip dengan kenyataan di dunia nyata. Algoritma *collision detection* adalah proses pengecekan apakah beberapa buah objek spasial, saling bertumpuk atau tidak. Jika ternyata ada paling sedikit dua buah objek yang bertumpuk, maka kedua objek tersebut dikatakan saling bertumpukkan. Pada ruang spasial dua dimensi. Objek yang bertumpuk berarti objek spasialnya beririsan (Nugraha, 2013).

Menurut Arsandi dkk (2012), *collision detection* adalah proses pendeteksian tabrakan antara dua objek, tabrakan tidak hanya terjadi antara dua

objek, tetapi dapat juga terjadi antara satu objek dengan banyak objek sehingga dibutuhkan *collision detection* yang akurat.

Collision detection juga berguna untuk menentukan posisi dari satu objek dengan objek yang lain agar tidak ada objek yang saling menembus, sehingga simulasi yang akan dibuat memiliki kesamaan dengan realita yang ada, contoh beberapa jenis *algoritma collision detection*

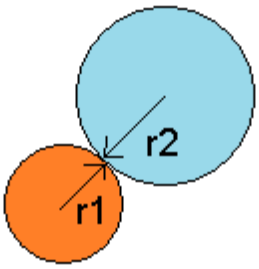
a. *Bounding Rectangle/Box Collisions*

Metode ini, deteksi tabrakan akan memeriksa objek bertabrakan vertikal dan horizontal dan membuat perubahan dalam segi pergerakan yang sesuai. Tergantung pada arah tumbukan pada objek lain, objek dapat diangkat ke atas atau bawah atau mendorong kiri atau kanan.

Gambar 3.2 Algoritma Collision Bounding Box*(http://www.kilobolt.com, 12-06-2016)*

b. *Bounding Circle Collisions*

Metode *bounding box circle collision* akan memeriksa apakah jarak antar 2 titik pusat lingkaran lebih besar dari pada jumlah kedua jari-jari. Jika ya, maka tidak terjadi tumbukan.Jika tidak maka telah terjadi tumbukan (handoyo,2014)



Gambar 3.3 Bounding Circle Collision*(http://gamedevelopment.tutsplus.com*)

BAB IV

METODE PENELITIAN

Bab ini membahas metode *collision detection* dan tahap-tahap yang akan dilakukan dalam perancangan game yang akan dibuat.

4.1 KONSEP GAME

Konsep game pada *game fly bee* ini mengacu pada konsep *endless game*, pada perjalanan sistem nya game ini hanya dijalan kan pada sistemoperasiandroid .

Alur permaianan game ini pemain akan di bawa menuju sebuah rintangan berupa pipa-pipa yang berposisi random, disana terdapat karakter lebah yang akan selalu jatuh kebawah sehingga pemain diharuskan mengetuk layar agar lebah mempertahankan posisinya untuk menghindari tabrakan dengan pipa-pipa yang menghalangi, dengan melewati sela diantara pipa-pipa tersebut maka pemain mendapatkan poin, Aturan yang di buat penulis dalam *game fly bee* ini adalah sebagai berikut :

1. Lebah pemain harus menghindari pipa yng berjalan menghadang lebah jika lebah tertabrak, maka pemain gagal lalu kembali ke semua
2. Apabila lebah pemain melewati sela pipa-pipa akan mendapatkan *score*, setiap pipa di lewati mendapatkan *score* satu poin.
3. Jika gagal pemain akan melihat *score* yang tampil setelah gagal dan kembali lagi pertama kali
4. Point di hitung setiap pemain satu kali main jika gagal, pemain akan memulai dari awal yaitu poin 0.

4.2 ANALISIS KEBUTUHAN

4.2.1 Kebutuhan Perangkat Keras

Adapun kebutuhan perangkat keras yang dibutuhkan untuk menjalankan aplikasi *game fly bee* akan dijelaskan dibawah ini :

a. Xiaomi redmi pro
b. Ram 3 GB
c. Layar 720 x 1280 pixel 5,5 inc
d. CPU : Deca-core 2.11GHz
e. OS : Android OS 6.0 (marshmallow)

4.2.2 Kebutuhan Perangkat Lunak

Perangkat lunak yang di gunakan pada proses pembutan aplikasi *game fly bee* adalah :

1. Microsof Windows 10 64 bit, adalah sistem operasi yang di gunakan dalam membuat game *fly bee*, logo windows 10 akan di jelaskan pada Gambar 4.1.

Gambar 4.1 Logo Windows 10

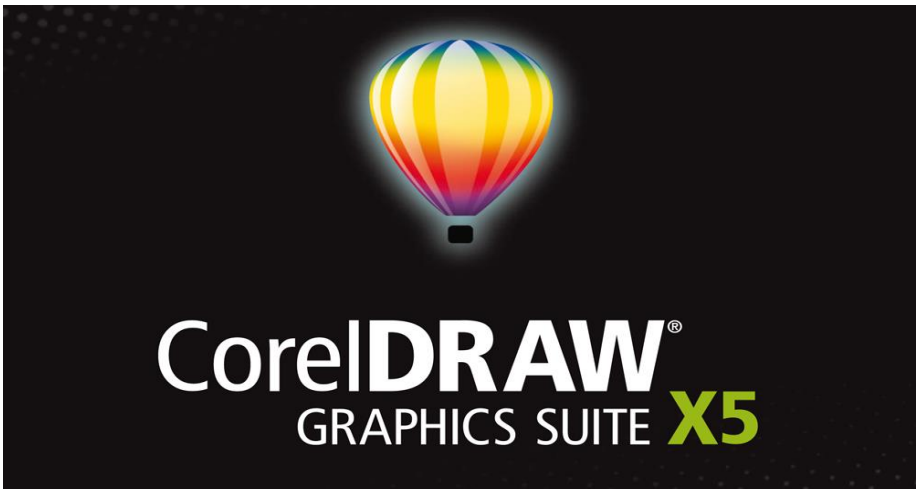
2. *Game maker professional version 1.4*, adalah perangkan lunak atau aplikasi untuk membuat game *fly bee*. Logo *game maker* akan di tampilkan pada Gambar 4.2.

Gambar 4.2 Logo Game Maker V1.4 *(http://www.androidauthority.com*)

3. Sistem operasi android 6.0 marshmallow, merupakan sistem operasi dalam menjalankan game *fly bee*. mengenai logo akan dijelaskan pada Gambar 4.3.

Gambar 4.3 logo android 6.0 marshmallow (http://thunderztech.com)

4. Corel draw x5 aplikasi untuk Membuatan *karakter,background*,serta tulisan pada game tersebut, adapun logo corel draw x5 akan di tampilkan pada Gambar 4.4.



Gambar 4.4 logo corel draw x5

5. Adobe photoshop cs3 aplikasi untuk pembuatan *karakter sprite*, *background* game, logo photoshop cs3 akan ditampilkan pada Gambar 4.5.

Gambar 4.5 Logo Photoshop Cs3

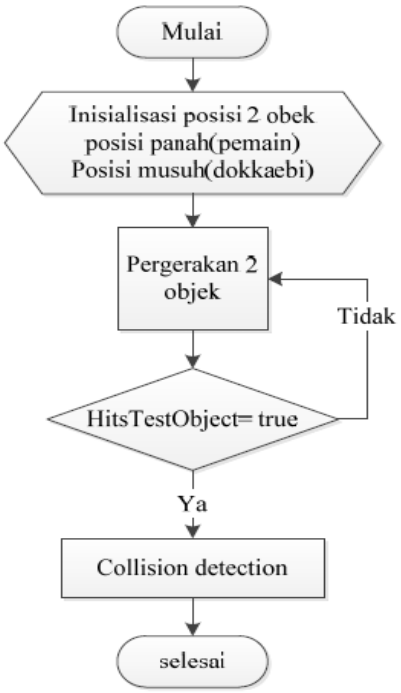
4.3 IMPLEMENTASI METODE COLLISION DETECTION

Permaianan *game fly bee* membutuhkan proses tabrakan antara satu objek dan objek lain, begitu juga game-game pada umumnya. Pada game juga membutuhkan *coliision detection* yang dibutuhkan antara satu objek dan objek lain dengan

kejadian yang berbeda antara yang menambrak dan ditabrak, *collision detection* juga diimplementasikan pada *game fly bee* agar tidak saling menembus antara objek satu dan yang lain sehingga terjadi kesamaan realita.

Mengacu penelitian putrady 2011 bahwa *collision detection membahas* bagaimana cara mengetahui objek-objek apa saja yang bersentuhan dalam bidang koordinat tertentu. Objek-objek ini bisa saja memiliki bentuk yang sangat bervariasi. Objek-objek pada game memiliki bentuk yang bervariasi, ada yang berbentuk kotak, segi-n, sampai bentuk pesawat pemain yang sangat mendetail. Untuk mempercepat proses pada *collision detection*, umumnya objek-objek ini direpresentasikan secara logik dengan bentuk primitif seperti segiempat dan lingkaran (jika pada koordinat dua dimensi), atau kubus dan bola (jika pada koordinat tiga dimensi). Bentuk primitif yang merepresentasikan objek ini biasa disebut sebagai *Bounding Box*atau *Bounding Circle.* Dalam penelitian nya juga dapat di pahami tentang *collision detection* mempresentasikan pada game.

Menurut Lia Musfiroh dkk 2014 pada penelitiannya menerapankan algoritma *collision detection* pada *game dokkaebi shooter diimplementasikan dalam Flowchart Collision Detection* Antara panah dengan musuh *(Dokkaebi)* yang menggambarkan alurdari algoritma *collision detection*berikut Gambar 4.1.



Gambar 4.6 ***Flowchart Collision Detection*** **Antara Panah dengan Musuh *(Dokkaebi)***

Collision detection dimulai dengan mendeteksi posisi tembakkan pemain dan posisi musuh (*dokkaebi*). Setelah dua objek tersebut bergerak dan saling bertumbukkan, persamaan logika akan mendeteksi apakah dua objek tersebut saling bertabrakan atau tidak. Jika persamaan logika tersebut bernilai '*true*', maka *collision detection* akan terjadi dan melanjutkan alur pemrograman ke langkah berikutnya seperti hancurnya *dokkaebi*. Begitu juga jika karakter pemain bertumbukan dengan objek musuh. *Algoritma detection collision* akan dijalankan kembali. Kita dapatkan kesimpulan bahwa *algoritma collision detection* sangat esensial dalam pembuatan sebuah game dan animasi dan jika *collision detection* tidak diterapkan, maka game atau animasi tersebut bisa dibilang tidak berfungsi

Dalam game ini mengimplementasikan *bounding box* berbentuk *bound*, deteksi tabrakan di perlukan objek yang di buat, memiliki *bounding box* setiap objeknya dengan demikian akan menemukan perbandingan irisan (*intersection*)

pada setiap objek dengan kordinat tertentu. Untuk menentukan regional *bounding box* pada objek di tentukan dengan rumusan berikut :

Regional R={(x,y)}| min x ≤ x ≤ max x

min y ≤ y ≤ max y

Di tentukan :

regional R = Regional *bounding box collision*

x,y = titik kordinat y,x

minx,maxy = Nilai minimum kordinat x,y

maxx,maxy = Nilai maximum kordinat x,y

kordinat diatas dapat di lihat pada Gambar 4.7.

Gambar 4.7 Min-max *bounding box*

Dilihat pada Gambar 4.7 pada *Bounding box* dilakukan pengujian terhadap regional- regional *collision detection* yang saling bertabrakan atau tidak, hal tersebut dilakukan sebuah pengujian dengan membandingkan nilai maksimum dan nilai minimum di area x.y, kordinat dua regional akan saling bertabrakan jika keadaan berikut :

AxMin < BxMax dan AMax > Bxmin

AyMin < ByMax dan AyMax > BxMin

Dari yang tersebutkan akan di jelaskan dimana:

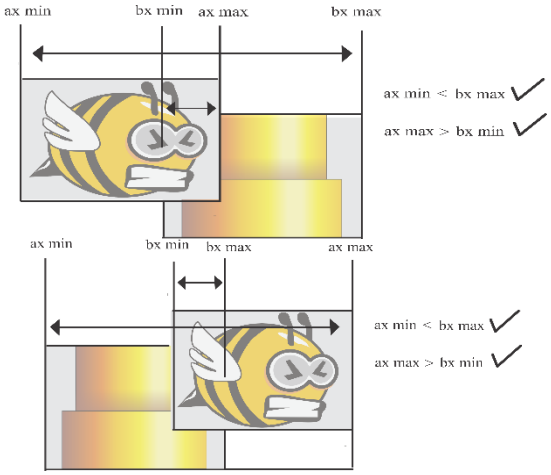
AxMin,AyMin = Nilai minimum kordinat x,y regional A

AxMax,AyMax= Nilai maximum kordinat x,y regional A

BxMin,ByMin = Nilai minimum kordinat x,y regional B

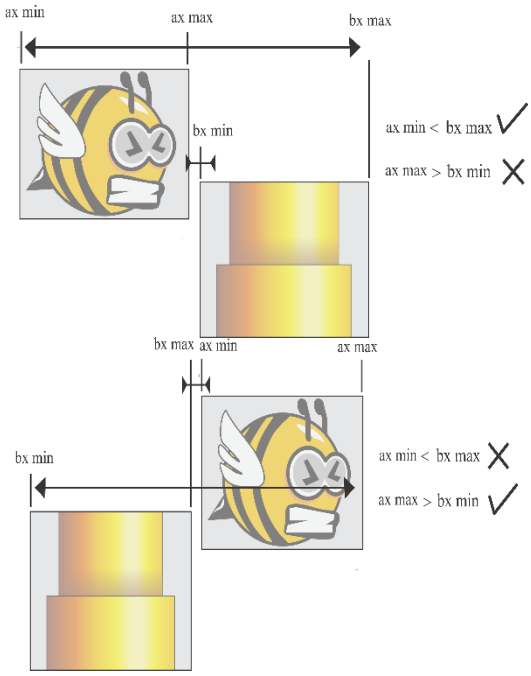
BxMax,MyMax = Nilai maximum kordinat x,y regional B

Dengan melihat rumuan diatas akan di jelaskan terjadinya tebrakan anatar dua bound kotak pada dimensi x akan terlihat pada Gambar 4.8.



Gambar 4.8 *bound* bertabrakan

Jika keadaan tidak memenuhi terjadinya tabrakan antara dua bound kotak pada dimensi x terlihat pada Gambar 4.9.



Gambar 4.9 *Bound* tidak bertabrakan

Game *fly bee* mengimplementasikan *collision detection* terjadi jika semua objek pada karater *fly bee* dibuat *bounding box* dengan *shape precise*, selanjutnya adalah menentukan terjadinya *collision detection* dengan menginisialisasi variabel untuk tata letak kordinat yang tepat untuk menetukan deteksi tabraka antara *be*e dan pipa .

Setelah terjadinya tabrakan pada tahap berikutnya menentukan respon ketika terjadinya tabrakan yang menentukan terjadinya tabrakan objek seakan menjadi realita ada beberapa tahap yang di perlukan yaitu :

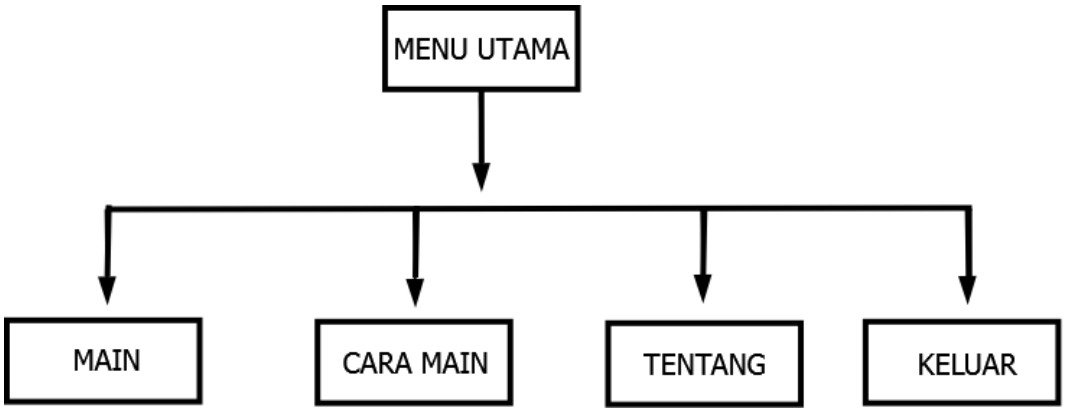
Pertama, permainan *fly bee* jika pada perjalanan game objek *bee* bertabrakan dengan pipa atas maupun bawah maka akan akan terjadi menghilang selanjutnya digantikan efek tabrakan dan karakter *bee* sedih. kedua apabila objek *bee* bertabrakan dengan rumput maka akan digantikan karakter *bee* jatuh ketiga

apabila *bee* melewati objek *garis unviseble* antara pipa atas dan bawah maka objek poin akan di tampilkan.

Pada *bounding box* disetiap objek permaianan *fly bee* untuk medeteksi tabrakan dengan membandingkan antara masing-masing objek sehingga terjadinya *collision detection*.

4.4 RANCANGAN STRUKTUR MENU

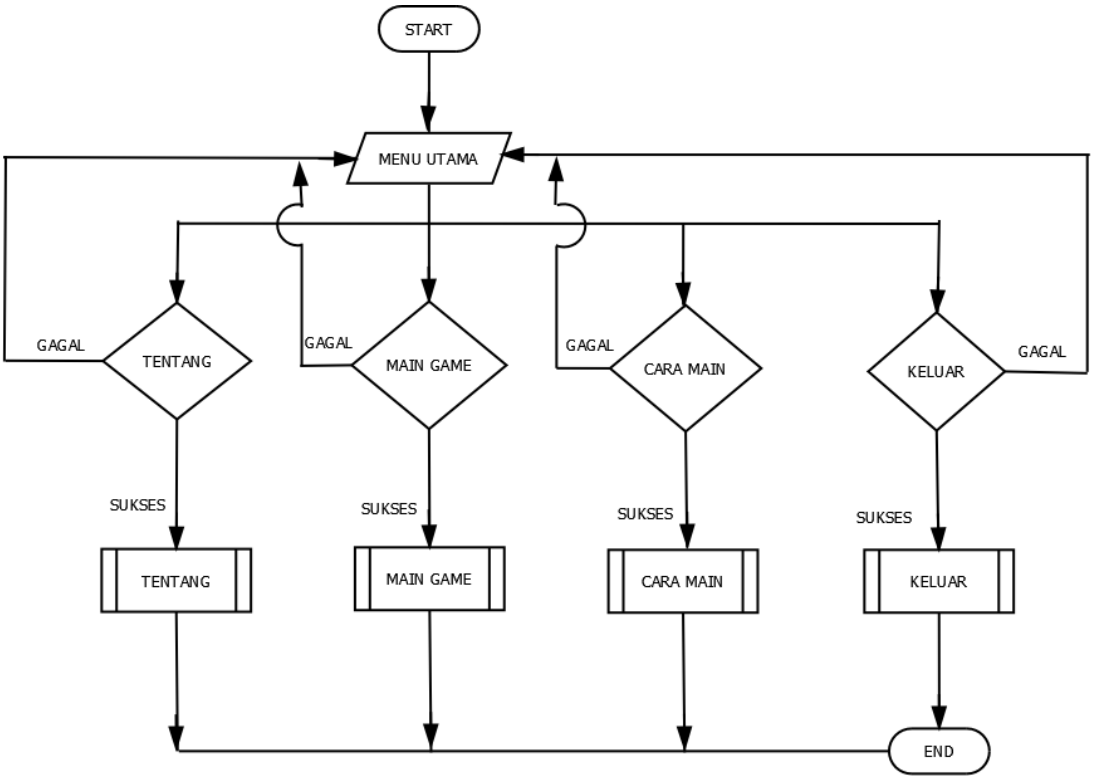
Memahami struktur pada *fly bee* ini, serta memudahkan untuk mengerti menu gambaran struktur pada game secara menyeluruh, adapun gambar struktur *game fly bee* akan di jelaskan pada Gambar 4.10.



Gambar 4.10 *struktur menu*

4.5 RANCANGAN ALGORITMA PROGRAM

*Flowchart*merupakan bagan yang menunjukkan alur kerja atau apa yang sedang dikerjakan di dalam sistem secara keseluruhan dan menjelaskan urutan dari prosedur-prosedur yang ada di dalam sistem. Dengan kata lain, *flowchart* ini merupakan deskripsi secara grafik dari urutan prosedur-prosedur yang terkombinasi yang membentuk suatu sistem.berikut adalah *flowchart* untuk menunjukan alur kerja pada *game fly bee* pada Gambar 4.11.



Gambar 4.11 Flowchart *Game Fly Bee*

Di bawah ini keterangan yang menjelaskan Gambar 4.11 :

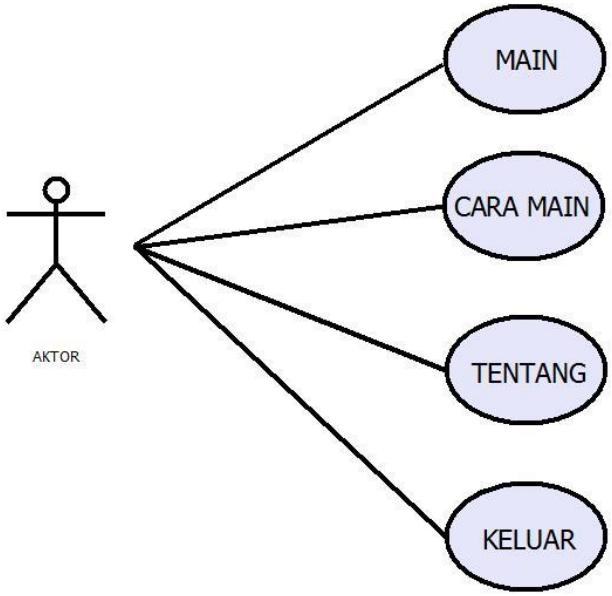
1. Pada menu utama terdapat 4 pilihan
2. Pada pilihan kondisi pertama *tentang,* akan menampilkan *tentang*, pilih kembali untuk menuju menu utama
3. Pada pilihan kondisi kedua *main game* ,akan menampilkan halaman game
4. Pada pilihan kondisi ketiga *cara main,* akan menampilkan *cara main*, pilih kembali untuk menuju menu utama
5. Pada pilihan kondisi keempat *keluar,* akan mengeluarkan dari permainan game

4.6 ANALISIS FUNGSI

Analisis fungsi merupakan proses pemecahan sesuatu kedalam beberapa bagian komponen untuk diidentifikasi dan mengetahui kontribusi masing-masing komponen dalam mencapai suatu tujuan. Analisis fungsi yang dilakukan penulis adalah dengan menggambarkan *use case diagram* dan *activity diagram* terhadap aplikasi game yang akan dibuat.

4.6.1 *Use Case Diagram*

Use case diagram yang digunakan untuk menggambarkan secara ringkas siapa yang menggunakan sistem dan apa saja yang bisa dilakukannya. hal ini akan dijelaskan *use case diagram* yang digunakan pada Gambar 4.12.



Gambar 4.12 *Use case diagram*

Adapun diskripsi masing-masing *use case* berdasarkan *use case diagram* pada Gambar 4.7 akan dijelaskan pada Table 4.1 – 4.4

Tabel 4.1 Deskripsi *Use Case Diagram* Mulai Permainan

Nama	Main game	
Aktor	Pemain	
Deskripsi	Aktor memilih main game untuk bermain	
Exception	Gagal mulai permainan	
Pre Condition	Aktor telah membuka aplikasi game	
Aktor		**Sistem**
Skenario normal		
1. Aktor memilih mulai permainan		2. Membuka koneksi *game*
		3. Menampilkan karakter dan objek pada arena permainan.
4. Aktor masuk ke dalam permainan		
Skenario alternative		
1. Aktor memilih tombol main untuk memulai		2. Membuka koneksi ke aset *game*
3. Aktor membuka aplikasi *game* kembali		
Post Condition		Aktor berhasil masuk ke dalam permainan

Tabel 4.2 Deskripsi *Use Case Diagram* melihat *tentang*

Nama	Melihat Tentang	
Aktor	Pemain	
Deskripsi	Aktor memilih tombol tentang untuk melihat tentang game	
Exception	Gagal melihat tentang	
Pre Condition	Aktor telah membuka aplikasi game	
Aktor		**Sistem**
Skenario normal		
1. Aktor memilih tombol tentang		2. Membuka koneksi ke *game*
		3. Menampilkan informasi tentang game fly bee
Skenario alternative		
1. Aktor memilih tombol tentang		2. Membuka koneksi ke aset *game*
3.memilih tombol keluar untuk kembali		
Post Condition		Aktor berhasil mengakses tentang

Tabel 4.3 Deskripsi *Use Case Diagram* melihat *cara main*

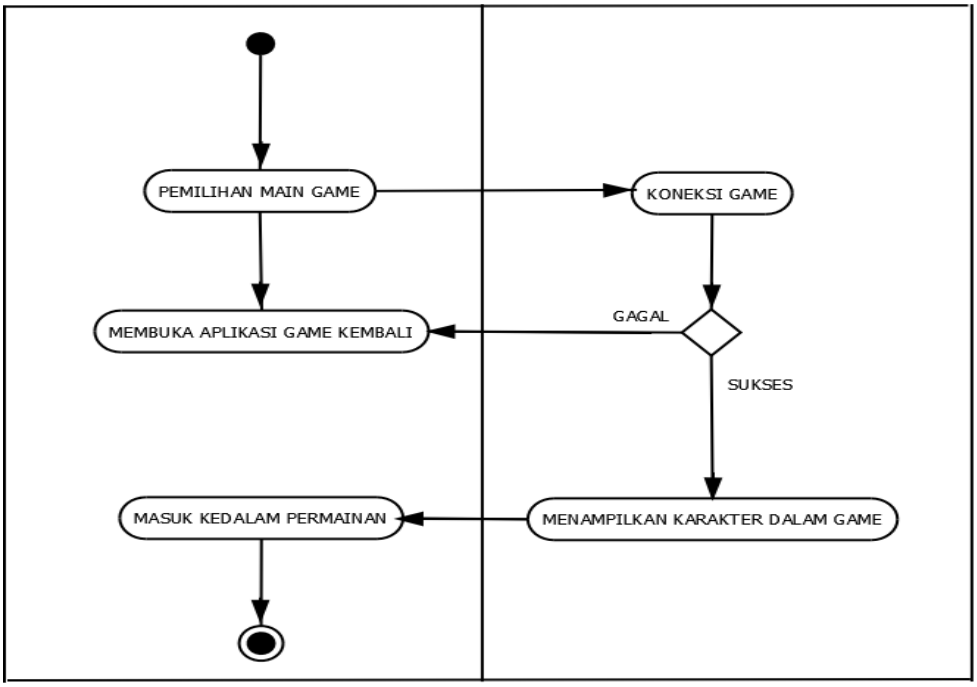
Nama	Melihat *cara main*	
Aktor	Pemain	
Deskripsi	Aktor memilih tombol *cara main* untuk melihat tutorial cara permainan game	
Exception	Gagal melihat cara main	
Pre Condition	Aktor telah membuka aplikasi game	
Aktor		**Sistem**
Skenario normal		
1. Aktor memilih tombol cara main		2. Membuka koneksi ke *game*
		3. Menampilkan informasi tentang cara bermain game
Skenario alternative		
1. Aktor memilih tombol cara main		2. Membuka koneksi ke aset *game*
		3. Koneksi ke aset *game* gagal, maka sistem kembali ke semula
4. Aktor membuka aplikasi *game* kembali		
5.memilih tombol keluar untuk kembali		
Post Condition		Aktor berhasil mengakses cara main

Tabel 4.4 Deskripsi *Use Case Diagram* keluar permainan

Nama	Keluar	
Aktor	Pemain	
Deskripsi	Aktor memilih tombol keluar untuk keluar dari *game*	
Exception	-	
Pre Condition	Aktor telah membuka aplikasi game	
Aktor		**Sistem**
Skenario normal		
1.Aktor memilih tombol keluar		2. Menutup aplikasi*game*
Skenario alternative		
Post Condition		Aktor berhasil keluar dari *game*

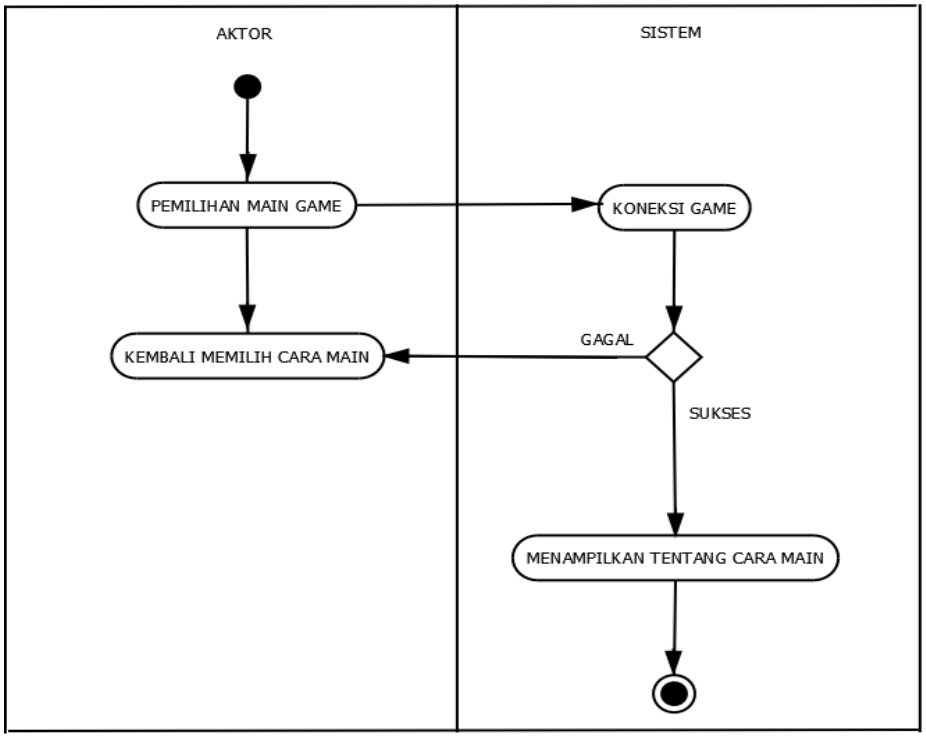
4.6.2 *Activity Diagram*

Dengan mengacu *use case diagram* yang telah dijelaskan di atas, dengan ini akan menjelaskan peracangan *activity diagram* yang akan terlihat pada Gambar 4.13 – 4.16.



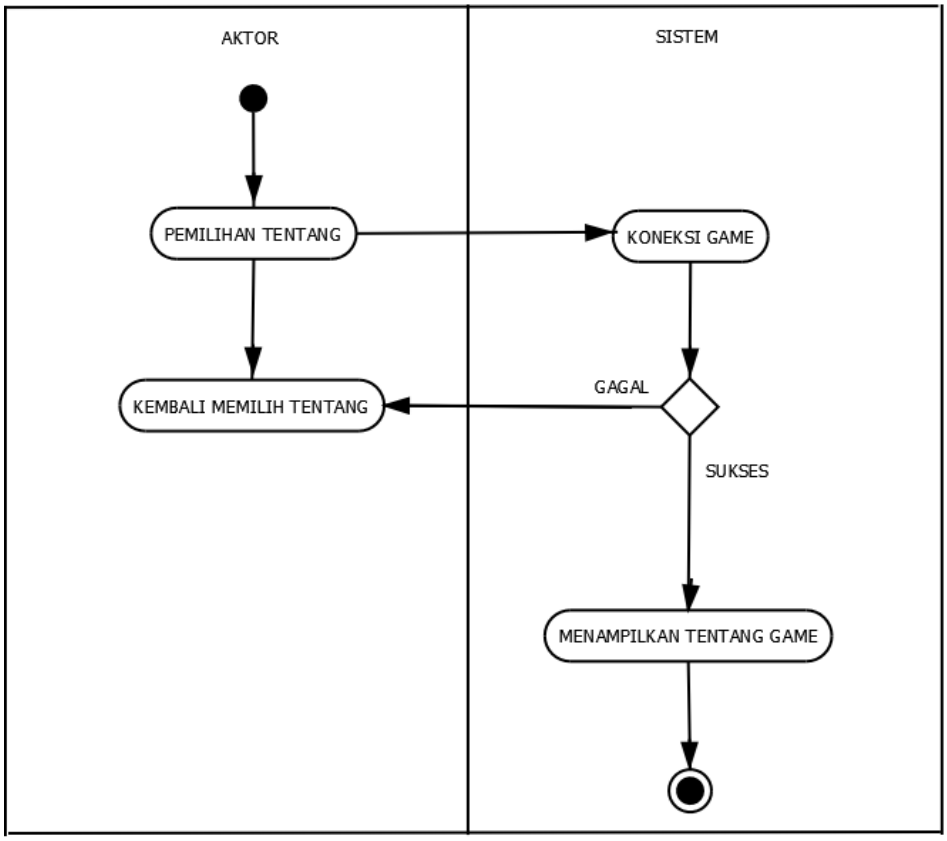
Gambar 4.13*activity diagram* main

Gambar 4.13 di atas merupakan *activity diagram* dari main, pemain memilih tombol main, jika koneksi ke dalam game berhasil akan dilanjutkan menuju penampilan karakter dan masuk permaian game *fly bee* tersebut, dan apabila koneksi pada game gagal atau tidak bisa dilanjutkan maka kembali ke semula pada saat masuk aplikasi.



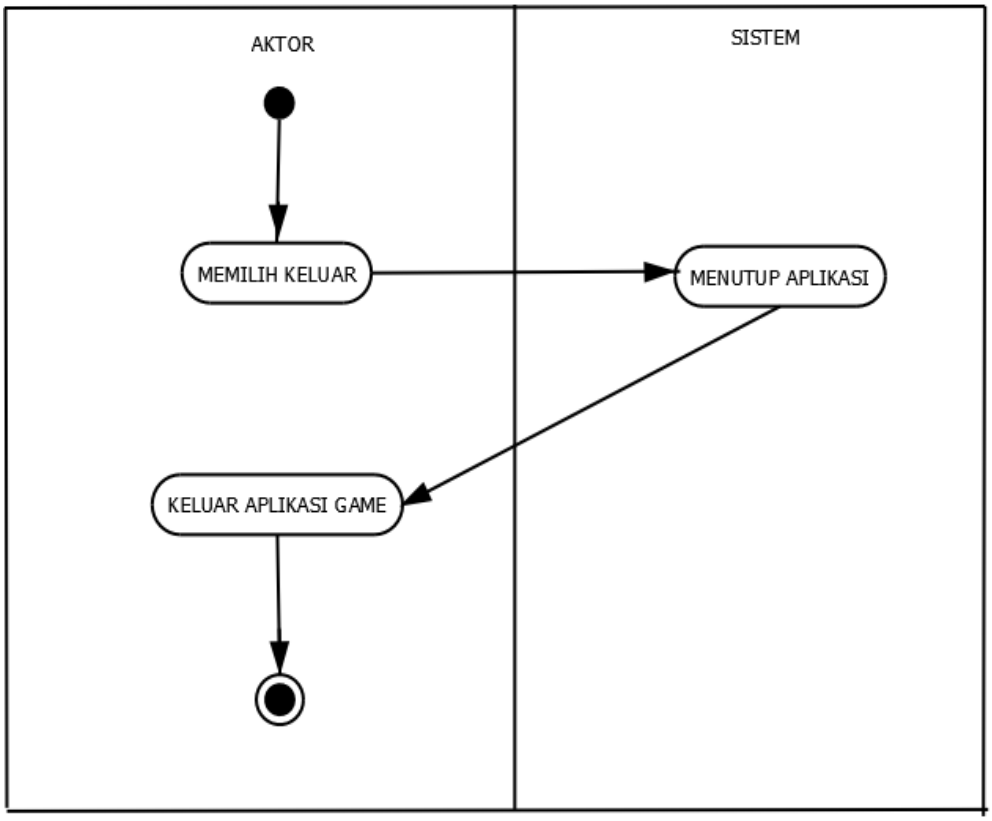
Gambar 4.14*activity diagram cara main*

Pada Gambar 4.14 *activity diagram cara main* yang akan menjelaskan cara memainkan game, pada pemilihan tombol *cara main* menuju koneksi game, jika gagal maka akan menuju kembali permulaan game, dan apabila sukses maka system menampilkan tentang *cara main* permainan game .



Gambar 4.15 *activity diagram tentang*

Gambar 4.15 diatas merupakan *activity diagram* tentang alur tentang, untuk dapat menujuk tentang, pemain memilih tombol tentang dan akan diarah menuju koneksi ke game jika gagal akan kembali ke permulaan saat membuka game, dan jika sukses menampilkan tentang yg menjelaskan tentang informasi game.



Gambar 4.16*activity diagram* keluar

Pada Gambar 4.16 diatas menjelaskan *activity diagram* keluar, pemain akan diajak keluar game dengan memilih tombol keluar yang terdapat pada menu utama.

4.7 RANCANGAN LAYOUT / TAMPILAN

Rancangan tampilan *output* dari aplikasi *game* yang akan dibangun adalah sebagai berikut :

1. Rancangan Tampilan Menu Utama

   Menu utama akan menampilkan pilihan kepada pemain. Pada menu utama terdapat tombol main, cara main, tentang dan keluar. Rancangan tampilan menu utama dapat dilihat pada Gambar 4.17.

Gambar 4.17 Rancangan Menu Utama

2. Rancangan Tampilan tentang

Tampilan *about* merupakan antar muka yang menampilkan informasi tentang aplikasi *game* ini. Rancangan tampilan tentang dapat dilihat pada Gambar 4.18.

Gambar 4.18 rancangan tentang

3. Rancangan cara main

Tampilan cara main merupakan tampilan informasi tentang cara bermain game *fly bee*,rancangan cara main akan dilihat pada gambar 4.19.

Gambar 4.19 Rancangan Cara Main

4. Rancangan Tampilan Permainan

Rancangan tampilan adalah rancangan di saat pemain menjalankan dan masuk permaianan *fly bee*, perancangan akan dilihat pada Gambar 4.20.

Gambar 4.19 Rancangan Saat Permainan

BAB V

IMPLEMENTASI

Pada bab ini akan menjelaskan tentang proses pengimplementasian pada aplikasi*fly bee*, sesuai rancangan aplikasi yang telah dilakukan pada bab sebelumnya serta melakukan pembangunan aplikasi yang telah dirancang.

5.1 Implementasi

Implementasi adalah rancangan aplikasi yang sudah di jelaskan sebelumnya dan perancangan sistem secara keseluruhan tentang implementasi *collision detection* permaianan *fly bee* menggunakan *game maker*.

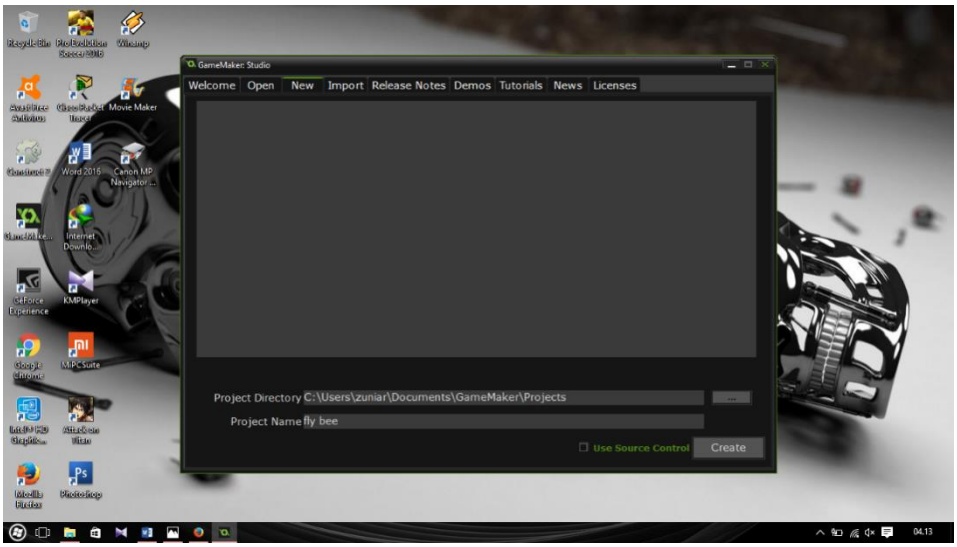
5.2 Membangun Dan Menginstal Aplikasi

5.2.1 Pembangunan Aplikasi Fly Bee

Pembangunan aplikasi terdapat beberapa tahapan, tahapan awal instalasi *game maker studio v1.4* pada pc aplikasi dapat didownload (http://www.yoyogames.com/get) serta menyiapkan android sdk dan ndk dapat di download pada (https://developer.android.com/ndk/downloads/) dan sdk pada (https://developer.android.com/studio/). Setelah terinstal aplikasi *game maker studio v.1.4* seperti Gambar 5.1.

Gambar 5.1 Hasil Instalasi *Game Maker*

Tahap selanjutnya pembuatan *project* baru dengan cara klik *New* lalu masukan nama *project* dan penempatan *projec*t dalam pc lalu *create* dapat dilihat pada gambar 5.2.



Gambar 5.2 Membuat *Project* Baru

Selanjutnyapembuatan game *fly bee* pada *game maker studio v.1.4* dengan memasukan sprite dengan cara :

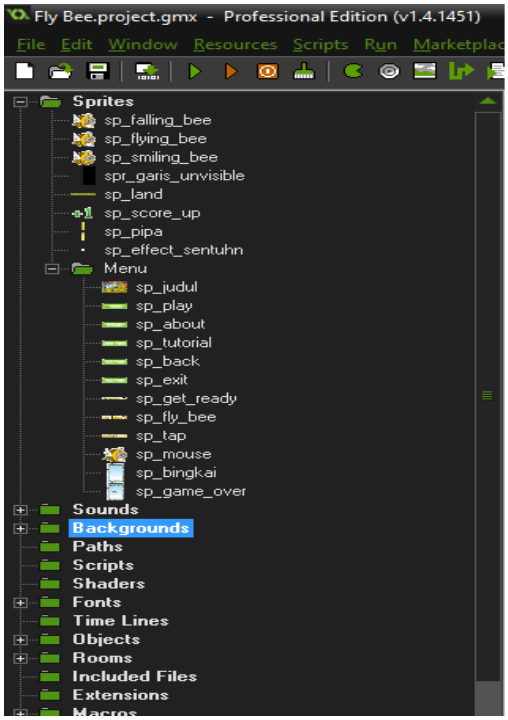
a. Klik kanan sprite => create Sprite

b. Load sprite => sp_falling_bee =>open

c. Modify mask => bounding box =>automatic => shape =>precise

d. Lebih lengkap nya dapat di lihat pada lampiran,

Sprite yang sudah dimasukan dapat di lihat pada Gambar 5.3.



Gambar 5.3 Sprite Yang Sudah Di Masukan

Selanjutnya tahap memasukan *background* didalamnya terdapat 4 background yaitu *back1*, *back_tutorial*, *back_menu*, *back_about.* Cara membuat *background* untuk masing halaman yaitu:

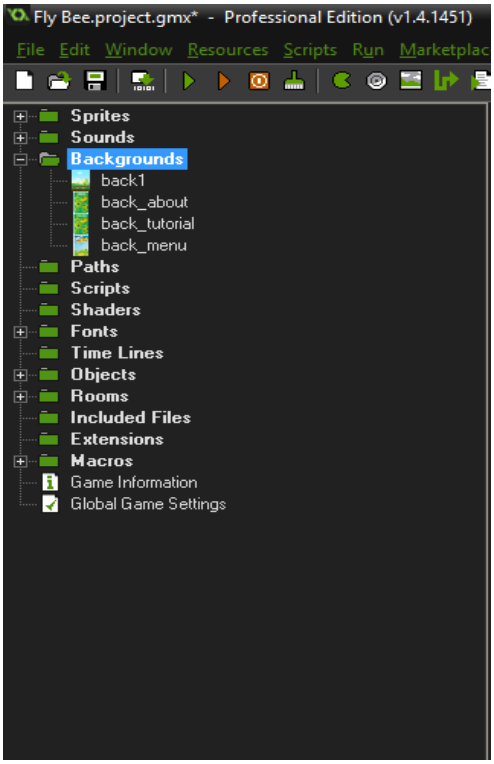
a. Klik kanan pada *background* =>create *background*

b. Isi nama dengan back1

c. *Load background* => masukan gambar => open

d. Lebih lengkapnya lihat pada lampiran

Jika telah dimasukan semua *background* seperti Gambar 5.4.

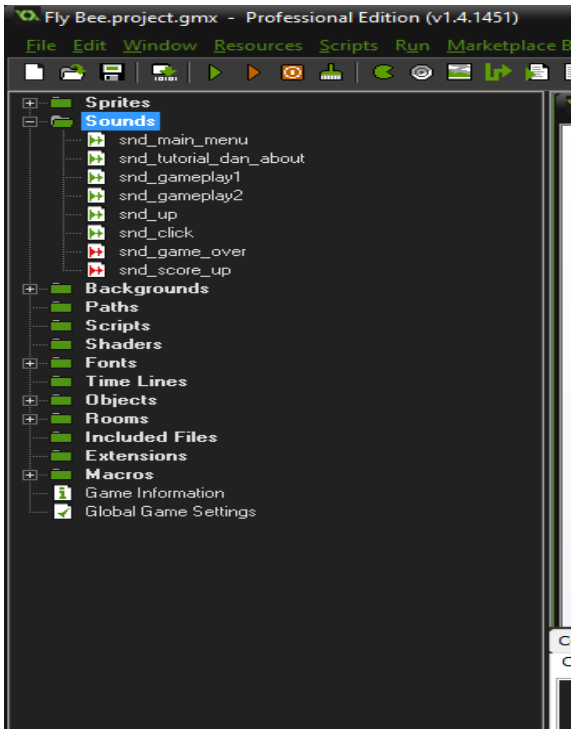


Gambar 5.4 Hasil Memasukan *Background*

Selanjutnya memasukan daftar *sound* atau musik yang akan dipanggil untuk latar musik yaitu dengan cara menyiapkan kebutuhan masing *music* yang harus diputar pada latar dalam *fly bee* terdapat *sound main menu, tutorial dan about, gameplay1,2, click, up, score, game ove*r, cara memasukannya sebagai berikut :

a. Klik kanan sounds =>create sound
b. Buka sound dalam folder => snd_main_menu => open => ok
c. Selengkapnya pada lampiran .

Jika sudah dimasukan semua music ke dalam sound game maker seperti Gambar 5.5.



Gambar 5.5 Sound Yang Telah Di Masukan

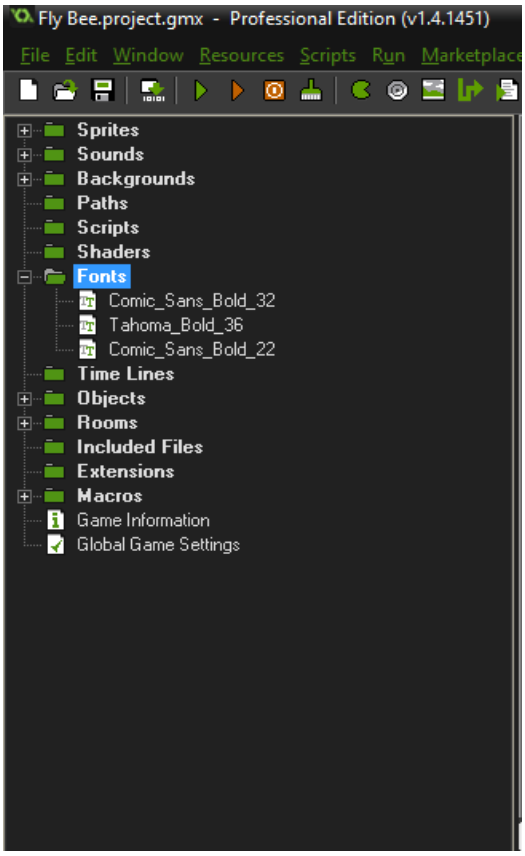
Pada tahap selanjutnya memasukan *font* yang akan kita panggil *object* nanti guna mengenalkan *font* yang dipakai pada pembuatan *project*, cara memasukan *font* sebagai berikut:

a. Klik kanan font => create font

b. Nama => comic_Sans_bold_32 => font => comic sans=> ok

c. Lebih lengkapnya lihat pada lampiran .

Jika *font* telah dimasukan yang sekiranya dibutuhkan seperti Gambar 5.6.



Gambar 5.6 *Font* yang telah dimasukan

Tahap selanjutnya adalah membuat object gunanya untuk menetukan gerakan serta perintah untuk semua *sprite*, *background*, *sound*, *font*, yang telah dimasukan, didalam object terdapat sub folder agar mudah membuatnya, dikelompok kan yaitu : *menu*, *player*, *room*, *anything* sebelumnya dalam membuat *object* berikut langkah pembuatan nya :

a. Klik kanan pada objects => create group => nama => menu => ok
b. Klik kanan menu => creat object
c. Beri nama yang di ingin kan => sprite => arahkan yang sprite yang diberi perintah
d. Add event => beri event yang diingin kan
e. Selengkapnya lihat pada lampiran

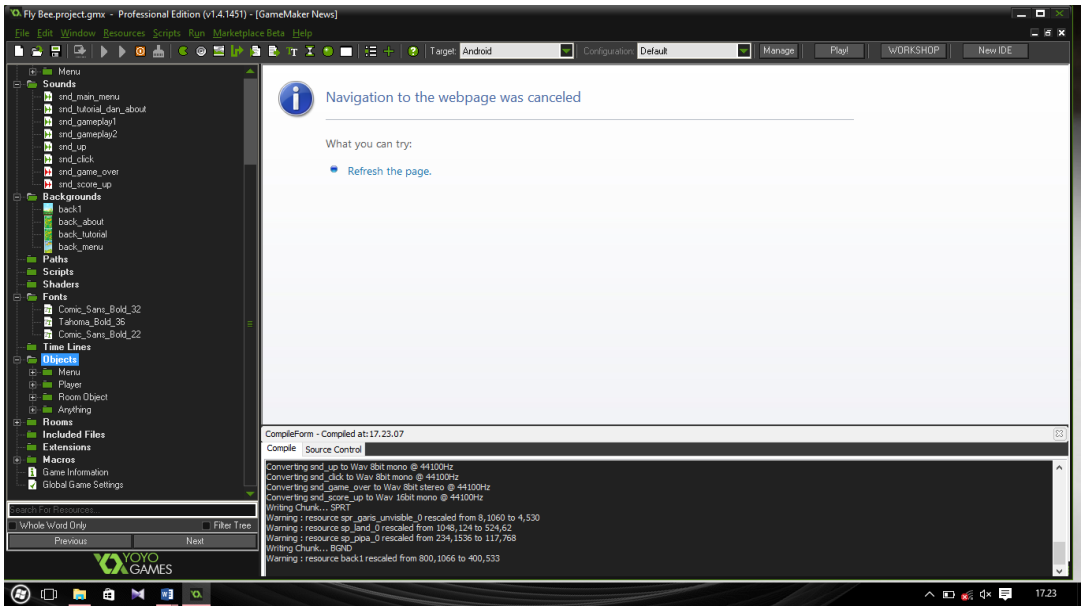
Setelah selesai memberikan perintah dan memasukan pada *objects*, tahap selanjutnya membuat *room*, gunanya untuk memasukan *background* dan letak sprite yang telah diberi perintah pada *objetcs*, berikut cara membuatnya:

a. Klik kanan pada room=> create room
b. Nama => beri nama yang diinginkan
c. Klik *background* => pilih *background* yang telah di masukan pada background sebelumnya
d. Pilih *object* pilih *objects* untuk di masukan diatas *background* guna penempatan
e. Lebih lengkap lihat pada lampiran

5.2.2 Menginstal Aplikasi

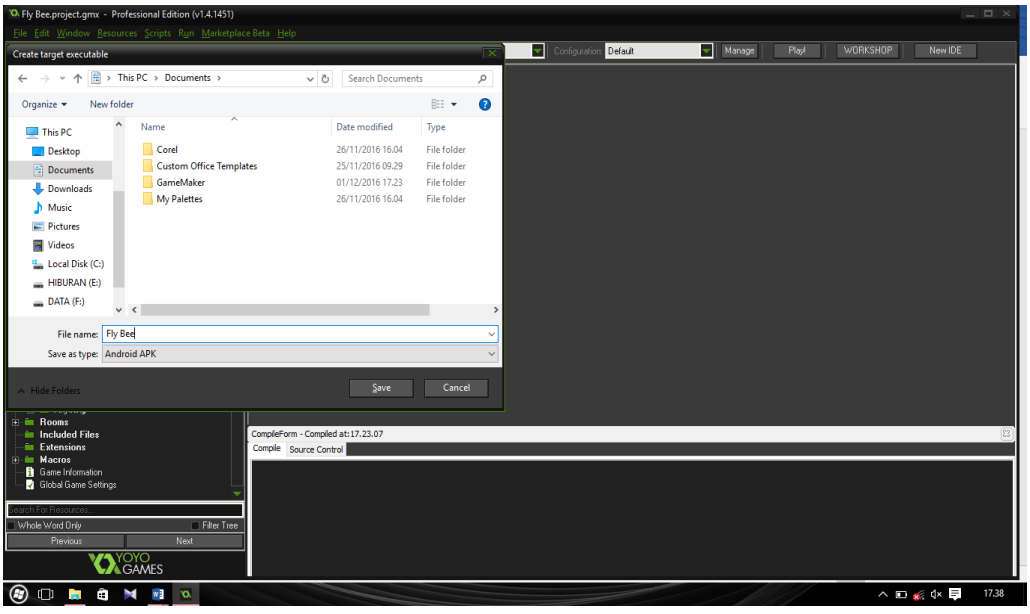
Pada proses melakukan instalasi aplikasi game *fly bee* pada android terdapat beberapa tahapan yang dilalui yaitu :

1. Melakukan pengecekan project yang telah dibuat pada aplikasi game maker, setelah benar *project* akan dilanjutkan dengan pengecekan *error* sebelum dijadikan *file apk*, pengecekan tersebut akan ditampilkan pada Gambar 5.7.



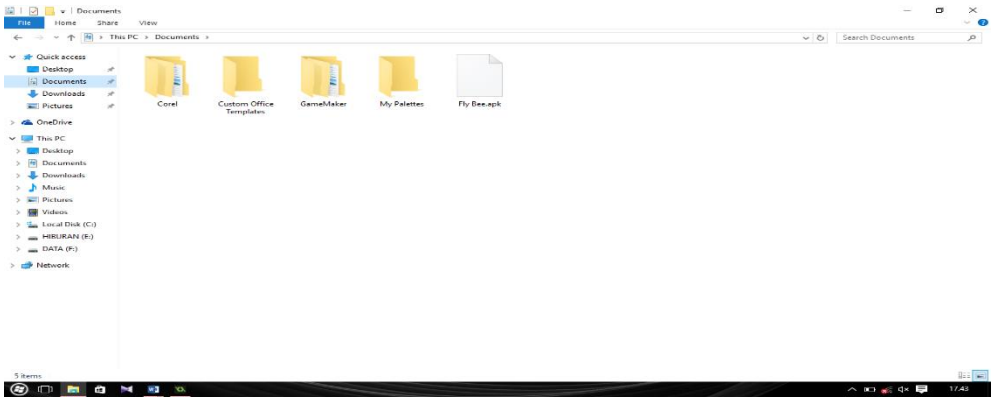
Gambar 5.7 Tampilan Project Pada Game Maker

2. Selanjutnya menjadikan *project file apk*, dengan cara merubah target menjadi android dengan ini bertujuan agar file yang di tujukan yaitu *apk*, Selanjutnya buka menu *file*, *create application* lalu beri nama *file* yang di inginkan arahkan *folder* yang akan ditaruh *apk* tersebut lalu *save*, dapat dilihat pada Gambar 5.8.



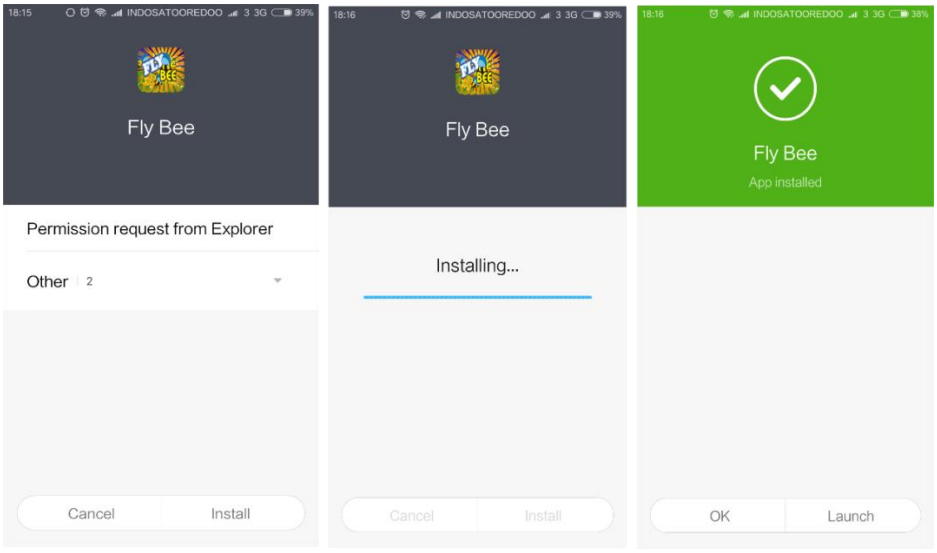
Gambar 5.8 Save Pada *Creat Application* Pada *Game Maker*

3. Kemudian untuk mengambil *file .apk* program yaitu dengan cara membuka *file explorer*, selanjutnya cari file yang telah disave pada tahap sebelumnya dengan format *file.apk*, dapat dilihat pada Gambar 5.9.



Gambar 5.9 Penyimpanan *Project File Apk* Pada *Game Maker*

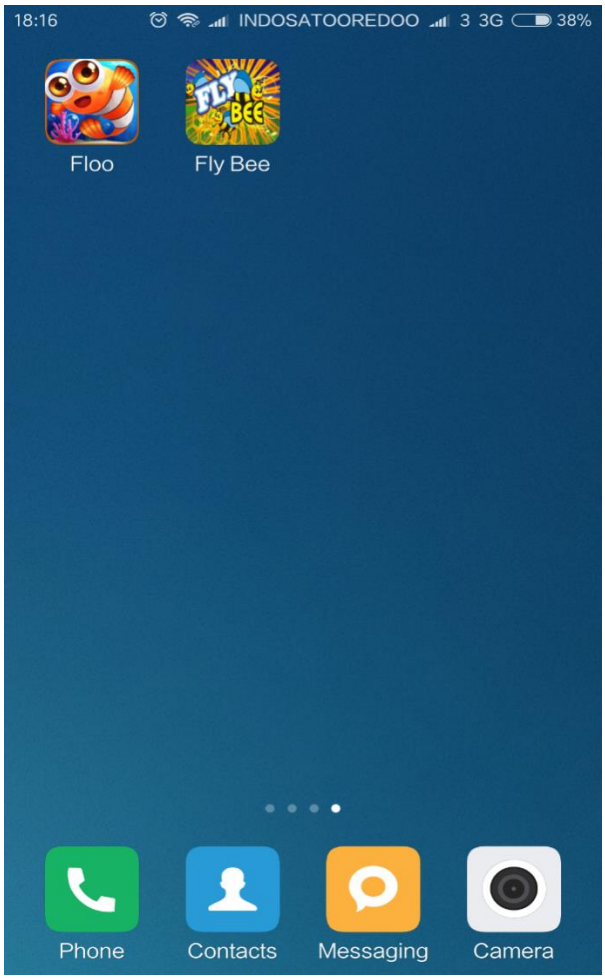
4. Kemudian *file fly bee.apk* dapat disalin pada*smartphon*e berbasis android dengan cara mentransferkan menggunakan kabel data atau *bluetooth*.
5. Selanjutnya penginstalan file *fly bee.apk* dismartphone yang telah ditransfer *apk* sebelumnya, akan dilihat pada Gambar 5.10.



Gambar 5.10 Instalasi Pada *Smartphone*

5.3 Hasil Instalasi Aplikasi

Hasil instalasi yang telah dilakukan sebelumnya dari *file fly bee.apk* akan tampil pada menu *smartphone*. Hasil instalasi dapat dilihat pada Gambar 5.11.

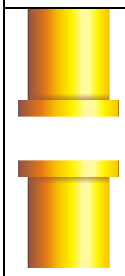
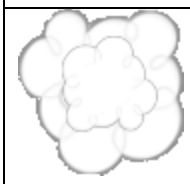


Gambar 5.11 Hasil Instalasi Aplikasi *Fly Bee*

5.4 Tampilan Karakter Pada Game Fly Bee

Game *fly bee* memiliki beberapa *sprite* atau karakter saat dimainkan, dan ini beberapa *sprite* karakter yang digunakan dalam game *fly bee* akan ditampilkan di Tabel 5.1.

Tabel 5.1 Karakter *Fly Bee*

Gambar Karakter	Nama Karakter	Keterangan
	• Bee user	• Lebah bee yang akan dimainkan oleh user
	• Pipa rintangan	• Pipa - pipa yang menghadang saat user memainkan bee user
	• Efek sentuh	• Efek tabrakan antara bee dengan pipa yang menghalangi
	• Efek Poin	• Efek poin jika melewati pipa - pipa

5.5 Skenario Pada Game Fly Bee

User akan memainkan *bee user* dengan mengontrol grafitasi *bee* agar tetap stabil menghindari rintangan pipa-pipa, yang akan secara acak *bee user* menhindarinya dengan melewati celah antara pipa atas dan bawah.

Permainan akan dimulai sejak *bee* di *tap* atau disentuh dan berjalannya *bee* menghindari pipa. Pada game ini menggunakan sistem *score*, semakin *score* bertambah maka akan bertambah kecepatan *bee*, apabila *bee user* mengenai pipa dan jatuh ke bawah sehingga mengenai rumput, maka akan gugur dan mengulang. *score* akan ditampilkan saat *game over* sebelum mengulang pada awal game lagi.

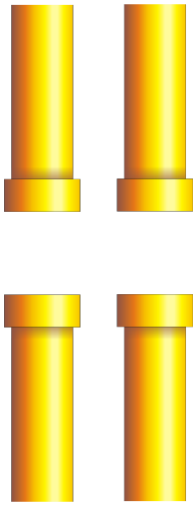
Ada beberapa animasi sprite yang di butuhkan pada game fly bee beberapa sprite tersebut mendukung segala pembuatan serta rancangan pada game ini yaitu :

Terdapat beberapa sprite animasi yang di kemukakan pada game fly bee yaitu :

1. Sprite falling bee dan flying bee

Gambar 5.12 sprite flying bee dan falling bee

2. Sprite pipa



Gambar 5.13 sprite pipa

3. Sprite background menu, tentang, cara main, menu 1

Gambar 5.14 Sprite background menu, tentang, cara main, menu 1

4. Sprite animasitombol main, tentang, cara main, keluar

Gambar 5.15 Sprite tombol main, tentang, cara main, keluar

5. Sprite animasi land

Gambar 5.16 sprite land

6. Sprite animasi effect sentuh

Gambar 5.17 sprite effect sentuh

7. Sprite animasi score up

Gambar 5.18 Sprite score up

8. Sprite animasi judul

Gambar 5.19 sprite judul

9. Sprite animasi bingkai dan game over

Gambar 5. 20 Sprite bingkai dan game over

10. Sprite animasi sentuh dan bersiap

Gambar 5.21 Sprite sentuh dan bersiap

11. Sprite animasi garis unvisible

Gambar 5.22 Sprite garis unvisible

5.6 Tampilan Menjalankan Game Fly Bee

Pada tampilan awal akan menampilkan *logo* kemudian akan muncul *menu : play, tutorial, about, exit.*

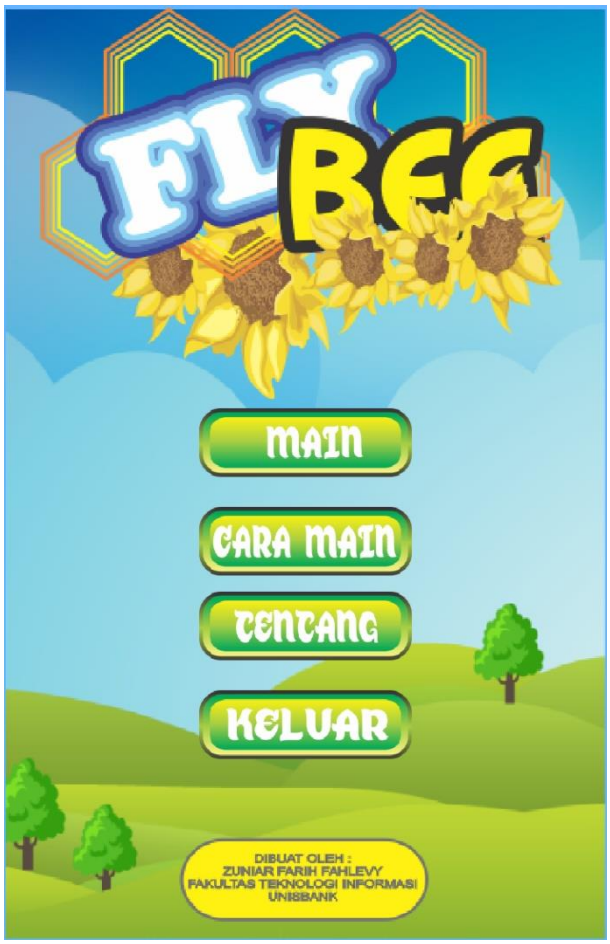
5.6.1 Tampilan Pembuka

Setelah game di jalan kan akan ditampilkan pertama kali seperti Gambar 5.23.

Gambar 5.23 Tampilan Pembuka

5.6.2 Tampilan Menu Utama

Setelah tampilan pembuka akan dilanjutkan dengan tampilan menu utama, dimenu tersebut terdapat *main*, *cara main*, *tentang*, *keluar* dapat dilihat pada Gambar 5.24.



Gambar 5.24 Menu Utama

5.6.3 Tampilan Halaman Cara Main

Apabila user memilih cara main pada menu utama makan akan menampilkan informasi cara permainan *fly bee* sepeti pada Gambar 5.25.

Gambar 5.25 Tampilan Cara Main

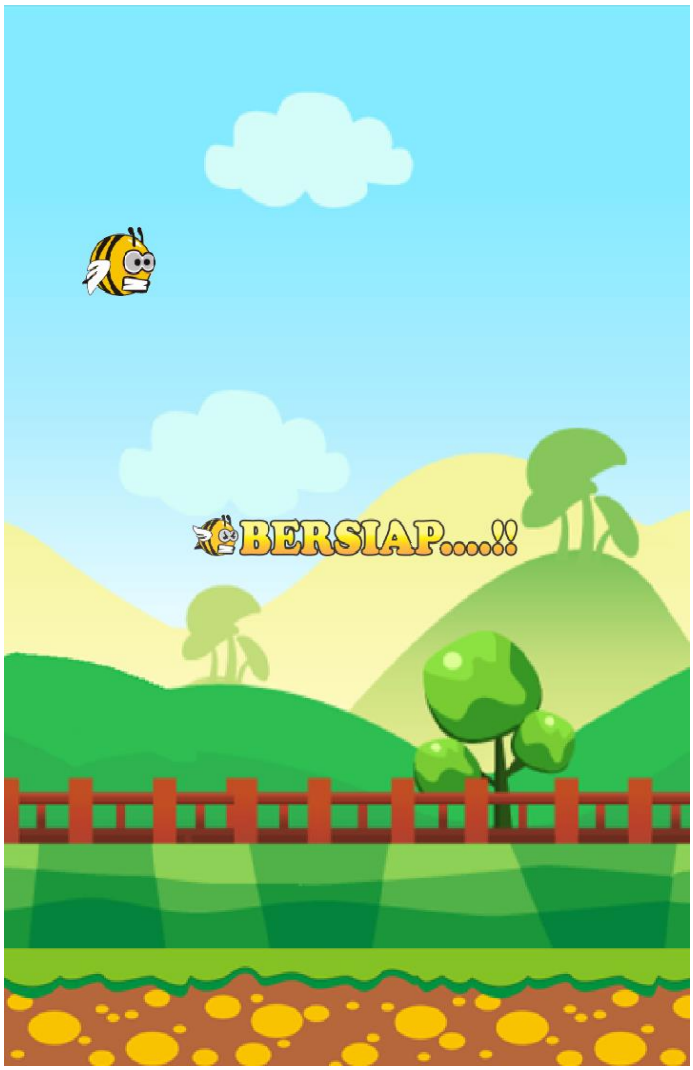
5.6.4 Tampilan Tentang

Apabila *user* memilih tentang pada menu utama maka akan menampilkan informasi pembuat game serta *email* pembuat seperti Gambar 5.26.

Gambar 5.26 Tampilan Tentang

5.6.5 Tampilan Memulai Game

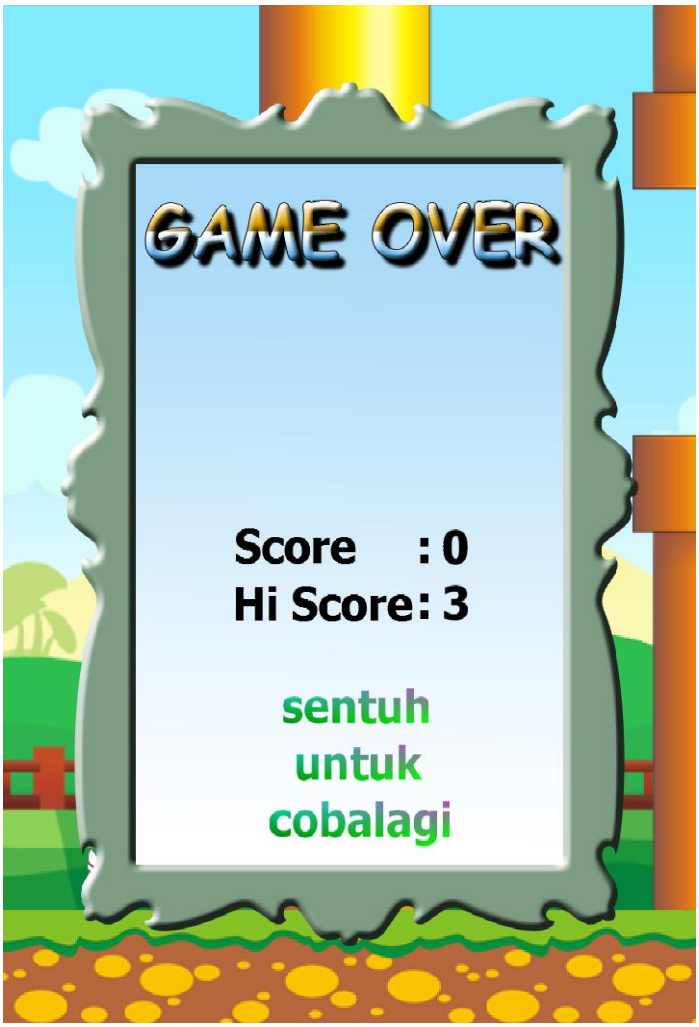
Apabila *user* memilih *main* akan diajak menunjuk permaianan game seperti Gambar 5.27.



Gambar 5.27 Game Berjalan

5.6.6 Tampilan *Game Over*

Tampilan *game over* menandai bawah game telah gagal, maka akan ditampilkan *score* sekarang serta *score* terbaik seperti Gambar 5.28.



Gambar 5.28*Game Over*

5.7 Pengujian Sistem

Pengujian sistem untuk mengetahui yang telah dicapai dalam menjalankan sebuah aplikasi, dengan hal ini peneliti akan menguji sistem berjalan dengan benar atau ada kendala dalam menjalan kan sesuai yang di kehendaki.

Pada pengujian aplikasi *fly bee* peneliti menggunakan perangkat keras sebaigai berikut :

a. Xiaomi redmi pro
b. RAM 3 gb
c. Layar : 1080 x 1920 pixels, 5.5 inch
d. CPU : Mediatex MT6797 helio x20
e. OS :Android, v 6.0 (Marshmallow)

Pengujian ini dilakukan pada game *fly bee* meliputi beberapa hal yaitu : main game, tata cara, tentang.

5.7.1 Pengujian Pada Smartphone

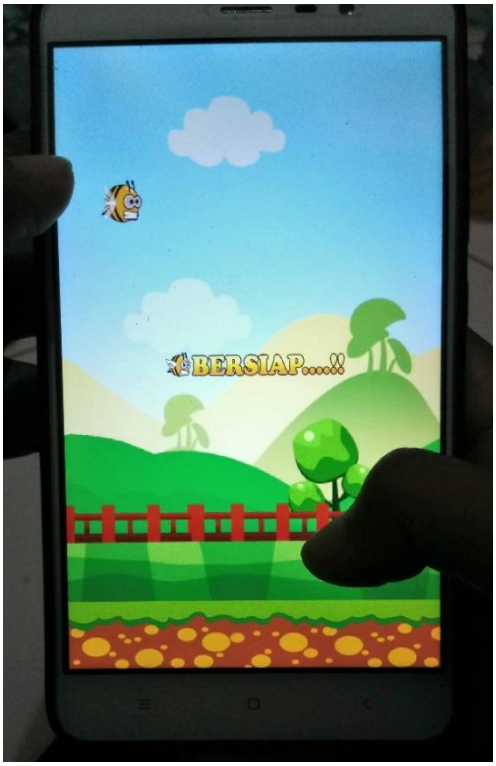
Setelah menginstal aplikasi *fly bee* pada *smartphone* terdapant *icon fly bee* yang muncul pada daftar aplikasi *smartphone*, jika dipilih *icon fly bee* maka akan masuk ke dalam permaian, dapat dilihat pada Gambar 5.29.

Gambar 5.29*Icon fly bee* yang telah terinstal

Selanjutnya jika dipilih *iconfly bee* akan masuk game dan menampilkan menu utama, pada menu utama terdapat 4 menu yaitu main, cara main, tentang, keluar dengan fungsi yang berbeda dapat dilihat pada Gambar 5.30.

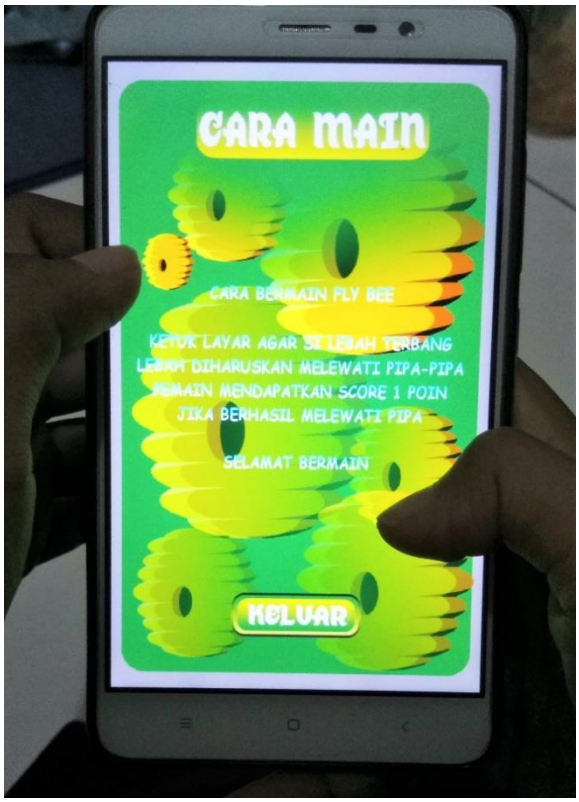
Gambar 5.30 Tampilan menu utama saat di jalankan

Tampilan selanjutnya jika dipilih menu main akan masuk ke dalam permaianan *fly bee* dapat dilihat pada Gambar 5.31.



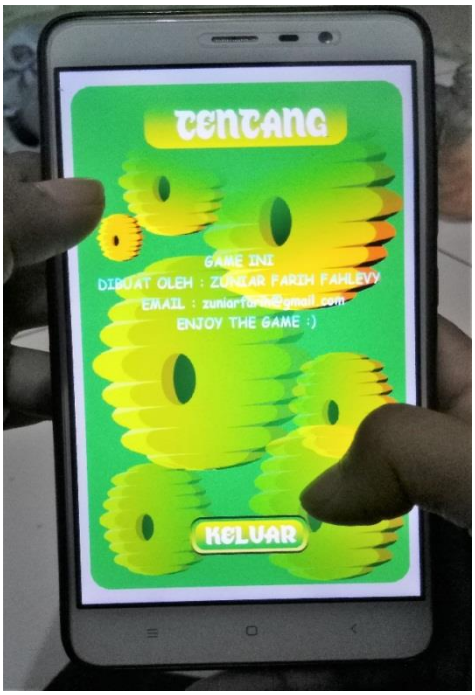
Gambar 5.31 Tampilan saat permainan berjalan

Apabila dipilih cara main akan menampilkan halaman tentang cara bermain *fly bee* dapat dilihat pada Gambar 5.32.



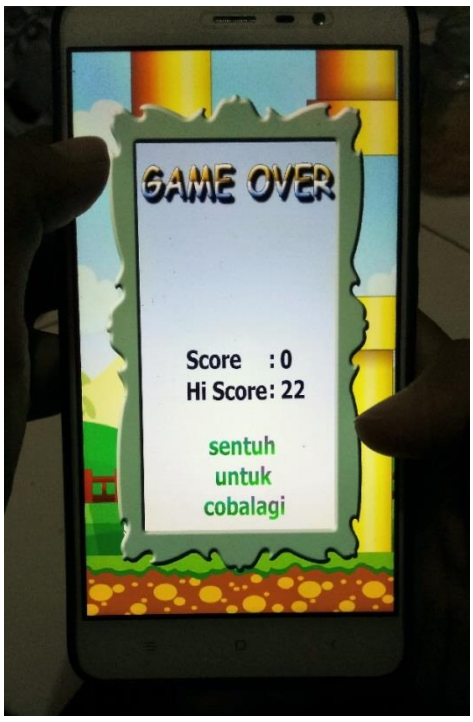
Gambar 5.32 Tampilan cara main

Selanjutnya jika dipilih tentang akan menampilkan halaman informasi tentang pembuat game serta *email* pembuat, dapat dilihat pada Gambar 5.33.



Gambar 5.33 Tampilan tentang

Tampilan *game over* serta *score* tertinggi akan di tampilkan jika pemain selesai pada terakhir permaian atau gagal, dapat dilihat pada Gambar 5.34.



Gambar 5.34 Tampilan *game over* serta *score* tertinggi telah dicapai

BAB VI

HASIL PENELITIAN DAN PEMBAHASAN

Pada bab ini akan menjelaskan tentang hasil penelitian dan pembahasan pada bab sebelumnya tentang implementasi game *fly bee* dengan *game maker*.

6.1 Hasil

Hasil penelitian menyimpulkan agar aplikasi *game fly bee* dapat berjalan dengan baik sesuai rancangan yang telah dibahas sebelumnya, hasil penelitian juga mengacu semua kegiatan dalam perancangan *game fly bee* dengan *game maker*. Pengujian aplikasi *game fly bee* dibuat untuk perangkat *mobile* bersistem operasi android dengan resolusi yang direkomendasikan 720x1080 5 inc.

Pada hasil ini peneliti akan penerangkan pengujian keseluruhan yang telah diimplementasikan sebelumnya, hasil pengujian aplikasi meliputi : pengujian, bentuk pengujian, masukan, keluaran yang diharapkan, hasil yang didapat, dan hasil pengujian, bentuk pengujian tersebut dapat dilihat pada Tabel 6.1.

Tabel 6.1 Hasil Pengujian Pada Aplikasi *Fly Bee*

Pengujian	Bentuk Pengujian	Masukan	Keluaran Yang Diharapkan	Hasil Yang Diharapkan	Hasil Pengujian
▪ Pengujian instalasi pada smartphones	▪ Memasukan file .apk kedalam smarphone	▪ Menekan file apk fly bee untuk instalasi	▪ Sukses instalasi fly bee	▪ Menghasilkan icon aplikasi fly bee yang dibuat	Berhasil
▪ Menu utama	▪ Buka aplikasi fly bee	▪ Menekan pada Aplikasi fly bee	▪ Menampilkan ▪ menu utama	▪ Menampilkan menu utama	Berhasil
▪ Memulai game	▪ Buka aplikasi ▪ Pada menu utama pilih tombol main	▪ Pilih pada tombol main	▪ Menampilkan main permainan fly bee	▪ Menampilkan Main permianan fly bee	Berhasil
▪ Cara main	▪ Buka aplikasi ▪ pada menu utama pilih tombol cara main	▪ Pilih pada tombol cara main	▪ Menampilkan Halaman cara main	▪ Menampilkan Cara main	Berhasil
▪ Tentang	▪ Buka aplikasi ▪ Pada menu utama pilih tombol tentang	▪ Pilih pada tombol tentang	▪ Menampilkan halaman tentang	▪ Menampilkan halaman tentang	Berhasil
▪ Keluar	▪ Buka aplikasi ▪ Pada menu utama pilih tombol keluar	▪ pilih pada tombol keluar	▪ Keluar dari aplikasi game	▪ Keluar dari aplikasi game	Berhasil
▪ Musik latar	▪ Buka aplikasi ▪ pilih tombol main	▪ Menekan pada Aplikasi ▪ pilih tombol main	▪ Menampilkan menu utama ▪ memainkan musik latar ▪ Masuk pada permainan ▪ Memainkan musik latar	▪ Menampilkan menu utama ▪ memainkan musik latar ▪ menampilkan berjalannya permaianan ▪ memainkan musik latar	Berhasil
▪ Suara Efek	▪ Buka aplikasi ▪ Pada menu utama ▪ pilih tombol mulai permainan	▪ Pilih pada tombol main game	▪ Menampilkan game main ▪ Memainkan musik efek saat melewati pipa	▪ Menampilkan game main ▪ Memainkan musik efek saat melewati pipa	Berhasil

Berikut ini adalah hasil yang di harapkan pada Tabel 6.1, hasil pengujian aplikasi dapat dilihat pada Gambar 6.1(a).-6.1 (d).

Gambar 6.1 (a) Hasil tampilan menu utama

Gambar 6.1 (b) Menampilkan main

v

Gambar 6.1(c) Hasil tampilan cara main

Gambar 6.1(d) Hasil tampilan tentang

6.2 Pembahasan

Peneliti mengemukakan beberapa hal yang meliputi pembahasan aplikasi game fly bee :

6.2.1 Game maker

Aplikasi game ini dibangun menggunakan *game maker v1.4* yaitu aplikasi yang mudah membuat dan merancang suatu aplikasi seperti game dengan metode *drop down* serta terdapat berberapa *language* dari game maker itu sendiri, dalam pembuatan *game fly bee* dari awal dan sampai membuat file apk dengan game maker tersebut.

6.2.2 Skenario dalam game *fly bee*

Permainan akan dimulai saat *bee* di tap atau disentuh dan berjalannya *bee* menghindari, dalam game ini menggunakan sistem *score*, semakin *score* bertambah maka akan bertambah kecepatan *bee*, apabila *bee user* mengenai pipa dan jatuh sehingga mengenai rumput, maka akan gugur dan mengulang. *score* akan ditampilkan saat *game over* sebelum mengulang pada awal game lagi.

6.2.3 Implementasi *collision detection*

Penerapan collision pada game fly bee terdapat beberapa hal yaitu rumusan saat bee bertabrakan dengan pipa dan Penerapan*collision tool*game fly bee.

a. Rumusan saat obj_bee bertabrakan obj_pipa

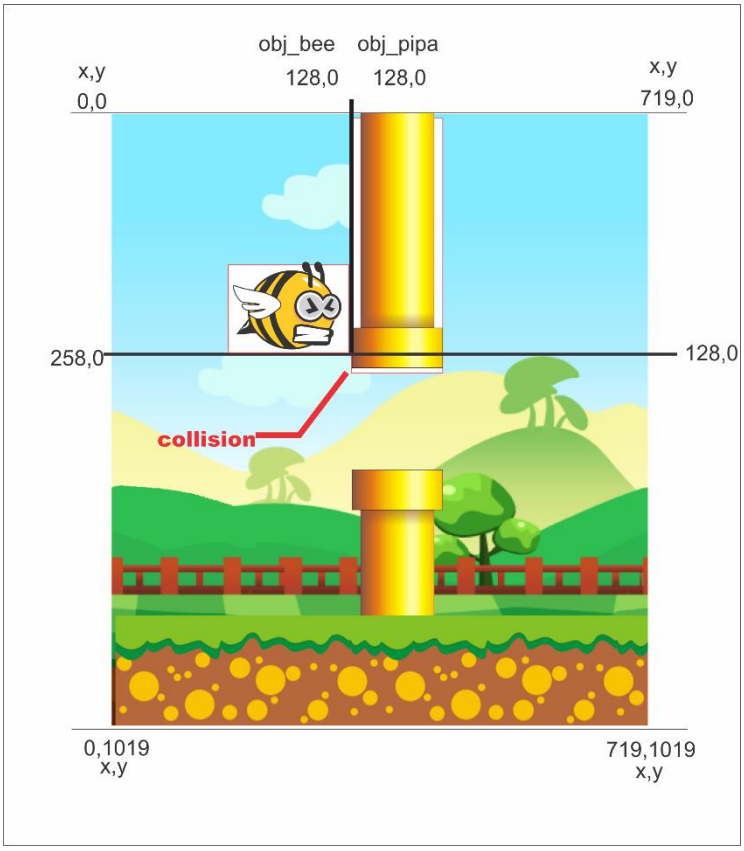
   Tahapan awal dalam rumusan *collision* pada obj_bee dengan pipa adalah menginisialisasi rumus pada *collision* sebagai berikut : Untuk Pemain (obj_bee) misal mempunyainilai (x1, y1, lebar1 dan panjang1) untuk (obj_pipa)misal mempunyai nilai(x2,y2, lebar2 dan panjang2)

Collision detection = ((x1 < x2 + lebar2)&& (x1+lebar1 > x2)&&

(y2 < y2 +panjang2) && (y1 +panjang1 > y2))

```
obj_bee left,x=128,;
        right,x+88,
        bottom,y=258,
        top y+72
obj_pipa_1 left, x= 128,;
        right,x+234,
        bottom,y=ay,by,cy,dy,ey,fy,gy,hy,iy,jy,ky,ly,my,
        top y+1536
```

Dari penjelasan diatas kordinat *bounding boxobj_bee* sisi kiri (left) x bernilai 128, sisi kanan (right) x bernilai 128 + lebar gambar obj_bee yaitu 88, sisi bawah y bernilai 258, sisi atas y + tinggi gambar obj_bee, begitu juga pada *obj_pipa* sisi kiri (left) x bernilai 128, sisi kanan (right) x bernilai 234 + lebar gambar obj_pipa yaitu 88, sisi bawah y bernilai random maka dipilih variabel ay yaitu 128, sisi atas y + tinggi gambar obj_pipa yaitu 1536. Titik kordinat x,y pada obj_bee dan obj_pipa dapat dilihat pada gambar 6.2.



Gambar 6.2 titik kordinat collision bee dan pipa

(obj_bee.left < obj_pipa.right) dan (obj_bee.right > pipa.left) dan

(obj_bee.bottom < obj_pipa.top) dan (obj_bee.top > pipa.bottom)

Jadi dalam rumusan diketahui :

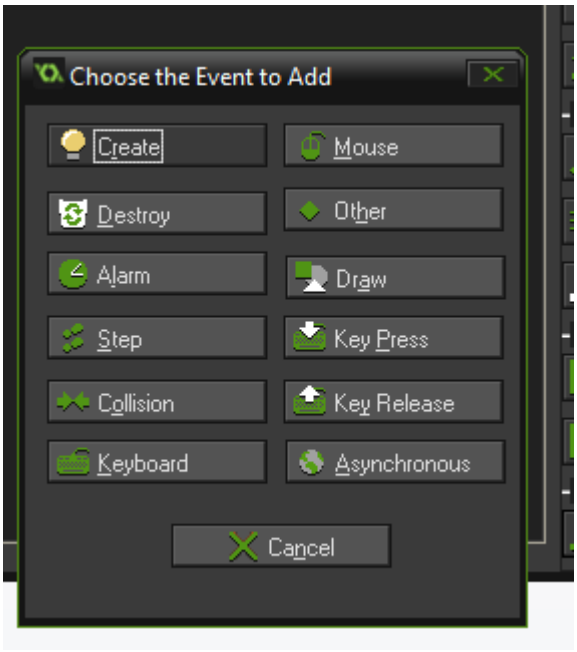
128 < 3114 dan 216 > 128

258 < 1664 dan 330 > 128

Jika kondisi di atas sama dengan rumus *collision detection* maka akan terjadi tabrakan.

b. Penerapan *collision tool* game fly bee

Pada game maker sendiri terdapat tool yang khusus mengimplementasikan *collision* antara objek, dalam hal ini dapat dilihat pada Gambar 6.3



Gambar 6.3 tool *collision* pada game maker

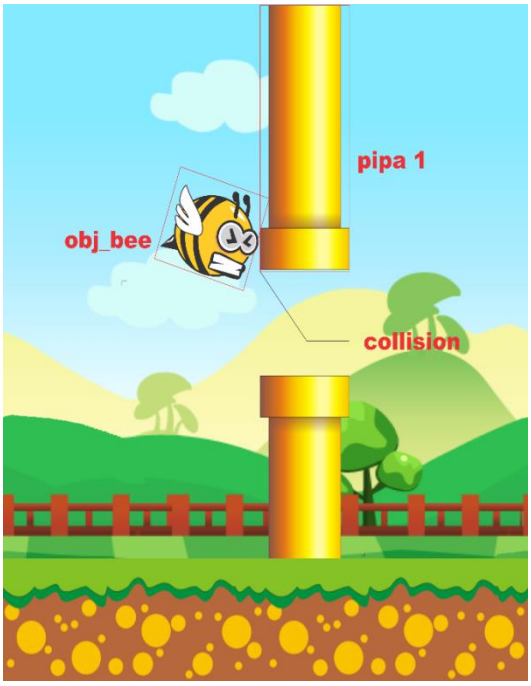
penerapan tool pada game maker sendiri terhadap game fly bee ada beberapa kondisi yaitu

1. kondisi saat obj_bee bertabrakan obj_pipa 1

kondisi saat tabrakan obj_bee dengan obj_pipa 1 di jelaskan dalam event *collision*, berikut adalah kodeprogram :

```
if fly == true // jika fly bernilai true
{
   instance_create(x+44,y,obj_effect_sentuhan);
   sprite_index = sp_falling_bee;
fly = false;
   sound_play(snd_up);
}
```

Pada kode program diatas, fly adalah variabel global yang dibuat untuk penanda terbang yang terdapat pada obj_control. Jika fly == true artinyaobj_bee bisa terbang bila layar disentuh. Sebaliknya jika fly = false, maka obj_bee tidak bisa terbang ketika layar disentuh. Bila obj_bee sedang terbang (fly = true) dan menabrak pipa maka fly = false, agar obj_bee tidak bias terbang lagi ketika layar disentuh, untuk menandakan bahwa obj_bee telah menabrak pipa, maka di create (diciptakan) object baru bernama obj_effect_sentuhan. Dapat dilihat pada Gambar 6.4



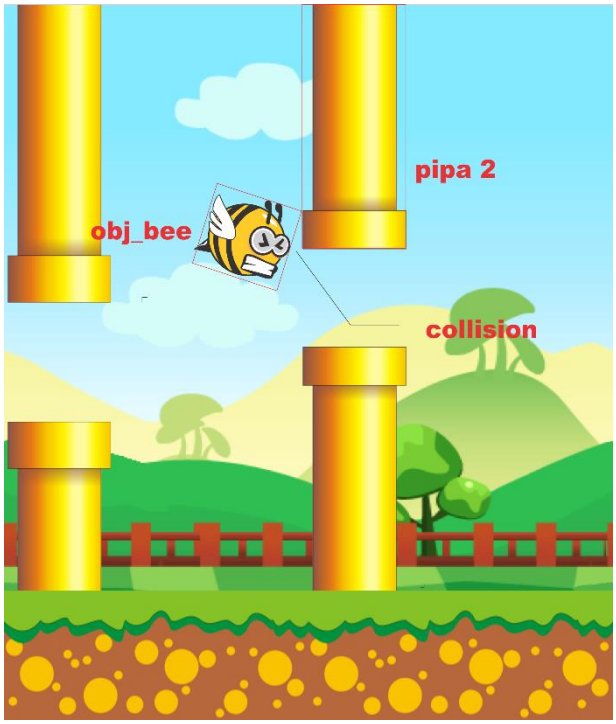
Gambar 6.4 tabrakan obj_bee dengan pipa1

2. kondisi saat obj_bee bertabrakan obj_pipa 2

Pada kondisi obj_bee bertabrakan obj_pipa 2 dilihat pada kodeprogramberikut :

```
if fly == true
{
    instance_create(x+44,y,obj_effect_sentuhan);
    sprite_index = sp_falling_bee;
fly = false;
    sound_play(snd_up);
```

Collision detection antara obj_bee dan obj_pipa2didapat dilihat pada Gambar 6.5.



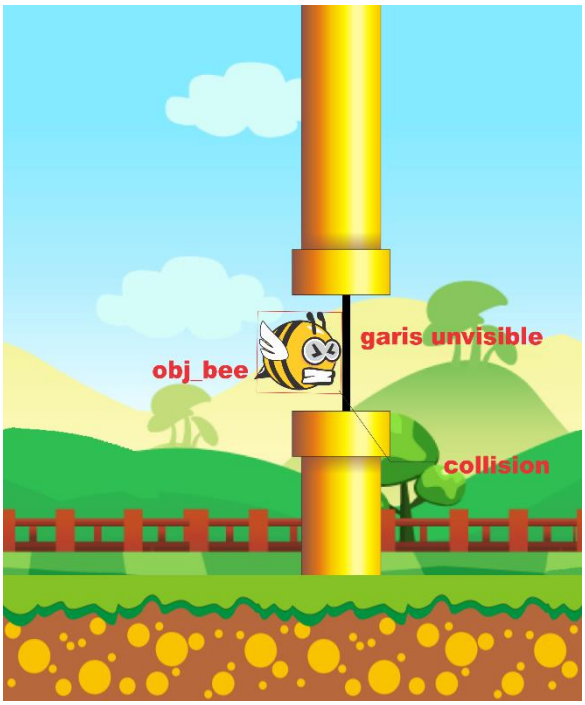
Gambar 6.5 tabrukan obj_bee dengan pipa 2

3. kondisi saat obj_bee brtabrakan garis unvisible

Selain pipa1 dan pipa 2 terdapat *implementasi coliision* dalam *game fly bee* seperti pada object garis *unvisible*, garis *unvisible* adalah garis yang tidak terlihat dan ditabrak *bee* ketika melewati pipa1 atau pipa2 dengen begitu *score* akan bertambah, kode programnya sebagai berikut :

```
instance_create(x,y-50,obj_score_up);
sprite_index = sp_smiling_bee;
image_speed = 0.3;
```

pada kode program diatas, jika *collision* antara *bee* dengan garis *unvisible*, maka akan tercipta object baru yaitu *obj_score_up* dan *obj_bee* berubah menjadi *sp_smiling_bee*, dapat dilihat pada Gambar 6.6.



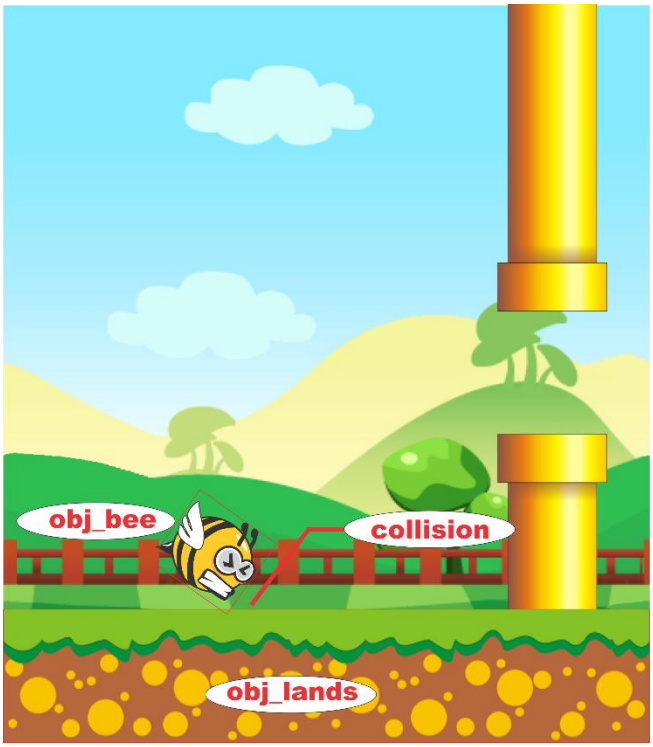
Gambar 6.6 tabrakan obj_bee dengan garis unvisible

4. kondisi saat obj_bee bertabrakan obj_land

saat kondisi obj_bee bertabrakan dengan obj_land, agar tidak melewati batas bawah, maka diberi obj_land kode programnya dapat dilihat sebagai berikut:

```
gravity = 0;
vspeed = 0;
sprite_index = sp_falling_bee;
image_speed = 0;
y = other.y;
fly = false;
if !instance_exists(obj_highscore)
{
instance_create(view_xview[0]+384,view_yview[0]-
50,obj_highscore);
}
```

pada kode program diatas, ketika obj_bee bersentuhan dengan obj_land, maka gravity (grafitasi) nilainya dijadikan 0 agar jatuhnya bee tidak melewati obj_land. Dan untuk memastikan obj_bee benar dan berada tepat di atas obj_land, maka vspeed nya juga dijadikan 0, dan spritenya dirubah menjadi sprite sp_falling_bee, agar spritenya berhenti bergerak, maka image_speed dijadikan 0. Selanjutnya ketika obj_bee berada tepat di atas obj_land (bukan ditengahnya), maka kordinat Y obj_bee disamakan dengan kordinat Y pada obj_land, Kemudian pada saat itu juga dilakukan pengecekan, jika belum ada obj_highscore (belum muncul) maka create (ciptakan) obj_highscore, tujuan pengecekan (if !instance_exists) tersebut agar obj_highscore tidak lebih dari 1. Dapat dilihat juga pada Gambar 6.7.



Gambar 6.7 tabrakan obj_bee dengan obj_land

BAB VII

KESIMPULAN DAN SARAN

7.1 Kesimpulan

Dari pembuatan program ini dapat ditarik kesimpulan sebagai berikut

a. Aplikasi *fly bee* di buat dengan aplikasi *game maker studio* v1.4 sehingga mudah bagi programmer
b. *Game fly bee* berbeda dengan yang lain *endless game* lain karena di dalamnya terdapat sistem score
c. Permainan sederhana dan penampilan yang baik membuat *game* ini unik
d. Permaian *fly bee* mengimplementasikan *collision detectio*n dalam metode pembuatannya

7.2 Saran

Pada pembuatan aplikasi *fly bee* terdapat kekurangan yang masih banyak, semoga dengan saran ini dapat membatu mengembangkan aplikasi *fly bee* yang lebih baik lagi, berikuta beberapa saran pembuatan aplikasi *game* ini :

a. Pada *interface fly bee* masih menggunakan 2d akan lebih baik jika mengunakan *interface* 3d.
b. Untuk menyempurnakan *game* ini bisa ditambah *level* serta lingkup *game* yang berbeda dari segi *background* atau diperbanyak musuh bukan hanya pipa.
c. Metode *collision detection* akan lebih baik di luaskan seperti menyangkut *quad tree* atau masalah *bounding box* dan *circle box* yang lainnya dalam mengunakan game maker.

DAFTAR PUSTAKA

Arsandi, A., SN, S. M., & Hariadi, M. (2012). VISUALISASI GERAKAN OBJEK 3D PADA AUGMENTED REALITY DENGAN DETEKSI TUMBUKAN BERBASIS BOUNDING BOX. *Pasca Sarjana Jaringan Cerdas Multimedia (Game Teknologi) Teknik Elektro,Teknologi Industri ITS*, 1-10.

Ganni, E., & Tendean, H. (2007). PERANCANGAN GAME LIVE FOR GYM. *JURNAL ILMU KOMPUTER DAN SISTEM INFORMASI*, 1-11.

Handoyo, E. D. (2010). Pembuatan Permainan Super Noseman. 1-11.

Haryono, A. N., Hendro, S., & Setiawan. (2010). penggunaan struktur data Quad-Tree dalam Algoritma Collision Detection pada Vertical Shooter Game penggunaan struktur data Quad-Tree dalam Algoritma Collision Detection pada Vertical Shooter Game penggunaan struktur data Quad-Tree dalam Algoritma Collisio. *JURNAL INFORMATIKA , VOLUME 6 NO 1 ,TAHUN 2010*, 1-10.

KOMPUTER, T. L. (2015). *Membuat Game 2 Dimensi Dengan Game Maker Studio.* Semarang: CV Andika Offset.

Mayendra, M. T., & Fadhli, M. (2013). Simulasi Animasi Tiga Dimensi Gerombolan Ikan dalam Akuarium Virtual Menggunakan Algoritma Artificial Bee Colony dan Bounding Box Collision Detection. *150 Jurnal Teknik Elektro dan Komputer, Vol.I, No.2, Oktober 2013, 150-160,* 15-160.

Musfiroh, L., Jazuli, A., & Latubessy, A. (2014). PENERAPAN ALGORITMA COLLISION DETECTION DAN BOIDS PADA GAME DOKKAEBI SHOOTER. *Prosiding SNATIF Ke-1 Tahun 2014 ISBN: 978-602-1180-04-4*, 1-8.

Putrady, E. (2011). Optimasi Collision Detection Menggunakan Quadtree. *Makalah IF3051 Strategi Algoritma – Sem. I Tahun 2010/2011*, 1-5.

Sibero, I. C. (2010). *Membuat Game 2D Menggunakan Game Maker.* Yogyakarta: MediaKom.

Susilawati. (2014). PERANCANGAN GAME SPACE SHIP DENGAN METODE QUAD TREE. *Pelita Informatika Budi Darma, Volume : ViI, Nomor: 3, Agustus 2014 ISSN : 2301-9425*, 1-5.

Tulleken, H. (2005). *Basic Collision Detection in 2D – Part 1 _ Dev.Mag*. Retrieved from http://devmag.org.za/: http://devmag.org.za/2009/04/13/basic-collision-detection-in-2d-part-1/

Winarno, E. &. (2011). *Membuat Sendiri Aplikasi Android untuk Pemula.* Elex Media Komputindo.

LAMPIRAN

MEMBUAT SPRITE

Sprite flying bee (terbang)

1. Klik kanan sprite => create Sprite
2. Load sprite => sp_flying_bee =>open
3. Edit sprite=> add a sprite from file
4. Modify mask => bounding box =>automatic => shape =>precise

Sprite smilling bee (tersenyum)

1. Klik kanan sprite => create Sprite
2. Load sprite => sp_smilling_bee =>open
3. Edit sprite=> add a sprite from file
4. Modify mask => bounding box =>automatic => shape =>precise

Sprite Garis unvisible

1. Klik kanan sprite => create Sprite
2. Load sprite => spr_garis_unvisible =>open
3. Collision checking => precise collision chekhing
4. Edit sprite=> add a sprite from file
5. Modify mask => bounding box =>automatic => shape =>precise

Sprite land

1. Klik kanan sprite => create Sprite
2. Load sprite => sp_land=>open

3. Modify mask => bounding box =>automatic => shape =>rectangle

Sprite score up

1. Klik kanan sprite => create Sprite
2. Load sprite => sp_score_up=>open
3. Edit sprite=> add a sprite from file
4. Modify mask => bounding box =>automatic => shape =>precise

Sprite pipa

1. Klik kanan sprite => create Sprite
2. Load sprite => sp_pipa=>open
3. Collision chekhing => precise collision chekhing =>separate collision mask
4. Modify mask => bounding box =>automatic => shape =>precise

Sprite efek sentuh

1. Klik kanan sprite => create Sprite
2. Load sprite => sp_effect_sentuhn=>open
3. Edit sprite=> add a sprite from file
4. Modify mask => bounding box =>automatic => shape =>precise

Folder menu

1. Klik kanan sprite => creat group
2. Name=> menu => ok

Sprite judul

1. Klik kanan menu => create Sprite
2. Load sprite => sp_judul=>open
3. Modify mask => bounding box =>automatic => shape =>rectangle

Sprite play

1. Klik kanan menu => create Sprite
2. Load sprite => sp_judul=>open
3. Edit sprite=> add a sprite from file
4. Modify mask => bounding box =>manual => left : 86 rigth : 184 top : 20 bottom:55 shape =>rectangle

Sprite about

1. Klik kanan menu => create Sprite
2. Load sprite => sp_about=>open
3. Edit sprite=> add a sprite from file
4. Modify mask => bounding box =>manual => left : 70 rigth : 198 top : 22 bottom:56 shape =>rectangle

Sprite tutorial

1. Klik kanan menu => create Sprite
2. Load sprite => sp_tutorial=>open
3. Edit sprite=> add a sprite from file
4. Modify mask => bounding box =>manual => left : 36 rigth : 232 top : 20 bottom:57 shape =>rectangle

Sprite back

1. Klik kanan menu => create Sprite
2. Load sprite => sp_back=>open
3. Edit sprite=> add a sprite from file
4. Modify mask => bounding box =>manual => left : 66 rigth : 209 top : 20 bottom:57 shape =>rectangle

Sprite exit

1. Klik kanan menu => create Sprite
2. Load sprite => sp_exit =>open
3. Edit sprite=> add a sprite from file
4. Modify mask => bounding box =>manual => left : 68 rigth : 200 top : 20 bottom:57 shape =>rectangle

Sprite get ready

1. Klik kanan menu => create Sprite
2. Load sprite => sp_get_ready=>open
3. Modify mask => bounding box =>automatic => shape =>rectangle

Sprite tap (sentuh)

1. Klik kanan menu => create Sprite
2. Load sprite => sp_tap=>open
3. Modify mask => bounding box =>automatic => shape =>rectangle

Sprite mouse

1. Klik kanan menu => create Sprite
2. Load sprite => sp_mouse =>open
3. Collision chekhing => precise collision chekhing =>separate collision mask
4. Edit sprite=> add a sprite from file
5. Modify mask => bounding box =>automatic => shape =>precise

Sprite bingkai

1. Klik kanan menu => create Sprite
2. Load sprite => sp_bingkai =>open
3. Modify mask => bounding box =>automatic => shape =>rectangle

Sprite game over

1. Klik kanan menu => create Sprite
2. Load sprite => sp_bingkai =>open
3. Modify mask => bounding box =>automatic => shape =>rectangle

MEMBUAT SOUND

Musik latar cara main dan tentang

1. Klik kanan sounds =>create sound
2. Rename snd_tutorial_dan_about
3. Buka sound dalam folder => snd_tutorial_dan_about=> open => ok

Musik latar main 1 (game play1)

1. Klik kanan sounds =>create sound
2. Rename snd_gameplay1
3. Buka sound dalam folder => snd_gameplay1=> open => ok

Musik latar main 2 (game play 2)

1. Klik kanan sounds =>create sound
2. Rename snd_gameplay2
3. Buka sound dalam folder => snd_gameplay2=> open => ok

Sound saat terbang

1. Klik kanan sounds =>create sound
2. Rename snd_up
3. Buka sound dalam folder => snd_up=> open => ok

Sound saat click

1. Klik kanan sounds =>create sound
2. Rename snd_click
3. Buka sound dalam folder => snd_click=> open => ok

Musik game over

1. Klik kanan sounds =>create sound
2. Rename snd_game_over
3. Buka sound dalam folder => snd_game_over => open => ok

Musik score up

1. Klik kanan sounds =>create sound
2. Rename snd_score_up_over
3. Buka sound dalam folder => snd_score_up => open => ok

MEMBUAT BACKGROUND

Background back 1

1. Klik kanan pada background =>create background
2. Rename back1
3. Load background => masukan gambar back 1 => open
4. ok

Background tentang (about)

1. Klik kanan pada background =>create background
2. Rename back_about
3. Load background => masukan gambar back_about=> open
4. Ok

Background cara main (tutorial)

1. Klik kanan pada background =>create background
2. Rename back_tutorial
3. Load background => masukan gambar back_tutorial=> open
4. Ok

Background menu

1. Klik kanan pada background =>create background
2. Rename back_menu
3. Load background => masukan gambar back_menu=> open
4. Ok

MEMBUAT FONT

Font 2

1. Klik kana font => create font
2. Nama => Tahoma_Bold_36=>font =>tahoma=> ok
3. Style => size => 36
4. ok

Font 3

1. Klik kana font => create font
2. Nama => Comic_Sans_Bold_22=>font => Comic Sans Ms => ok
3. Style => size => 22
4. ok

MEMBUAT OBJECT

Membuat folder menu dalam object

1. klik kanan objects => create group
2. rename menu => ok

Membuat folder player dalam object

1. klik kanan objects => create group
2. rename player => ok

Membuat folder room object dalam object

1. klik kanan objects => create group
2. rename roomobject => ok

Membuat folder anytihng dalam object

1. klik kanan objects => create group
2. rename anything => ok

Membuat object play

1. objects=>klik kanan folder menu =>create object
2. rename obj_play =>sprite add =>sp_play => visible
3. add event => create => action => control => drop down exute code pada action
4. create =>action pada exute code ketik :

   mage_speed = 0; // kecepatan image = 0 artinya image tdk berganti-ganti

   image_index = 0; // index image = 0 artinya image yg ditampilkan adalah image 0

   hspeed = 30; // bergerak ke kanan dgn kecepatan 30

   move = true; // Membuat variabel sendiri, move = true

   if !sound_isplaying(snd_main_menu) // Jika snd_menu_utama tidak dimainkan

{

sound_loop(snd_main_menu); // Mainkan snd_menu_utama secara berulang-ulang

}

if sound_isplaying(snd_gameplay1) // Jika snd_gameplay1 sdg dimainkan

{

sound_stop(snd_gameplay1); // hentikan snd_gameplay1

}

if sound_isplaying(snd_gameplay2) // Jika snd_gameplay2 sdg dimainkan

{

sound_stop(snd_gameplay2); // hentikan snd_gameplay2

}

if sound_isplaying(snd_tutorial_dan_about) // Jika snd_tutorial_dan_about sdg dimainkan

{

sound_stop(snd_tutorial_dan_about); // hentikan snd_tutorial_dan_about

}

5. add event => step => action => control => drop down exute code pada action
6. step =>action pada exute code ketik :

if move == true // jika move bernilai true,

{

if x > 370 && x < 384 // jika koordinat x lebih dari 370 dan kurang 384

```
    {
        x = 384; // maka koordinat x sama dengan 384
        hspeed = 0; // kecepatan bergerak ke kanan jadi 0 (berhenti bergerak)
        move = false; // variabel move menjadi bernilai false
    }
}else // selain itu (kebalikan dari nilai variabel move yaitu false) artinya jika move bernilai false
{
    if x > 904 // jika koordinat x lebih dari 904
    {
        room_goto(Stage_1); // room berpindah ke room Stage_1
        instance_destroy(); // object di hancurkan
    }
}
if place_meeting(x,y,obj_mouse) // jika object bertemu di koordinat x dan y nya dengan object obj_mouse
{
    image_index = 1; // image yg ditampilkan adalah image 1 (cek di sprite editor)
    if mouse_check_button_pressed(mb_left) // jika mouse kiri ditekan
    {
        hspeed = 30; // bergerak ke kanan dgn kecepatan 30
        with(obj_exit) { vspeed = 30; } // memerintahkan obj_exit utk bergerak ke bawah dgn kecepatan 30
```

with(obj_about) { vspeed = 30; } // memerintahkan obj_about utk bergerak ke bawah dgn kecepatan 30

with(obj_tutorial) { vspeed = 30; } // memerintahkan obj_tutorial utk bergerak ke bawah dgn kecepatan 30

}

}

else image_index = 0; // selain itu (kebalikannya) atau artinya jika object tidak bertemu di koordinat x dan y nya dengan object obj_mouse

if position_meeting(mouse_x,mouse_y,obj_play) // jika koordinat mouse x dan mouse y berada di posisi object obj_play

{

image_index = 1; // maka tampilkan image 1

}

7. add event => left pressed => action => control => drop down exute code pada action
8. left presed =>action pada exute code ketik :

if move == false // jika variabel move bernilai false

{

image_index = 1; // tampilkan image 1

hspeed = 30; // bergerak ke kanan dgn kecepatan 30

with(obj_exit) { vspeed = 30; } // memerintahkan obj_exit utk bergerak ke bawah dgn kecepatan 30

with(obj_about) { vspeed = 30; } // memerintahkan obj_about utk bergerak ke bawah dgn kecepatan 30

with(obj_tutorial) { vspeed = 30; } // memerintahkan obj_tutorial utk bergerak ke bawah dgn kecepatan 30

}

Membuat objects tutorial (cara main)

1. objects=>klik kanan folder menu =>create object
2. rename obj_tutorial =>sprite add =>sp_tutorial => visible
3. add event => create => action => control => drop down exute code pada action
4. create =>action pada exute code ketik :

   image_speed = 0; // kecepatan image = 0 artinya image tdk berganti-ganti

   image_index = 0;

   hspeed = -30; // bergerak ke kiri dgn kecepatan 30

   move = true; // index image = 0 artinya image yg ditampilkan adalah image 0
5. add event => step => action => control => drop down exute code pada action
6. step =>action pada exute code ketik :

   if move == true

   {

   if x < 400 && x > 383

   {

   x = 384;

```
        hspeed = 0;
        move = false;
    }
}
else
{
    if x < -128
    {
        room_goto(Room_Tutorial);
        instance_destroy();
    }
}
if place_meeting(x,y,obj_mouse)
{
    image_index = 1;
    if mouse_check_button_pressed(mb_left)
    {
        hspeed = -30;
        with(obj_play) { vspeed = 30; }
        with(obj_about) { vspeed = 30; }
        with(obj_exit) { vspeed = 30; }
    }
}
else image_index = 0;
```

```
if position_meeting(mouse_x,mouse_y,obj_tutorial)
{
   image_index = 1;
}
```

7. add event => left presed => action => control => drop down exute code pada action
8. left presed =>action pada exute code ketik :

```
if move == false
{
   image_index = 1;
   hspeed = -30;
   with(obj_play) { vspeed = 30; }
   with(obj_about) { vspeed = 30; }
   with(obj_exit) { vspeed = 30; }
}
```

Membuat object about (tentang)

1. objects=>klik kanan folder menu =>create object
2. rename obj_about =>sprite add =>sp_about => visible
3. add event => create => action => control => drop down exute code pada action
4. create =>action pada exute code ketik :

```
image_speed = 0;
image_index = 0;
hspeed = 30;
```

```
move = true;
```

5. add event => step => action => control => drop down exute code pada action

6. step =>action pada exute code ketik :

```
if move == true
{
   if x > 370 && x < 384
   {
      x = 384;
      hspeed = 0;
      move = false;
   }
}
else
{
   if x > 904
   {
      room_goto(Room_About);
      instance_destroy();
   }
}
if place_meeting(x,y,obj_mouse)
{
   image_index = 1;
```

```
    if mouse_check_button_pressed(mb_left)
    {
        hspeed = 30;
        with(obj_play) { vspeed = 30; }
        with(obj_tutorial) { vspeed = 30; }
        with(obj_exit) { vspeed = 30; }
    }
}
else image_index = 0;
if position_meeting(mouse_x,mouse_y,obj_about)
{
    image_index = 1;
}
```

7. add event => left presed => action => control => drop down exute code pada action
8. left pressed =>action pada exute code ketik :

```
if move == false
{
    image_index = 1;
    hspeed = 30;
    with(obj_play) { vspeed = 30; }
    with(obj_tutorial) { vspeed = 30; }
    with(obj_exit) { vspeed = 30; }
}
```

Membuat objects exit (keluar)

1. objects=>klik kanan folder menu =>create object
2. rename obj_exit =>sprite add =>sp_exit => visible
3. add event => create => action => control => drop down exute code pada action
4. create =>action pada exute code ketik :

   image_speed = 0;

   image_index = 0;

   hspeed = -30;

   move = true;
5. add event => step => action => control => drop down exute code pada action
6. step =>action pada exute code ketik :

```
if move == true
{
   if x < 400 && x > 383
   {
      x = 384;
      hspeed = 0;
      move = false;
   }
```

```
}
else
{
   if x < -128
   {
      game_end();
   }
}
if place_meeting(x,y,obj_mouse)
{
   image_index = 1;
   if mouse_check_button_pressed(mb_left)
   {
      hspeed = -30;
      with(obj_play) { vspeed = 30; }
      with(obj_about) { vspeed = 30; }
      with(obj_tutorial) { vspeed = 30; }
   }
}
```

```
else image_index = 0;

if position_meeting(mouse_x,mouse_y,obj_exit)

{

   image_index = 1;

}
```

7. add event => left presed => action => control => drop down exute code pada action
8. left pressed =>action pada exute code ketik :

```
if move == false

{

   image_index = 1;

   hspeed = -30;

   with(obj_play) { vspeed = 30; }

   with(obj_about) { vspeed = 30; }

   with(obj_tutorial) { vspeed = 30; }

}
```

Membuat object back (kembali)

1. objects=>klik kanan folder menu =>create object
2. rename obj_back =>sprite add =>sp_back => visible
3. add event => create => action => control => drop down exute code pada action
4. create =>action pada exute code ketik

```
image_speed = 0;
image_index = 0;
vspeed = -30;
move = true;

if !sound_isplaying(snd_tutorial_dan_about) // Jika snd_tutorial_dan_about tidak dimainkan
{
   sound_loop(snd_tutorial_dan_about); // mainkan snd_tutorial_dan_about secara berulang-ulang
}
if sound_isplaying(snd_main_menu) // Jika snd_main_menu sedang dimainkan
{
   sound_stop(snd_main_menu); // hentikan snd_main_menu
}
```

5. add event => step => action => control => drop down exute code pada action
6. step =>action pada exute code ketik

```
if move == true
{
   if y < 940 && y > 927
   {
      y = 928;
```

```
        vspeed = 0;
        move = false;
    }
}
else
{
    if place_meeting(x,y,obj_mouse)
    {
        image_index = 1;
        if mouse_check_button_pressed(mb_left)
        {
            vspeed = 30;
        }
    }
    else { image_index = 0; }
    if y > 1100
    {
        room_goto(Main_Menu);
        instance_destroy();
    }
}
if position_meeting(mouse_x,mouse_y,obj_back)
{
    image_index = 1;
```

}

5. add event => left pressed=> action => control => drop down exute code pada action
6. left pressed =>action pada exute code ketik

   vspeed = 30; // bergerak ke bawah dgn kecepatan 30

   image_index = 1;

Membuat object mouse

1. objects=>klik kanan folder menu =>create object
2. rename obj_mouse=>sprite add =>sp_mouse => visible
3. depth -10
4. add event => create => action => control => drop down exute code pada action
5. create =>action pada exute code ketik :

   image_speed = 0.5; // kecepatan image utk berganti frame adalah 0.5

   window_set_cursor(cr_none); // membuat cursor window jd tdk ada (tdk ditampilkan)

   action_move_to(mouse_x,mouse_y); // selalu bergerak ke koordinat mouse_x dan mouse_y
6. add event => step => action => control => drop down exute code pada action
7. step =>action pada exute code ketik :

   action_move_to(mouse_x,mouse_y); // selalu bergerak ke koordinat mouse_x dan mouse_y

Membuat object highscore

1. objects=>klik kanan folder menu =>create object
2. rename obj_highscore=>sprite add =>sp_game_over => visible
3. depth -1000
4. add event => create => action => move => drop down speed vertical pada action =>applies to => self
5. vert.speed =>40
6. create =>action pada exute code ketik :

   ///Save Score

   if score > hi_score // jika nilai score lebih besar dari nilai hi_score

   {

   if file_exists("save.sav") file_delete("save.sav"); // jika ada file "save.sav" di directory, maka hapus file tsb

   ini_open("save.sav"); // bikin dan buka file "save.sav'

   ini_write_real("Status","Score",score); // menulis "Status","Score", dan nilai score ke dlm file "save.sav"

   ini_close(); // menutup dan menyimpan file "save.sav"

   alarm[1] = 45; // jalankan event alarm[1] setelah 45 step (1,5 detik)

   }

   klik_to_restart = false; // membuat variabel sndiri klik_to_restart bernilai false

   alarm[0] = 90; // menjalankan event alarm[0] setelah 90 step (3 detik)
7. add action => drop down exute code padaaction dan ketik code :

   /// Sound

sound_play(snd_game_over); // mainkan snd_game_over

if sound_isplaying(snd_gameplay1) // Jika snd_gameplay1 sedang dimainkan

{

sound_stop(snd_gameplay1); // hentikan snd_gameplay1

}

if sound_isplaying(snd_gameplay2) // Jika snd_gameplay2 sedang dimainkan

{

sound_stop(snd_gameplay2); // hentikan snd_gameplay2

}

8. add event => create => action => move => drop down speed vertical pada action =>applies to => self
9. vert.speed =>10
10. add event =>alarm 0=> action pada alarm=>drop down exute code
11. ketik pada exute code :

    klik_to_restart = true; // mengubah nilai variabel klik_to_restart yg tadinya false menjadi true
12. add event =>alarm 1=> action pada alarm=>drop down exute code
13. ketik pada exute code :

    hi_score = score; // membuat nilai variabel hi_score menjadi sama dengan nilai variabel score
14. add event => step => action => control => drop down exute code pada action

15. step =>action pada exute code ketik :

```
if klik_to_restart == false // Jika variable klik_to_restart bernilai false
{
   if y >= view_yview[0]+510 // jika koordinat y lebih besar atau sama dgn 510 ditampilan view (kamera)
   {
      action_move_to(view_xview[0]+384,view_yview[0]+530); // maka bergerak ke koordinat 384 dan 530 ditampilan view (kamera)
      vspeed = 0; // pergerakan ke bawah bernilai 0 (berhenti bergerak)
   }
}
if klik_to_restart == true // Jika variable klik_to_restart bernilai true
{
   if y >= view_yview[0]+1400 // jika koordinat y lebih besar atau sama dgn 1400 ditampilan view (kamera)
   {
      room_restart(); // room di ulang dari awal
      instance_destroy(); // instance di hilangkan
   }
}
```

16. add event => left pressed=> action => control => drop down exute code pada action
17. left pressed=>action pada exute code ketik :

```
if klik_to_restart == true // jika variabel klik_to_restart bernilai true
```

{

vspeed = 40; // maka bergerak ke bawah dgn kecepatan 40

}

18. add event => draw => action => control => drop down exute code pada action
19. draw=>action pada exute code ketik :

draw_self(); // menampilkan sprite diri sndiri

draw_set_font(Tahoma_Bold_36); // mengeset font huruf

draw_set_halign(fa_left); // membuat huruf dimulai dari kiri

draw_text(x+102,y+4,string(score)); // menuliskan text di koordinat x+102,dan y+4 dgn tulisan nilai dari score

draw_text(x+102,y+62,string(hi_score)); // menuliskan text di koordinat x+102,dan y+62 dgn tulisan nilai dari hi_score

if klik_to_restart = true && y < view_yview[0]+540 // jika klik_to_restart bernilai true dan koordinat y lebih kecil dari 540 di tampilan view (kamera)

{

draw_text_colour(x-90,y+170,"

sentuh#untuk#cobalagi",c_aqua,c_fuchsia,c_lime,c_green,1);// menuliskan text berwarna di koordinat x-204,dan y+250 dgn tulisan "sentun untuk coba lagi"

}

Membuat object draw text

1. objects=>klik kanan folder menu =>create object
2. rename obj_draw_text=> visible
3. add event => create => action => control => drop down exute code pada action
4. create=>action pada exute code ketik :

   text = ""; // membua variabel sndiri text sama dgn ""
5. add event => step => action => control => drop down exute code pada action
6. step=>action pada exute code ketik :

   if room == Room_About // Jika di Room About

   {

   text = "GAME INI #DIBUAT OLEH : ZUNIAR FARIH FAHLEVY #EMAIL : zuniarfarih@gmail.com# ENJOY THE GAME :)"; // variabel text sama dgn Game ini dibuat untuk mendukung tugas akhir mahasiswa (skripsi) Universitas Stikubank Semarang

   }

   if room == Room_Tutorial // Jika di Room Tutorial

   {

   text = "CARA BERMAIN FLY BEE# #KETUK LAYAR AGAR SI LEBAH TERBANG#LEBAH DIHARUSKAN MELEWATI PIPA-PIPA#PEMAIN MENDAPATKAN SCORE 1 POIN#JIKA BERHASIL MELEWATI PIPA# #SELAMAT BERMAIN ";

   }

7. add event => draw => action => control => drop down exute code pada action
8. draw=>action pada exute code ketik :

   draw_set_font(Comic_Sans_Bold_22);

   draw_set_halign(fa_center);

   draw_text_colour(x,y-180,text,c_aqua,c_aqua,c_white,c_white,1);

Membuat object control

1. objects=>klik kanan folder player =>create object
2. rename obj_control=> visible
3. depth=> -1
4. add event => create => action => control => drop down exute code pada action
5. create=>action pada exute code ketik :

   /// Menginisialisasi Variabel

   globalvar ay,by,cy,dy,ey,fy,gy,hy,iy,jy,ky,ly,my,fly,tap,hi_score; // GlobalVar = Variabel yg dpt dibaca/digunakn oleh semua object

   ay = 128; // ay sama dgn koordinat 128

   by = 192; // by sama dgn koordinat 192

   cy = 256; // cy sama dgn koordinat 256

   dy = 320; // dy sama dgn koordinat 320

   ey = 384; // ey sama dgn koordinat 384

   fy = 448; // fy sama dgn koordinat 448

   gy = 512; // gy sama dgn koordinat 512

   hy = 576; // hy sama dgn koordinat 576

```
iy = 640; // iy sama dgn koordinat 640
jy = 704; // jy sama dgn koordinat 704
ky = 768; // ky sama dgn koordinat 768
ly = 832; // ly sama dgn koordinat 832
my = 896; // my sama dgn koordinat 896
fly = true; // fly bernilai true
tap = 0; // tap bernilai 0
hi_score = 0; // hi_score bernilai 0
score = 0; // score bernilai 0
```

6. drop down exute code pada action create
7. Pada action pada exute code ketik :

```
/// Load
if file_exists("save.sav") // jika ada file "save.sav" di directory
{
   ini_open("save.sav"); // buka file save.sav
   hi_score = ini_read_real("Status","Score",0); // membaca file save.sav dan membuat nilai hi_score sama dgn nilai yg ditertulis di file tsb
   ini_close(); // menutup dan menyimpan data/tulisan di file save.sav
}
```

8. drop down exute code pada action create
9. Pada action pada exute code ketik :

```
/// Sound
sound_loop(choose(snd_gameplay1,snd_gameplay2)); // pilih salah satu snd_gameplay1 atau snd_gameplay2 utk dimainkan secara berulang-ulang
```

if sound_isplaying(snd_main_menu) // Jika snd_main_menu sedang dimainkan

{

sound_stop(snd_main_menu); // hentikan snd_main_menu

}

10. add event => step => action => control => drop down exute code pada action
11. step =>action pada exute code ketik :

if tap = 0 || tap = 1 // Jika variabel tap bernilai 0 atau bernilai 1

{

fly = false; // maka variabel fly bernilai false

}

if tap == 2 // jika variabel tap bernilai 2

{

fly = true; // maka fly bernilai true

instance_create(1024,choose(cy,dy,ey,fy,gy,hy,iy,jy),obj_pipa_1); // menciptakan object baru yaitu obj_pipa_1 di koordinat x = 1024 dan y dipilih/ditentukan secara acak

instance_create(1024,choose(cy,dy,ey,fy,gy,hy,iy,jy),obj_garis_unvisible); // menciptakan object baru yaitu obj_garis_unvisible di koordinat x = 1024 dan y dipilih/ditentukan secara acak

instance_create(1460,choose(cy,dy,ey,fy,gy,hy,iy,jy),obj_pipa_2);

```
instance_create(1460,choose(cy,dy,ey,fy,gy,hy,iy,jy),obj_garis_unvisible);
   instance_create(2145,choose(cy,dy,ey,fy,gy,hy,iy,jy),obj_pipa_1);

instance_create(2145,choose(cy,dy,ey,fy,gy,hy,iy,jy),obj_garis_unvisible);
   instance_create(2878,choose(cy,dy,ey,fy,gy,hy,iy,jy),obj_pipa_2);

instance_create(2878,choose(cy,dy,ey,fy,gy,hy,iy,jy),obj_garis_unvisible);
   tap = 3; // tap bernilai 3
}
if tap < 2 // jika tap bernilai lebih kecil dari 2
{
   if mouse_check_button_pressed(mb_left) // jika mouse kiri dipencet
   {
      tap += 1; // nilai tap di tambah 1
   }
}
if fly == true // jika fly bernilai true
{
   background_hspeed[0] = -1; // background bergerak ke kiri dgn
kecepatan 1
}
else // sebaliknya
```

background_hspeed[0] = 0; // background bergerak dgn kecepatan 0 (berhenti)

9. add event =>backspace =>action=>main1=>dropdown go to room=> main_menu
10. add event => draw => action => control => drop down exute code pada action
11. draw =>action pada exute code ketik :

    draw_set_font(Comic_Sans_Bold_32);

    draw_set_halign(fa_left);

    if tap == 0 // jika tap bernilai 0

    {

      draw_sprite(sp_get_ready,0,view_xview+384,view_yview+530); // maka tampilkan sprite sp_get_ready di koordinat 384 dan 530 pada tampilan view (kamera)

    }

    if tap == 1 // jika tap bernilai 1

    {

      draw_sprite(sp_tap,0,view_xview+384,view_yview+530); // maka tampilkan sprite tap di koordinat 384 dan 530 pada tampilan view (kamera)

    }

    if tap >= 2 && !instance_exists(obj_highscore) // jika tap bernilai lebih dari atau sama dgn 2 dan tdk ada object obj_highscore

{

draw_roundrect_colour(view_xview[0]+256,view_yview[0]+16,view_xview[0]+504,view_yview[0]+92,c_gray,c_dkgray,false); // maka tampilkan gambar segi4 berwarna abu-abu di koordinat dgn tanpa garis tepi

draw_text_colour(view_xview[0]+266,view_yview[0]+20,"Score : "+string(score),c_aqua,c_aqua,c_fuchsia,c_fuchsia,1); // tampilkan text berwarna dgn tulisan Score : dan nilai dari variable score tsb

}

Membuat object bee

1. objects=>klik kanan folder player =>create object
2. rename obj_bee=> add sprite => sp_flying_bee=> visible
3. depth=> -100
4. add event => create => action => control => drop down exute code pada action
5. create=>action pada exute code ketik :

   image_speed = 0.3; // kecepatan image per frame adalah 0.3
6. add event => step => action => control => drop down exute code pada action
7. step=>action pada exute code ketik :

   if fly == true // Jika fly bernilai true

   {

   gravity = 2; // set gravitasi bernilai 2

   gravity_direction = 270; // set arah gravitasi ke bawah

   if mouse_check_button(mb_left) // jika mouse kiri ditekan

```
    {
        vspeed = -10; // maka bergerak ke atas dgn kecepatan 10
    }
    if mouse_check_button_pressed(mb_left) // jika mouse kiri ditekan
    {
        sound_play(snd_click); // mainkan snd_click
    }
}
```

8. add event => coliision=>room objetct=> obj_pipa_1 => action => control => drop down exute code pada action
9. collision=>action pada exute code ketik :

```
if fly == true // jkia fly bernilai true
{
    instance_create(x+44,y,obj_effect_sentuhan); // maka ciptakn object baru yaitu obj_effect_sentuhan di koordinat x+44 dan y
    sprite_index = sp_falling_bee; // ubah sprite menjadi sp_falling_bee
    fly = false; // variable fly menjadi false
    sound_play(snd_up); // mainkan snd_up tnpa diulang-ulang
}
```

10. add event => coliision=>room objetct=> obj_pipa_2 => action => control => drop down exute code pada action
11. collision=>action pada exute code ketik :

```
/// penjelasan nya sama sprit event sebelumnya
if fly == true
```

```
{
   instance_create(x+44,y,obj_effect_sentuhan);
   sprite_index = sp_falling_bee;
   fly = false;
   sound_play(snd_up);
}
```

12. add event => coliision=>room objetct=> obj_land => action => control => drop down exute code pada action
13. collision=>action pada exute code ketik :

```
gravity = 0; // set gravitasi menjadi bernilai 0
vspeed = 0; // pergerakan ke atas/ke bawah dgn nilai 0 (berhenti)
sprite_index = sp_falling_bee; // ubah sprite menjadi sp_falling_bee
image_speed = 0; // kecepatan image berubah frame menjadi 0 (berhenti mengubah frame image)
y = other.y; // koordinat y sama dgn koordinat y obj_land (note : other maksud nya adalah obj_land)
fly = false; // fly bernilai false

if !instance_exists(obj_highscore) // jika obj_highscore belum ada
{
   instance_create(view_xview[0]+384,view_yview[0]-50,obj_highscore);
// maka ciptakan object baru yaitu obj_highscore
}
```

12. add event => coliision=>room objetct=> obj_garis_unvisible => action => control => drop down exute code pada action
13. collision=>action pada exute code ketik :

    instance_create(x,y-50,obj_score_up); // ciptakan object baru yaitu obj_score_up

    sprite_index = sp_smiling_bee; // sprite berubah menjadi sp_smiling_bee

    image_speed = 0.3; // kecepatan image per frame adalah 0.3
14. add event =>keyboard=>letter=>E=> action => main2 => drop down end the game pada action
15. add event =>keyboard=>letter=>R=> action => main1 => drop down restat the current room pada action

Membuat object pipa 1

1. objects=>klik kanan folder object room =>create object
2. rename obj_pipa_1=> add sprite => sp_pipa=> visible
3. depth=> 10
4. add event => step => action => control => drop down exute code pada action
5. step=>action pada exute code ketik :

    /// Pergerakan

    if fly == true // jika fly benilai true

    {

    if score >= 0 && score < 10 // jika score bernilai antara 0 smpai 9

    {

    hspeed = -3; // bergerak ke kiri dgn kecepatan 3

```
}
if score > 9 && score < 20 // jika score bernilai antara 10 smpai 19
{
   hspeed = -4; // bergerak ke kiri dgn kecepatan 4
}
if score > 19 && score < 30 // jika score bernilai antara 20 smpai 29
{
   hspeed = -5; // bergerak ke kiri dgn kecepatan 5
}
if score > 29 && score < 40 // jika score bernilai antara 30 smpai 39
{
   hspeed = -6; // bergerak ke kiri dgn kecepatan 6
}
if score > 39 && score < 50 // jika score bernilai antara 40 smpai 49
{
   hspeed = -7; // bergerak ke kiri dgn kecepatan 7
}
if score > 49 && score < 60 // jika score bernilai antara 50 smpai 59
{
   hspeed = -8; // bergerak ke kiri dgn kecepatan 8
}
if score > 59 && score < 70 // jika score bernilai antara 60 smpai 69
{
   hspeed = -9; // bergerak ke kiri dgn kecepatan 9
```

```
    }
    if score > 69 && score < 80 // jika score bernilai antara 70 smpai 79
    {
        hspeed = -10; // bergerak ke kiri dgn kecepatan 10
    }
}
else // sebaliknya (jika fly bernilai false)
hspeed = 0;  // bergerak horizontal dgn kecepatan 0 (berhenti)

if x < -100 // jika nilai koordinat x lebih kecil dari -100 koordinat room
{
    x = 2880; // maka nilai x melompat ke koordinat 2880
    y = choose(cy,dy,ey,fy,gy,hy,iy,jy); // koordinat y dirandom secara acak
}
if x == 2880 // jika nilai koordinat x sama dgn 2880 koordinat room
{
    instance_create(x,y,obj_garis_unvisible); // maka ciptakan object baru
yaitu obj_garis_unvisible
}
```

Membuat object pipa 2

1. objects=>klik kanan folder object room =>create object
2. rename obj_pipa_2=> add sprite => sp_pipa=> visible
3. depth=> 10

4. add event => step => action => control => drop down exute code pada action
5. step=>action pada exute code ketik:

```
/// Pergerakan
if fly == true // jika fly benilai true
{
   if score >= 0 && score < 10 // jika score bernilai antara 0 smpai 9
   {
      hspeed = -3; // bergerak ke kiri dgn kecepatan 3
   }
   if score > 9 && score < 20 // jika score bernilai antara 10 smpai 19
   {
      hspeed = -4; // bergerak ke kiri dgn kecepatan 4
   }
   if score > 19 && score < 30 // jika score bernilai antara 20 smpai 29
   {
      hspeed = -5; // bergerak ke kiri dgn kecepatan 5
   }
   if score > 29 && score < 40 // jika score bernilai antara 30 smpai 39
   {
      hspeed = -6; // bergerak ke kiri dgn kecepatan 6
   }
   if score > 39 && score < 50 // jika score bernilai antara 40 smpai 49
   {
```

```
        hspeed = -7; // bergerak ke kiri dgn kecepatan 7
    }
    if score > 49 && score < 60 // jika score bernilai antara 50 smpai 59
    {
        hspeed = -8; // bergerak ke kiri dgn kecepatan 8
    }
    if score > 59 && score < 70 // jika score bernilai antara 60 smpai 69
    {
        hspeed = -9; // bergerak ke kiri dgn kecepatan 9
    }
    if score > 69 && score < 80 // jika score bernilai antara 70 smpai 79
    {
        hspeed = -10; // bergerak ke kiri dgn kecepatan 10
    }
}
else // sebaliknya (jika fly bernilai false)
hspeed = 0;  // bergerak horizontal dgn kecepatan 0 (berhenti)

if x < -100 // jika nilai koordinat x lebih kecil dari -100 koordinat room
{
    x = 2880; // maka nilai x melompat ke koordinat 2880
    y = choose(cy,dy,ey,fy,gy,hy,iy,jy); // koordinat y dirandom secara acak
}
if x == 2880 // jika nilai koordinat x sama dgn 2880 koordinat room
```

{

instance_create(x,y,obj_garis_unvisible); // maka ciptakan object baru yaitu obj_garis_unvisible

}

Membuat object land

1. objects=>klik kanan folder object room =>create object
2. rename obj_land=> add sprite => sp_land=> visible=>solid
3. depth=> 0
4. add event => create => action => control => drop down exute code pada action
5. create=>action pada exute code ketik:

   hspeed = 0;
6. add event => step=> action => control => drop down exute code pada action
7. step =>action pada exute code ketik :

   if fly == false

   {

   hspeed = 0;

   }

   else hspeed = -3;

   if x < -524

   {

   x = 2612;

```
}
```

Membuat garis unvisible

1. objects=>klik kanan folder object room =>create object
2. rename obj_garis_unvisible=> add sprite => spr_garis_unvisible
3. depth=> 0
4. add event => step => action => control => drop down exute code pada action
5. step =>action pada exute code ketik:

```
/// Pergerakan
if fly == true // jika fly benilai true
{
   if score >= 0 && score < 10 // jika score bernilai antara 0 smpai 9
   {
      hspeed = -3; // bergerak ke kiri dgn kecepatan 3
   }
   if score > 9 && score < 20 // jika score bernilai antara 10 smpai 19
   {
      hspeed = -4; // bergerak ke kiri dgn kecepatan 4
   }
   if score > 19 && score < 30 // jika score bernilai antara 20 smpai 29
   {
      hspeed = -5; // bergerak ke kiri dgn kecepatan 5
   }
   if score > 29 && score < 40 // jika score bernilai antara 30 smpai 39
```

```
    {
       hspeed = -6; // bergerak ke kiri dgn kecepatan 6
    }
    if score > 39 && score < 50 // jika score bernilai antara 40 smpai 49
    {
       hspeed = -7; // bergerak ke kiri dgn kecepatan 7
    }
    if score > 49 && score < 60 // jika score bernilai antara 50 smpai 59
    {
       hspeed = -8; // bergerak ke kiri dgn kecepatan 8
    }
    if score > 59 && score < 70 // jika score bernilai antara 60 smpai 69
    {
       hspeed = -9; // bergerak ke kiri dgn kecepatan 9
    }
    if score > 69 && score < 80 // jika score bernilai antara 70 smpai 79
    {
       hspeed = -10; // bergerak ke kiri dgn kecepatan 10
    }
  }
  else // sebaliknya (jika fly bernilai false)
  hspeed = 0;  // bergerak horizontal dgn kecepatan 0 (berhenti)
```

3. add event => collision =>player=>obj_bee => action => control => drop down exute code pada action

4. collision=>action pada exute code ketik:

   score += 1; // nilai variable score ditambah 1

   instance_destroy(); // object di hilangkan

   sound_play(snd_score_up); // mainkan snd_score_up

Membuat object score up

1. objects=>klik kanan folder anything =>create object
2. rename obj_score_up=> add sprite => sp_score_up =>visible
3. depth=> -1000
4. add event => create => action => control => drop down exute code pada action
5. create =>action pada exute code ketik:

   image_speed = 0.5; // kecepatan image per frame adalah 0.5

   vspeed = -1; // bergerak vertikal ke atas dgn kecepatann 1

   alarm[0] = 45; // jalankan event alarm[0] setelah 45 step (1,5 detik)
6. add event => destroy => action => control => drop down comment pada action
7. comment=> mengubah sprite menjadi sp_flying_bee
8. add action pada destroy =>control =>sprite=>applies to=> object =>obj_bee=>sprite=>sp_flying_bee=>subimage=>image_index=>speed= >0,3
9. add event => alram0 => action => control => drop down destroy the instance pada action => applies to=>self

Membuat object effek sentuh

1. objects=>klik kanan folder anything =>create object
2. rename obj_effect_sentuhan => add sprite => sp_effect_sentuh =>visible
3. depth=> -150
4. add event => create => action => control => drop down exute code pada action
5. create =>action pada exute code ketik:

   image_speed = 0.75;
6. add event => other => animation end => drop down destroy the instance pada action=> applies to self

MEMBUAT ROOMS

Membuat room main menu

1. Klik kanan pada room=> create room
2. Rename =>main_menu
3. Klik background =>load background => main_menu
4. Setting=> width :768 higth : 1060 speed : 30
5. Klik object =>object to add with left mouse =>load sprite =>menu =>obj_play => taruh posisi y: 480 x:-144
6. object to add with left mouse =>load sprite =>menu =>obj_tutorial => taruh posisi y: 608 x:880
7. object to add with left mouse =>load sprite =>menu =>obj_about => taruh posisi y: 704 x:-128

8. object to add with left mouse =>load sprite =>menu =>obj_exit => taruh posisi y: 800 x:912
9. ok

Membuat background stage 1

1. Klik kanan pada room=> create room
2. Rename =>back_1
3. Klik background =>load background =>back_1
4. Setting=> width :3000 higth : 1024 speed : 30
5. Klik object =>object to add with left mouse =>load sprite =>menu =>obj_control => taruh posisi y:130 x: 3
6. Klik object =>object to add with left mouse =>load sprite =>menu =>obj_bee => taruh posisi y:252 x: 134
7. Klik object =>object to add with left mouse =>load sprite =>menu =>obj_land => taruh posisi y:1012 x: 766

Membuat background tentang (about)

1. Klik kanan pada room=> create room
2. Rename =>Room_about
3. Klik background =>load background =>back_about
4. Name => Room_about
5. Setting=> width :768 higth : 1060 speed : 30
6. Klik object =>object to add with left mouse =>load sprite =>menu =>obj_back => taruh posisi y:1056 x: 384

7. Klik object =>object to add with left mouse =>load sprite =>menu =>obj_draw_text=> taruh posisi y:512 x: 384

Membuat background cara main (tutorial)

1. Klik kanan pada room=> create room
2. Rename =>Room_tutorial
3. Klik background =>load background =>back_tutorial
4. Name => Room_tutorial
5. Setting=> width :768 higth : 1060 speed : 30
6. Klik object =>object to add with left mouse =>load sprite =>menu =>obj_back => taruh posisi y:1056 x: 384
7. Klik object =>object to add with left mouse =>load sprite =>menu =>obj_draw_text=> taruh posisi y:512 x: 384

LEMBAR BIMBINGAN

NAMA : Zunar Farih Fahley
NIM : 12.01.53.0048
Program Studi : teknik informatika
Jenjang Program : S1
Judul : Implementasi collision detection permainan
: fly bee menggunakan game maker

Tgl	MATERI	SARAN	Paraf
14/4/16	BAB I	Revisi BAB I	
21/4/16	BAB I	~~Revisi~~ Acc BAB I	
21/6/16	BAB II	Acc BAB II	
7/11/16	BAB III	Acc BAB III	
25/11/16	BAB IV	Acc BAB IV	
21/12/16	BAB V	Acc BAB V	
9/1/17	BAB VI	Acc BAB VI	
16/1/17	—	Acc Semua BAB SIAP UJIAN!	

Pembimbing

(Edy Winarno)