BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Game

Analisis *game* merupakan analisis yang dilakukan terhadap *game* yang akan dikembangkan. Analisis dilakukan dengan maksud untuk mengidentifikasi dan mengevaluasi kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya. Analisis yang dibahas mulai dari analisis *storyline*, *gameplay*, *scoring*, pergerakan karakter NPC, *collision detection*, sampai dengan analisis teknologi *augmented reality* terhadap *game*.

Game duck hunt yang akan dikembangkan adalah *game* bertipe *shooting game* yang dimainkan dengan tujuan menembak sasaran yang bergerak. *Game* ini ditujukan untuk pengguna *mobile platform* android. Teknologi *augmented reality* yang diterapkan pada *game* ini berfungsi sebagai media interaksi antara user dengan *game*. *Game* ini dibuat dengan mengambil latar pada lingkungan nyata kemudian digabungkan dengan obyek-obyek 3D yang ditampilkan pada layar perangkat android melalui kamera.

Game ini mempunyai misi untuk menembak sasaran berupa karakter NPC (bebek) yang beterbangan diantara pepohonan untuk mendapatkan *point*. Tempat memunculkan karakter NPC dalam *game* ini ditentukan dengan cara diacak agar menjadikan *game* ini lebih menarik.

Berikut ini merupakan kelebihan *game duck hunt* yang dikembangkan menggunakan teknologi *augmented reality* dibandingkan dengan sebelumnya:

1. *Game* dapat dilihat dari berbagai sudut pandang, tergantung dari arah kamera pada perangkat android.
2. *Gaming experience* akan lebih terasa karena lingkungan nyata dijadikan sebagai latar dari *game* ini.

3.1.1 Storyline

Game Duck Hunt ini menceritakan mengenai seorang pemburu yang ingin menangkap bebek-bebek yang berterbangan bebas di alam liar dengan menggunakan sebuah senapan. Untuk memburu bebek ini, dia harus mengamati setiap pohon yang ada dilingkungan sekitar karena bebek-bebek tersebuat mulai terbang dari satu pohon ke pohon lainnya. Tujuan akhir dari perburuan ini adalah untuk mendapatkan bebek yang sebanyak-banyaknya.

3.1.2 Gameplay

Gameplay merupakan interaksi antara user dengan suatu game yang akan dimainkannya yang berkenaan dengan aturan bermain dari game tersebut. Berikut merupakan gameplay pada game duck hunt:

1. Pemain memilih menu mulai game untuk memainkan game ini.
2. Pemain mengarahkan kamera perangkat android pada marker yang terlah dicetak.
3. Waktu pada game akan mulai berkurang setelah marker berhasil dideteksi oleh game.
4. Pemain menembak sasaran (karakter bebek) yang bergerak terbang melintasi pohon disekitarnya dengan menekan tombol tembak.
5. Setiap sasaran yang tertembak dan jatuh, pemain akan diberikan tambahan skor sebesar 100 poin.
6. Peluru yang disediakan berjumlah 5 butir, jika habis pemain dapat mengisinya dengan cara menekan tombol isi peluru.
7. Waktu yang berikan untuk setiap gamenya adalah 60 detik, jika waktu habis maka permainan akan berakhir.
8. Skor akhir yang didapatkan dalam game akan ditampilkan pada saat permainan berakhir dan kemudian dibandingkan dengan skor tertinggi yang ada sebelumnya.
9. Skor tertinggi akan disimpan di dalam game.

10. Setelah permainan berakhir akan muncul 2 buah tombol pilihan yaitu main lagi atau keluar. Jika pemain memilih tombol main lagi maka permainan akan diulang kembali, sedangkan tombol keluar untuk kembali ke main menu.

3.1.3 Pergerakan Non Player Character

Non Player Character atau yang biasa disebut NPC merupakan karakter di dalam suatu *game* atau simulasi yang perilakunya tidak bisa dikendalikan oleh pemain. Contoh yang paling banyak ditemui dalam suatu *game* adalah musuh yang dihadapi ketika bermain *game*. Keberadaan NPC seringkali menjadi faktor penting dalam menentukan menarik atau tidaknya suatu *game*.

Non Player Character dalam game ini adalah karakter bebek yang merupakan sasaran/target tembak pemain. Pergerakan NPC dalam *game* ini dibuat bervariasi dan dilakukan secara *random* dengan menggunakan algoritma fisher-yates shuffle. Hal ini bertujuan agar pemain tidak dapat memprediksi kapan munculnya dan bagaimana pergerakan NPC sehingga permainan menjadi lebih menarik.

Karakter NPC pada *game duck hunt* memiliki 10 jalur yang berbeda. Pemilihan jalur yang akan dilalui oleh setiap karakter NPC akan dilakukan dengan menggunakan algoritma fisher-yates shuffle. Algoritma tersebut berfungsi untuk melakukan pengacakan elemen pada suatu *array*. Berikut merupakan simulasi yang dilakukan dengan menggunakan algoritma fisher-yates shuffle:

Misalkan jalurNPC = [“p1”, “p2”, “p3”, “p4”, “p5”, “p6”, “p7”, “p8”, “p9”, “p10”]. *Array* tersebut dimasukan kedalam prosedur *shuffle* dimana proses pengacakan terjadi.

Tahapan yang dilakukan dalam pengacakan jalur menggunakan algoritma fisher-yates shuffle adalah sebagai berikut:

1. Ambil satu elemen secara acak dari elemen yang tersisa.

   Pengambilan elemen acak dilakukan berdasarkan elemen yang tersisa, misalkan jika panjang array sama dengan 10, maka elemen acak yang boleh diambil

adalah antara 0 sampai 9 (array[0…9]). Variabel yang digunakan untuk mendapatkan nilai acak adalah variabel r.

2. Tukar dengan elemen saat ini.

   Penukaran yang dilakukan adalah antara array dengan elemen hasil acak (array[r]) terhadap array dengan elemen saat ini (array[i]) yang akan melibatkan satu buah variabel penampung nilai sementara bernama t. Langkah penukarannya adalah sebagai berikut:

   a. t = array[r]
   b. Array[r] = array[i]
   c. Array[i] = t

3. Ulangi selama masih ada elemen yang tersisa.

   Selanjutnya dilakukan pengulangan sebanyak elemen *array* (dalam kasus ini panjang array = 10) kali untuk mengacak urutan jalur. Iterasi dari proses tersebut dapat dilihat pada tabel 3.1

Tabel 3. 1 Iterasi Fisher-Yates Shuffle

i	r	Array[i]	Array[r]	Array yang sudah fix	Isi array setelah ditukar
9	8	p10	p9	p9	p1, p2, p3, p4, p5, p6, p7, p8, p10, p9
8	4	p10	p5	p5, p9	p1, p2, p3, p4, p10, p6, p7, p8, p5, p9
7	1	p8	p2	p2, p5, p9	p1, p8, p3, p4, p10, p6, p7, p2, p5, p9
6	1	p7	p8	p8, p2, p5, p9	p1, p7, p3, p4, p10, p6, p8, p2, p5, p9
5	2	p6	p3	p3, p8, p2, p5, p9	p1, p7, p6, p4, p10, p3, p8, p2, p5, p9
4	1	p10	p7	p7, p3, p8, p2, p5, p9	p1, p10, p6, p4, p7, p3, p8, p2, p5, p9
3	0	p4	p1	p1, p7, p3, p8, p2, p5, p9	p4, p10, p6, p1, p7, p3, p8, p2, p5, p9
2	1	p6	p10	p10, p1, p7, p3, p8, p2, p5, p9	p4, p6, p10, p1, p7, p3, p8, p2, p5, p9
1	0	p6	p4	p4, p10, p1, p7, p3, p8, p2, p5, p9	p6, p4, p10, p1, p7, p3, p8, p2, p5, p9

Keterangan :

i = menunjukan *index* elemen saat ini

r = menunjukan *index* dari elemen yang akan ditukar, nilai r diambil secara acak dari *range* yang diperbolehkan (0-9)

Array[i] = menunjukan nilai yang terdapat pada *array* ke – i

Array[r] = menunjukan nilai yang terdapat pada *array* ke - r

Array yang sudah *fix* = menunjukan *array* yang sudah tidak boleh ditukar kembali nilainya, jumlahnya bertambah seiring bertambahnya iterasi

Array yang sudah ditukar = menunjukan isi *array* setelah elemennya ditukar pada setiap iterasi.

Tahapan dan langkah algoritma fisher-yates shuffle yang dimasukan kedalam *pseudecode* dapat dilihat pada tabel 3.2

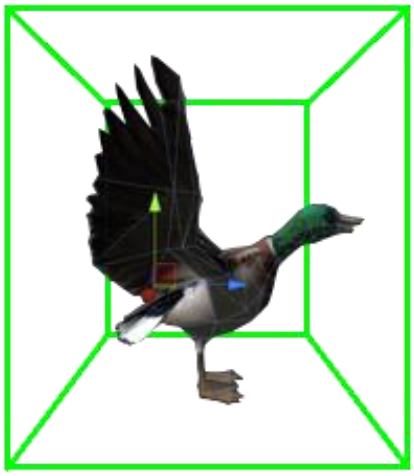
Tabel 3. 2 Tahapan Algoritma Fisher-Yates Shuffle

No	**Algoritma**	***Pseudocode***
1	Selama masih ada elemen tersisa untuk diacak	`for i in <array.length - 1, 1> do`
2	Ambil elemen secara acak dari elemen tersisa	`r = random number from interval <0, i)`
3	Kemudian tukar dengan elemen saat ini	`swap(array[i], array[r])`

3.1.4 Collision Detection

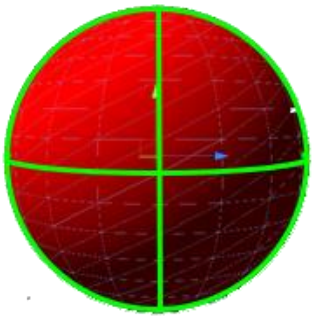
Collision detection atau deteksi tumbukan merupakan suatu hal mendasar dalam bidang *game programming* dan hampir sebagian besar *game* membutuhkan hal ini sebagai inti dari *game mechanic*. *Collision detection* adalah fungsi yang digunakan untuk dapat mengetahui obyek-obyek apa saya yang bersentuhan satu sama lain dalam bidang koordinat 2 dimensi ataupun 3 dimensi. Selain itu, *collision detection* juga berguna untuk menentukan posisi dari satu obyek dengan obyek yang lain sehingga tidak ada obyek yang saling menembus.

Fungsi collision detection sudah tersedia di dalam *software* unity 3d. Teknik yang digunakan unity adalah *bounding box* yang membentuk batas kotak yang menyelimuti obyek 3D.



Gambar 3. 1 Bounding Box Bebek

Gambar 3.1 merupakan *bounding box* dengan garis berwarna hijau yang diterapkan pada karakter bebek dalam game ini. *Bounding box* yang diterapkan pada karakter bebek merupakan sebuah kubus. Alasannya karena jika *bounding box* dibuat menyerupai dengan bentuk model 3d karakter maka akan menambah beban kerja *processor*. Selain itu karakter bebek yang akan dimunculkan dalam *game* bisa mencapai puluhan dalam sekali *render*. Jelas hal ini harus dihindari karena *game* duck hunt ini dikembangkan untuk platform android yang mempunyai *processor* lebih kecil dibandingkan sebuah komputer *desktop*.



Gambar 3. 2 Bounding Box Peluru

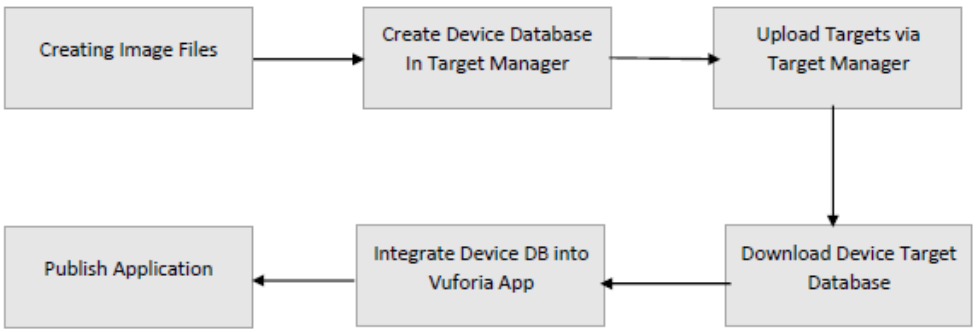
Gambar 3.2 menggambarkan *bounding box* yang diterapkan pada model peluru. Peluru yang digunakan dalam *game* duck hunt berbentuk *sphere* yang memiliki *bounding box* sesuai dengan bentuknya yaitu *sphere*.

Unity 3D menyediakan fungsi yang akan dipanggil ketika 2 buah obyek atau lebih saling bertabrakan yaitu dengan menggunakan fungsi `OnCollisionEnter()`. *Pseudocode collision detection* pada *game* duck hunt adalah sebagai berikut:

```
Function OnCollisionEnter()
If (collision = "firebulletgun") then
     If (isHit = false) then
          Call plusScore()
          isHit <- true
     end if
     play animation (duckObject, "falling")
     destroy (duckObject)
end if
End
```

3.1.5 Analisis Augmented Reality

Augmented reality pada *game duck hunt* ini digunakan sebagai media interaksi antara pemain dengan *game*. *Library augmented reality* yang digunakan dalam *game* ini adalah vuforia qualcomm. Vuforia merupakan *library* yang dapat digunakan untuk *mobile platform* seperti android dan ios. Alur kerja pembuatan aplikasi *augmented reality* menggunakan *library* vuforia dapat dilihat pada gambar 3.3



Gambar 3. 3 Alur Kerja Vuforia

3.1.5.1 Creating Image Files

Tahapan awal dalam mengembangkan aplikasi dengan teknologi *augmented reality* menggunakan *library* vuforia adalah pembuatan gambar yang akan dijadikan *marker* atau biasa disebut *image target*. *Image target* merupakan gambar yang dapat dideteksi dan dilacak oleh vuforia SDK. Tidak seperti *marker* tradisional, kode matriks data, dan kode QR, *image target* tidak perlu berwarna hitam dan putih atau memiliki kode untuk dapat dikenali.

Vuforia SDK menggunakan algoritma yang canggih untuk mendeteksi dan melacak *natural features* (fitur alami) yang ditemukan didalam gambar itu sendiri. Vuforia SDK mengenali *image target* dengan cara membandingkan fitur alami pada *image target* terhadap *database resource* yang telah didaftarkan (diupload) sebelumnya. Setelah *image target* terdeteksi, SDK akan melacak gambar menggunakan kamera *device*.

Beberapa hal yang harus diperhatikan dalam pembuatan *image target* adalah sebagai berikut:

a. Kaya akan detail
b. Kontras yang baik
c. Tidak ada pola berulang

d. Gambar harus 8 atau 24-bit berformat PNG atau JPG, besar ukuran minimal 320 x 320 *pixel* dan kurang dari 2 MB, dan format gambar RGB atau *grayscale*.

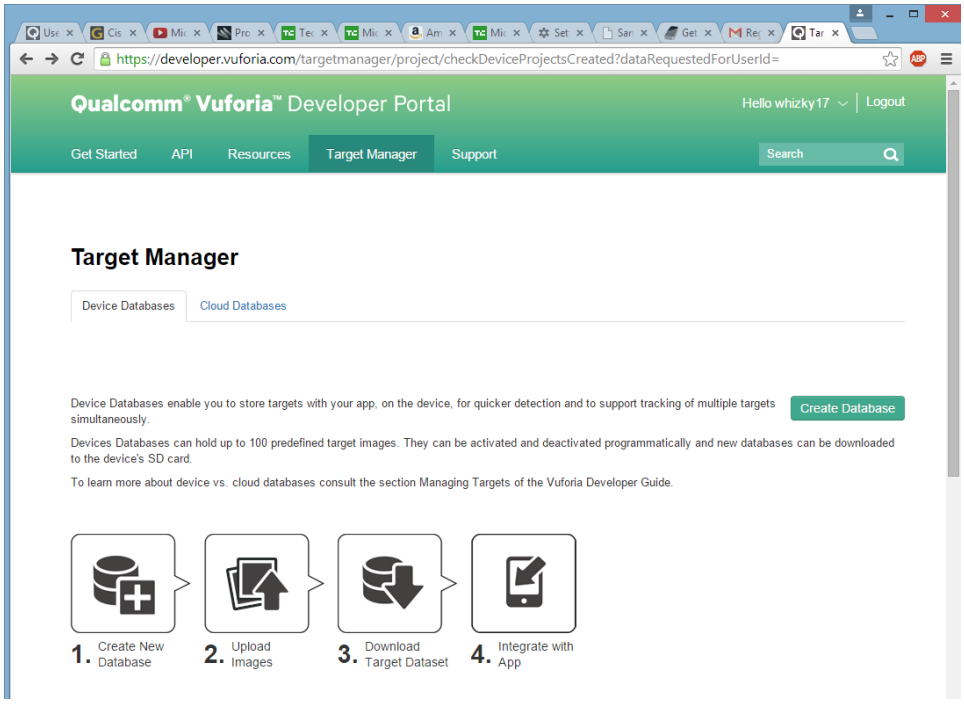
Gambar yang digunakan sebagai *image target* pada *game duck hunt* dengan teknologi *augmented reality* dapat dilihat pada gambar 3.4 berukuran 640 x 640 px.



Gambar 3. 4 Image Target Game Duck Hunt

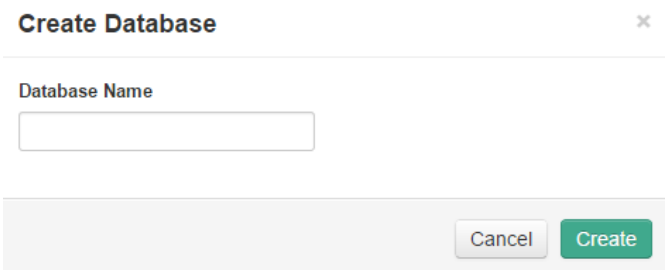
3.1.5.2 Create Device Database In Target Manager

Tahapan ini merupakan pembuatan *database* pada *target manager* yang telah disediakan oleh vuforia. *Target manager* merupakan sebuah *tool* untuk mengelola *image target* secara *online*. Sebelum menggunakan *target manager*, *developer* diharuskan *login* terlebih dahulu atau *register* untuk *developer* yang belum mempunyai *account*. Setelah berhasil *login* akan muncul tampilan awal dari *target manager*, tampilannya dapat dilihat pada gambar 3.5



Gambar 3. 5 Target Manager Vuforia

Langkah selanjutnya adalah membuat *device database* dengan cara mengklik tombol *create database*. Lalu akan muncul kotak dialog seperti pada gambar 3.6, selanjutnya isikan nama *device database* yang akan dibuat dalam kasus ini nama *database* yang digunakan *duck hunt* dan klik tombol *create* untuk membuatnya.

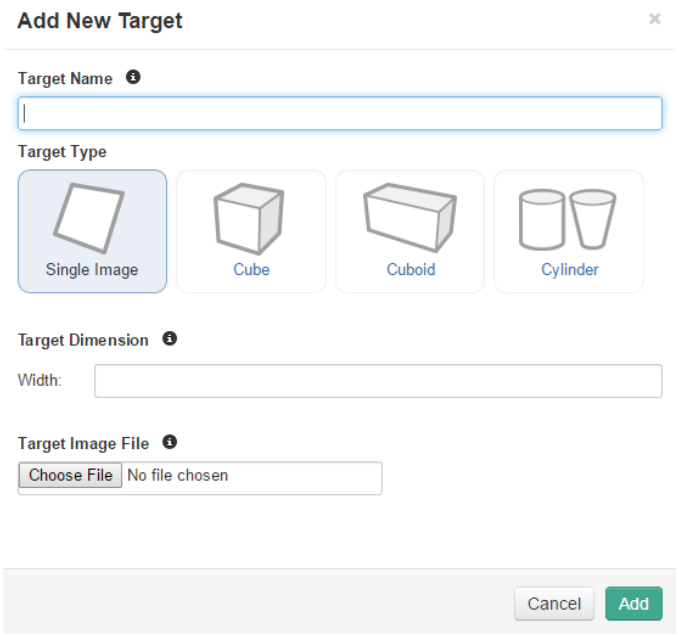


Gambar 3. 6 Pembuatan Device Database

3.1.5.3 Upload Targets via Target Manager

Tahapan selanjutnya adalah *upload* gambar yang akan dijadikan *image target* pada *database* yang telah dibuat sebelumnya di *target manager*. Untuk membuat *image target* pada *database* dilakukan dengan cara mengklik tombol *add*

target pada *device database*. Selanjutnya akan muncul kotak dialog *add new target* yang dapat dilihat pada gambar 3.7



Gambar 3. 7 Pembuatan Image Target

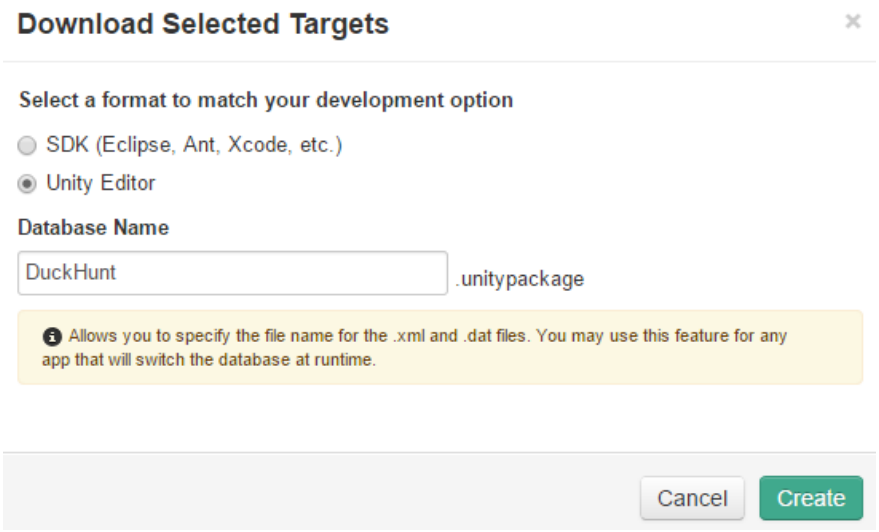
Isi seluruh data yang dibutuhkan, *target name* diisi dengan nama target1, *target type single image*, *target dimension* 640 px, lalu pilih *file* gambar yang akan dijadikan *image target*, dan klik tombol *add*. Gambar yang diupload akan diproses oleh *image analyzer* untuk menghasilkan *natural features* (fitur alami) yang terdapat pada gambar tersebut. *Natural features* yang dihasilkan oleh gambar yang telah diupload dapat dilihat pada gambar 3.8



Gambar 3. 8 Natural Features Dari Gambar

3.1.5.4 Download Device Target Database

Image target yang telah diproses oleh *image analyzer* pada *target manager* dapat di-*download* kembali untuk kemudian digunakan dalam aplikasi. Untuk mendownload *image target* dapat dilakukan dengan cara memilih *image target* yang akan digunakan lalu mengklik tombol *download selected targets*. Kemudian akan muncul kotak dialog seperti gambar 3.9



Gambar 3. 9 Download Image Target

Format yang dipilih dari *image target* yang akan didownload adalah unity *editor* atau .unitypackage karena aplikasi yang dikembangkan menggunakan *game engine* unity 3D.

3.1.5.5 Integrate Device DB into Vuforia App

Aplikasi *game duck hunt* dikembangkan dengan menggunakan *game engine* unity 3D. *Game engine* ini mendukung *library* vuforia dalam pembuatan aplikasi *augmented reality*. Beberapa *target platform* yang didukung oleh Vuforia SDK dapat dilihat pada gambar 3.10



	Development Options				
	Unity		Native SDK		
	Android	iOS	Android		iOS
			Java API	Java & C++	Objective C
Windows	✓		✓	✓	
OSX	✓	✓	✓	✓	✓
Linux			✓	✓	

Gambar 3. 10 Target Platform Vuforia SDK

Gambar 3.10 merupakan gambaran mengenai *target platform* yang didukung oleh *library* vuforia berdasarkan *operating system* yang digunakan dalam pengembangannya.

Vuforia SDK dapat didownload secara gratis pada websitenya. *Library* vuforia yang digunakan dalam pengembangan *game duck hunt* ini adalah vuforia AR *extension* yang dapat digunakan pada *game engine* unity 3D. File vuforia AR *extension* untuk unity berformat .unitypackage yang langsung dapat diimport oleh *game engine* unity 3D. Secara garis besar *library* vuforia memiliki beberapa proses dalam penerapan *augmented reality*, diantaranya adalah:

1. Proses Inisialisasi

Proses inisialisasi merupakan tahap awal yang dilakukan oleh sistem saat aplikasi mulai dijalankan. Beberapa komponen yang diinisialisasi pada tahap ini adalah kamera, *dataset*, *image target*, dan *tracker manager*.

Kamera merupakan salah satu komponen yang penting dalam proses *augmented reality* karena melalui kamera proses pelacakan *marker* dapat dilakukan. Fungsi yang digunakan dalam proses inisialisasi perangkat kamera adalah menggunakan fungsi `CameraDevice.init()`. Ketika melakukan inisialisasi kamera terdapat beberapa parameter yang dimiliki oleh kamera tersebut, diantaranya adalah:

a. *Camera Device Mode*

Camera device mode memiliki beberapa pilihan yaitu `MODE_DEFAULT` (merupakan *mode* standar yang digunakan oleh *library* vuforia), `MODE_OPTIMIZE_QUALITY` (merupakan *mode* yang lebih mementingkan kualitas dari kamera, namun akan menjadi terasa berat saat menggunakannya),

dan MODE_OPTIMIZE_SPEED (merupakan *mode* yang lebih mementingkan kecepatan dari pergerakan kamera, namun kualitas yang dihasilkan kurang maksimal). Masing-masing *mode* tersebut dapat dipilih sesuai dengan kebutuhan dari aplikasi yang dikembangkan.

b. *Max Simultaneous Image Target*

Merupakan batas maksimal dari jumlah *image target* yang diperbolehkan dideteksi oleh sistem pada waktu yang bersamaan.

c. *Camera Direction*

Camera direction merupakan parameter yang digunakan untuk memilih perangkat kamera mana yang akan digunakan. Terdapat 3 pilihan didalamnya, yaitu CAMERA_DEFAULT (kamera standar yang digunakan yaitu kamera belakang pada perangkat android), CAMERA_BACK (kamera belakang), dan CAMERA_FRONT (kamera depan).

d. *Camera Focus Mode*

Merupakan parameter yang akan berpengaruh pada fokus dari suatu kamera. *Camera focus mode* berisi beberapa pilihan, diantaranya adalah FOCUS_MODE_NORMAL (merupakan standar dari *mode* fokus kamera), FOCUS_MODE_TRIGGERAUTO (merupakan *trigger* dari sebuah operasi *autofocus*), FOCUS_MODE_CONTINUOUSAUTO (merupakan *mode autofocus* yang berkelanjutan), FOCUS_MODE_INFINITY (merupakan *mode* fokus yang tidak terbatas), dan FOCUS_MODE_MACRO (merupakan *mode* makro untuk fokus pada obyek yang dekat).

Setelah proses inisialisasi kamera berhasil dilakukan, kemudian proses selanjutnya adalah menjalankan kamera dengan fungsi CameraDevice.start().

Dataset merupakan *database* dari vuforia SDK yang menyimpan seluruh *image target* dalam suatu aplikasi. Inisialisasi dataset dilakukan dengan menggunakan fungsi DataSet.load("DuckHunt.xml", DataSet.STORAGE_TYPE.STORAGE_APPRESOURCE). Dimana *file* DuckHunt.xml merupakan *dataset* yang digunakan dalam mengembangkan

game duck hunt. *Image target* yang terdapat di dalam satu *dataset* maksimal berjumlah 100 buah. Untuk menghitung berapa banyak *image target* yang digunakan dalam suatu *dataset* dapat menggunakan fungsi `DataSet.getNumTrackables()`.

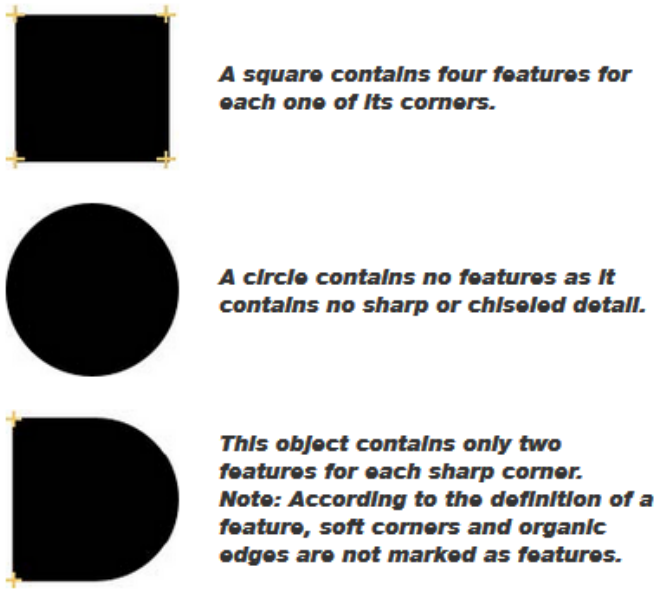
Tracker manager merupakan komponen dari vuforia yang berfungsi untuk melakukan pelacakan pada *image target*. Inisialisasi pada *tracker manager* dilakukan dengan menggunakan fungsi `TrackerManager.initTracker()`. Setelah proses inisialisasi *tracker manager* selesai, proses selanjutnya adalah *tracker* dapat dijalankan dengan menggunakan fungsi `TrackerManager.start()`.

2. Proses *Tracking*

Proses pelacakan yang dilakukan oleh *library* vuforia menggunakan metode *markerless* yang berarti tidak menggunakan *marker* berwarna hitam putih yang berbentuk kotak dan berpola, melainkan menggunakan suatu gambar yang memiliki *detail* dan *contrass* warna yang kuat sehingga banyak terdapat *natural features* (fitur alami) didalamnya.

Kelebihan menggunakan metode *markerless* dibanding metode lainnya adalah metode ini menggunakan *natural features tracking* dari suatu gambar atau obyek yang akan dilacak, seperti garis, tepi dan sudut sehingga menghasilkan proses *tracking* yang lebih kuat dan cocok untuk digunakan pada *mobile augmented reality*.

Natural features dari suatu gambar didapatkan dengan cara deteksi sudut atau *corner detection* yang merupakan pendekatan yang digunakan untuk mengekstraksi beberapa jenis fitur dan menyimpukan isinya.



Gambar 3. 11 Fitur Alami Dalam Gambar

Gambar 3.11 menggambarkan bahwa fitur akan muncul pada permukaan sudut yang tajam. Semakin banyak sudut, maka fitur yang dihasilkan pun akan semakin banyak.

Tracking dapat bekerja dengan baik jika image target dicetak pada kertas yang non-reflektif (tidak memantulkan cahaya) dan berada pada permukaan yang datar. Selain itu kondisi pencahayaannya juga harus menyebar/merata untuk mendapatkan hasil *tracking* yang baik. Terdapat beberapa komponen dalam proses pelacakan yang dilakukan dengan menggunakan *library* vuforia. Adapun komponen-komponennya adalah sebagai berikut:

A. Parameter

Parameter yang digunakan oleh vuforia SDK dalam proses pelacakan adalah sebagai berikut:

a. Tipe

UNKNOWN_TYPE : pelacakan yang tidak diketahui.

IMAGE_TARGET : pelacakan berdasarkan gambar.

MULTI_TARGET : pelacakan gabungan.

MARKER : pelacakan *marker*.

b. Nama

Sebuah kalimat yang unik yang digunakan untuk mengidentifikasi pelacakan dari *database*. Untuk penulisan nama hanya diperbolehkan maksimal 64 karakter dan hanya mengandung karakter a-z, A-Z, 0-9, dan [-_.]

c. Status

UNKNOWN : tempat/lokasi pelacakan tidak diketahui. Biasanya dikembalikan sebelum *tracker* melakukan *initialization*.

UNDEFINE : tempat/lokasi pelacakan tidak didefinisikan.

NOT_FOUND : lokasi pelacakan tidak ditemukan pada *database* yang dituju.

DETECTED : lokasi pelacakan dideteksi dalam *frame*.

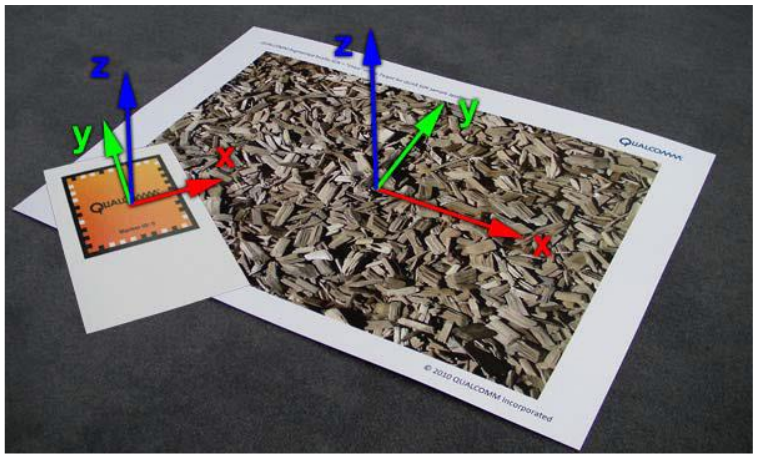
TRACKED : pelacakan telah terlacak dalam *frame*.

d. Posisi

Matriks 3x4 digunakan untuk menentukan posisi *marker* yang sudah diidentifikasi.

B. Koordinat Sistem

Output yang dihasilkan berupa identifikasi posisi *marker* dengan menggunakan tiga sumbu koordinat yaitu x,y, dan z. koordinat ini dimaksudkan agar posisi objek dapat dengan mudah diatur berdasarkan sumbu koordinat.



Gambar 3. 12 Koordinat Sistem

Gambar 3.12 menggambarkan bahwa Vuforia SDK menggunakan sistem koordinat tangan kanan. Setiap *Image Target* dan *Frame Marker* mendefinisikan sistem koordinat lokal dengan (0,0,0) di pusat (tengah) dari target. + X pergi ke kanan, + Y naik dan + poin Z keluar dari dilacak (ke arah dari yang dapat dilihat). Asal sistem koordinat lokal *Multi Target* didefinisikan oleh Bagian Target gambar yang berubah relatif terhadap komponen ini. Laporan pose *Multi Target* adalah posisi dari asal, independen dimana bagian individu dilacak dalam *Multi Target*. Fitur ini memungkinkan sebuah objek geometris (kotak) yang harus dilacak terus menerus dengan koordinat yang sama, bahkan jika bagian *Image Target* lain yang terlihat dalam tampilan kamera komponen-komponennya.

`Tracker.start()` merupakan fungsi yang digunakan untuk memulai proses pelacakan pada *image target* melalui kamera. `TrackableResult.STATUS` merupakan suatu *public variable* yang bertipe *integer* yang dapat diakses oleh seluruh *class* dalam *library* vuforia. Variabel ini memiliki beberapa parameter status yang berhubungan dengan proses pelacakan, diantaranya adalah:

a. `TrackableResult.STATUS.UNKNOWN`

   Merupakan keadaan dimana *image target* yang terdeteksi tidak diketahui.

b. `TrackableResult.STATUS.UNDEFINED`

   Merupakan keadaan dimana *image target* tidak ditemukan.

c. `TrackableResult.STATUS.DETECTED`

   Merupakan keadaan dimana *image target* sedang dideteksi oleh *tracker*.

d. `TrackableResult.STATUS.TRACKED`

   Merupakan keadaan dimana *image target* sudah berhasil dideteksi dan statusnya menjadi terlacak.

e. `TrackableResult.STATUS.EXTENDED_TRACKED`

   Merupakan keadaan dimana *image target* yang sudah terlacak diperpanjang jangkauannya.

Fungsi yang dapat digunakan untuk mengetahui status dari *trackable result* adalah `TrackableResult.getStatus()`.

3. Proses *Rendering*

Proses *rendering* merupakan proses untuk menampilkan obyek-obyek 3D yang terdapat didalam aplikasi game. Obyek 3D akan direnderer apabila *image target* dapat berhasil dilacak atau status dari variabel *trackable result* adalah `TrackableResult.STATUS.TRACKED` atau `TrackableResult.STATUS.EXTENDED_TRACKED`.

`Renderer.setARProjection(float nearPlane, float farPlane)` merupakan fungsi yang digunakan untuk menghitung sebuah perspektif dari *projection matrix* untuk proses *rendering augmented reality* dan memasukannya ke OpenGL.

3.1.5.6 Publish Application

Tahapan ini merupakan tahapan akhir dari pembuatan aplikasi, setelah aplikasi siap untuk digunakan selanjutnya aplikasi dapat dipublish menjadi *file* dengan tipe .apk yang dapat diinstall pada perangkat dengan sistem operasi android.

3.2 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak merupakan suatu proses untuk mendapatkan informasi, model, dan spesifikasi tentang perangkat lunak yang akan dibuat sesuai dengan kebutuhan. Analisis yang dilakukan berupa analisis terhadap kebutuhan dalam implementasi teknologi *augmented reality* pada *game duck hunt* berbasis android. Analisis kebutuhan dibagi menjadi 2, yaitu analisis kebutuhan non fungsional dan kebutuhan fungsional.

3.2.1 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional merupakan analisis yang dilakukan untuk menentukan spesifikasi kebutuhan dalam implementasi teknologi *augmented reality*. Spesifikasinya meliputi analisis kebutuhan perangkat keras (*hardware*),

perangkat lunak(*software*), dan pengguna(*user*) yang dibutuhkan dalam proses pengembangan *game duck hunt*.

3.2.1.1 Analisis Perangkat Keras

Spesifikasi minimum perangkat keras android yang dibutuhkan oleh pengguna untuk menjalankan *game duck hunt* dengan teknologi *augmented reality* dijelaskan pada tabel 3.3

Tabel 3. 3 Perangkat Keras Pengguna

No.	Perangkat Keras	Spesifikasi
1	Processor	1 GHz
2	Layar	Touchscreen
3	Kamera	3 MP Resolution
4	Baterai	Standard
5	Speaker	Standard

3.2.1.2 Analisis Perangkat Lunak

Perangkat lunak minimun yang dibutuhkan untuk menjalankan *game duck* hunt dengan teknologi *augmented reality* dijelaskan pada tabel 3.4

Tabel 3. 4 Perangkat Lunak Pengguna

No.	Perangkat Lunak	Keterangan
1	Sistem Operasi	Android 4.0

3.2.1.3 Analisis Pengguna

Analisis pengguna dimaksudkan untuk mengetahui siapa saja pengguna yang terlibat dalam sistem. Terdapat 2 hal yang akan memudahkan pengguna ketika memainkan suatu *game* yaitu pengetahuan dan pengalaman.

1. *User Knowledge* and *Experience*

Klasifikasi *knowledge and experience* dari pengguna aplikasi *game duck hunt* dapat dilihat pada tabel 3.5

Tabel 3. 5 Klasifikasi Knowledge and Experience

Klasifikasi	**Keterangan**
Education Level	Bisa digunakan oleh berbagai kalangan, seperti pelajar, mahasiswa hingga masyarakat awam.
Reading Level	Bisa digunakan oleh berbagai *level* pendidikan dengan *reading level* yang sedang.
Typing Skills	Tidak memerlukan *typing skills* yang tinggi.
Computer Literacy	*Moderate* (menengah)
Task Experience	Bisa digunakan oleh pengguna dengan pengalaman penggunaan *smartphone* dan *game* yang sedang.
System Experience	Bisa digunakan oleh pengguna dengan pengalaman penggunaan *smartphone* dan *game* yang sedang.
Application Experience	Bisa digunakan pada sistem operasi android 4.0 atau lebih tinggi
Native Language	Menggunakan satu bahasa yaitu bahasa indonesia
Use of Other System	Bisa dijalankan tanpa perlu *install* aplikasi lain.

2. *User Physical Characteristic*

Terdapat beberapa hal yang harus diperhatikan terhadap karakteristik fisik seseorang untuk menggunakan aplikasi *game duck hunt* ini yaitu *color blind*, *handedness*, dan *gender*. Klasifikasi *user physical characteristic* dari pengguna aplikasi *game duck hunt* dapat dilihat pada tabel 3.6

Tabel 3. 6 Klasifikasi Physical Characteristic

Klasifikasi	**Keterangan**
Age	15 – 25 Tahun
Gender	Pria dan Wanita
Handedness	Kanan dan Kiri
Color Blind	Pengguna yang tidak bisa membedakan warna yang satu dengan yang lainnya (buta warna) masih mampu memainkan *game*, tetapi tidak akan optimal.

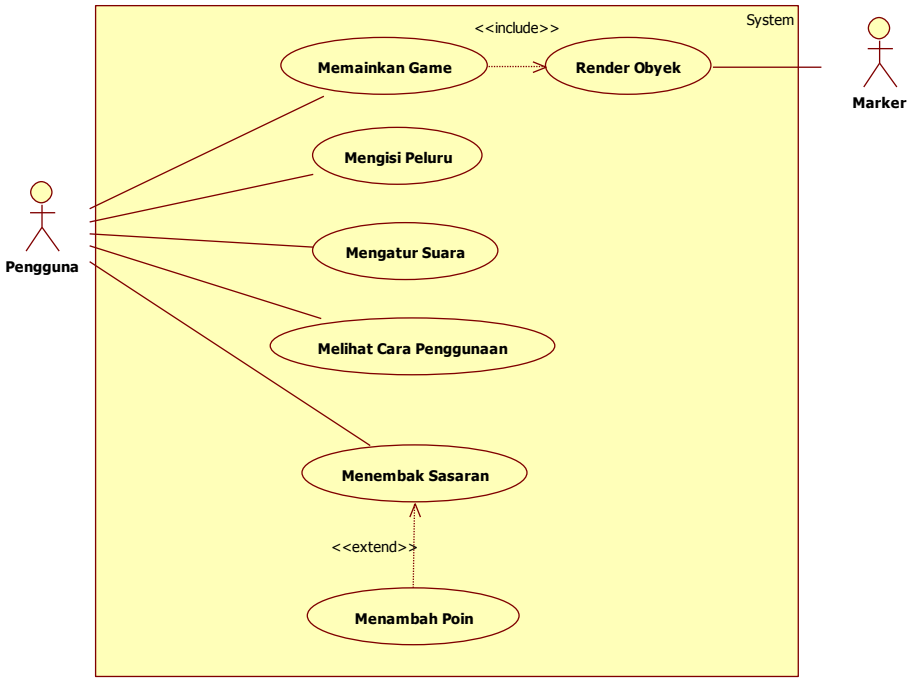
3.2.2 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem dan menjelaskan kebutuhan yang diperlukan sistem agar berjalan dengan baik serta sesuai dengan kebutuhan.

Pemodelan sistem dilakukan dengan metode berorientasi obyek dengan menggunakan standarisasi *Unified Modeling Language* (UML). Tahap-tahap pemodelan dalam menggunakan UML antara lain *use case diagram*, *activity diagram*, *sequence diagram* dan *class diagram*.

3.2.2.1 Use Case Diagram

Use case diagram digunakan untuk menggambarkan sejumlah *external actor* dengan *use case* yang terdapat dalam sistem. *Use case* diagram ini hanya menggambarkan keadaan lingkungan sistem yang dapat dilihat dari luar oleh actor. *Use case* terdiri dari tiga bagian yaitu definisi *actor*, definisi *use case*, dan scenario *use case*. *Use case* yang terdapat dalam *game duck hunt* dapat dilihat pada gambar 3.13



Gambar 3. 13 Use Case Diagram Game Duck Hunt

1. Definisi *Actor*

Definisi *actor* berfungsi untuk mendeskripsikan peran *actor* yang terlibat dalam *use case* ini. Untuk lebih jelasnya dapat dilihat pada tabel 3.7

Tabel 3. 7 Definisi Aktor

Nama Aktor	Deskripsi
Pengguna	Orang yang berinteraksi dengan sistem/ orang yang mengakses semua fungsi dalam sistem.
Marker	*Marker* adalah alat bantu berupa gambar yang digunakan sebagai penanda dimana objek virtual 3D akan ditampilkan.

2. Definisi *Use Case*

Definisi *use case* berfungsi sebagai penjelasan mengenai proses yang terdapat pada setiap *use case*. Untuk lebih jelasnya dapat dilihat pada tabel 3.8

Tabel 3. 8 Definisi Use Case

Nama *Use Case*	Deskripsi
Memainkan Game	Pengguna memilih menu mulai game dan memainkan game.
Render Obyek	Pengguna mengarahkan kamera pada marker kemudian sistem melakukan *render* obyek-obyek yang terdapat di dalam *game* setelah proses pelacakan *marker* berhasil.
Menembak Sasaran	Pengguna menekan tombol tembak pada *game* saat sasaran telah ditemukan.
Menambah Poin	Sistem akan menambahkan poin jika pemain dapat menembak mengenai sasaran.
Mengisi Peluru	Sistem menambahkan peluru dalam *game* ketika pemain menekan tombol isi peluru.
Mengatur Suara	Pengguna dapat mengaktifkan atau menonaktifkan suara dalam *game*.
Melihat Cara Penggunaan	Sistem menampilkan cara penggunaan *game* setelah pengguna memilih menu cara penggunaan.

3. *Scenario* Use *Case*

Use Case Scenario mendeskripsikan urutan langkah-langkah dalam proses bisnis baik yang dilakukan aktor terhadap sistem maupun yang dilakukan oleh sistem terhadap aktor. Berdasarkan *use case* diagram pada gambar 3.13 maka *use case scenario* untuk aplikasi yang akan dibangun akan dijelaskan pada tabel 3.9 sampai dengan tabel 3.14

Tabel 3. 9 Skenario Use Case Memainkan Game

Identifikasi	
Nama *Use Case*	Memainkan Game
Aktor	Pengguna, Marker
Kondisi Awal	Pengguna berada di menu utama
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Pengguna menekan tombol mulai pada *game*.	
	2. Sistem memproses dan menampilkan tampilan dalam game.
	3. Sistem mengaktifkan kamera perangkat android.
	4. Sistem menampilkan pesan marker tidak terdeteksi.
5. Pengguna mengarahkan kamera perangkat android pada marker.	
	6. Sistem mendeteksi dan mencocokan marker yang tertangkap oleh kamera dengan *dataset* dalam sistem.
	7. Marker cocok dengan *dataset* dalam sistem kemudian sistem merender seluruh obyek dalam *game*.
	8. Sistem menjalankan timer pada game.
9. Pengguna memainkan game.	
	10. Sistem memunculkan dan menggerakan karakter npc.
11. Pengguna menembak karakter npc	
	12. Sistem mengecek collision karakter npc terhadap peluru.
	13. Peluru mengenai sasaran lalu sistem menambahkan poin.

Tabel 3.9 Skenario Use Case Memainkan Game (Lanjutan)

Aksi Aktor	Reaksi Sistem
	14. Peluru habis, sistem menampilkan tombol reload
15. Pengguna menekan tombol reload peluru	
	16. Sistem menambahkan peluru menjadi 5 butir
	17. Timer game habis kemudian sistem menampilkan pesan game over.
	18. Sistem membandingkan poin yang didapat dengan poin tertinggi yang tersimpan dalam game.
Kondisi Akhir	Permainan telah berakhir

Tabel 3. 10 Skenario Use Case Render Obyek

Identifikasi	
Nama *Use Case*	*Render* Obyek
Aktor	Pengguna, Marker
Kondisi Awal	Obyek dalam *game* belum ditampilkan
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Pengguna menekan tombol mulai pada *game*.	
	2. Sistem memproses dan mengaktifkan kamera pada perangkat android.
3. Pengguna mengarahkan kamera perangkat android pada marker.	
	4. Sistem mendeteksi dan mencocokan marker yang tertangkap oleh kamera dengan *dataset* dalam sistem.
	5. Marker cocok dengan *dataset* dalam sistem kemudian sistem merender seluruh obyek dalam *game*.
Skenario Alternatif	
	1. Marker tidak cocok dengan *dataset* dalam sistem, Sistem tidak dapat merender obyek dalam *game*.
Kondisi Akhir	Sistem merender seluruh obyek dalam *game*.

Tabel 3. 11 Skenario Use Case Menembak Sasaran

Identifikasi	
Nama *Use Case*	Menembak Sasaran
Aktor	Pengguna
Kondisi Awal	Seluruh obyek ditampilkan dalam *game*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Pengguna menekan tombol tembak pada *game*.	
	2. Sistem memanggil fungsi dalam *game* untuk mengeluarkan tembakan.
	3. Peluru ditembakan dan tepat mengenai sasaran
	4. Poin bertambah sebanyak 100 poin
Skenario Alternatif	
	1. Tembakan tidak mengenai sasaran dan poin tidak bertambah
Kondisi Akhir	Poin bertambah sebanyak 100 poin

Tabel 3. 12 Skenario Use Case Mengisi Peluru

Identifikasi	
Nama *Use Case*	Mengisi Peluru
Aktor	Pengguna
Kondisi Awal	Peluru habis
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
	1. Peluru habis dan sistem menampilkan tombol isi peluru
2. Pengguna menekan tombol isi peluru	
	3. Sistem menambahkan peluru sebanyak 5 butir.
Kondisi Akhir	Peluru terisi penuh yaitu 5 butir.

Tabel 3. 13 Skenario Use Case Mengatur Suara

Identifikasi	
Nama *Use Case*	Mengatur Suara
Aktor	Pengguna
Kondisi Awal	Pengguna berada di menu utama
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Pengguna memilih menu pengaturan	

Tabel 3.13 Skenario Use Case Mengatur Suara (Lanjutan)

Aksi Aktor	**Reaksi Sistem**
	2. Sistem menampilkan menu pengaturan.
3. Pengguna menonaktifkan suara kemudian menekan tombol simpan	
	4. Sistem menonaktifkan suara dalam *game*
Skenario Alternatif	
1. Pengguna mengaktifkan suara kemudian menekan tombol simpan	
	2. Sistem mengaktifkan suara dalam game
Kondisi Akhir	Suara dalam *game* tidak aktif

Tabel 3. 14 Skenario Use Case Melihat Cara Penggunaan

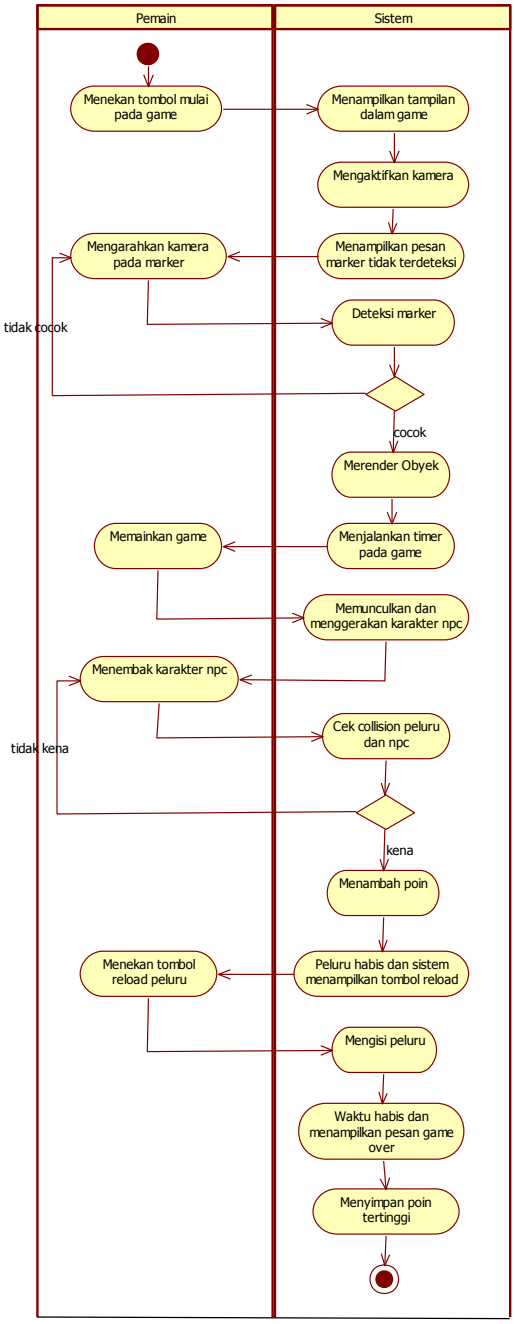
Identifikasi	
Nama *Use Case*	Melihat Cara Penggunaan
Aktor	Pengguna
Kondisi Awal	Pengguna berada di menu utama
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Pengguna memilih menu cara penggunaan	
	2. Sistem menampilkan menu cara penggunaan
Kondisi Akhir	Pengguna berada di menu cara penggunaan

3.2.2.2 Activity Diagram

Diagram activity menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi dan bagaimanan mereka berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem)

secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

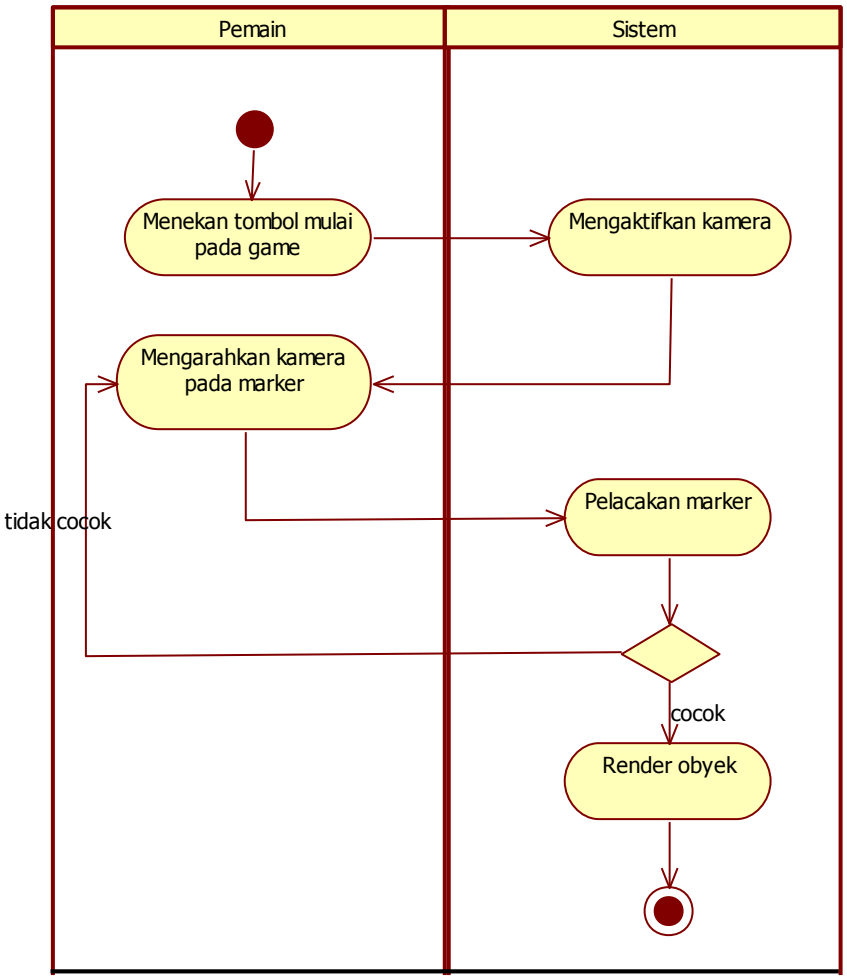
1. Activity Diagram Memainkan *Game*



Gambar 3. 14 Activity Diagram Memainkan Game

Gambar 3.14 menggambarkan aktivitas ketika pengguna mulai memainkan *game*, setelah pengguna memilih menu mulai *game*, sistem akan menampilkan tampilan dan mengaktifkan kamera pada perangkat android. Kemudian pengguna diharuskan mengarahkan kamera tersebut pada *marker* dan sistem akan mendeteksi *marker* tersebut, jika cocok maka pengguna dapat mulai memainkan *game* sampai dengan waktu dalam *game* tersebut habis.

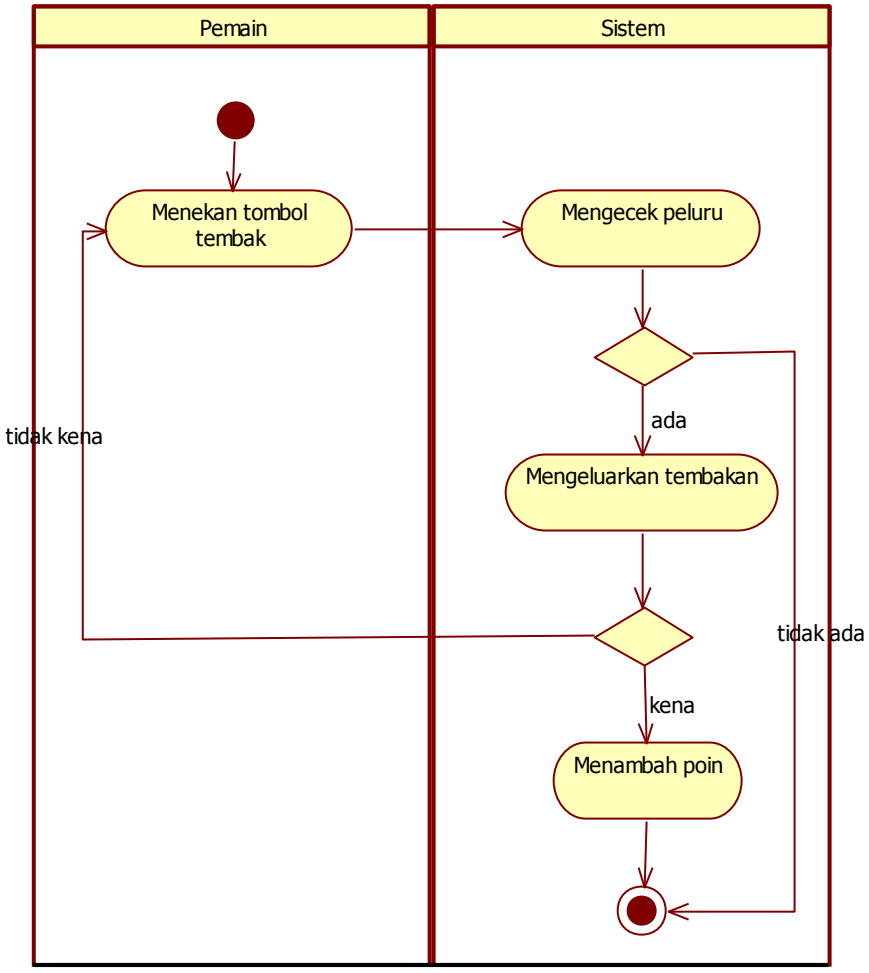
2. *Activity Diagram Render* Obyek



Gambar 3. 15 Activity Diagram Render Obyek

Gambar 3.15 menggambarkan aktivitas ketika sistem akan merender obyek, awalnya pengguna memilih menu mulai *game* lalu sistem akan mengaktifkan kamera pada perangkat android. Pengguna perlu mengarahkan kamera tersebut pada marker lalu sistem akan melakukan pelacakan pada *marker* tersebut, jika *marker* tersebut terdaftar didalam *dataset* sistem maka sistem akan merender seluruh obyek yang ada di dalam *game* dan pemain pun dapat mulai memainkan *game*.

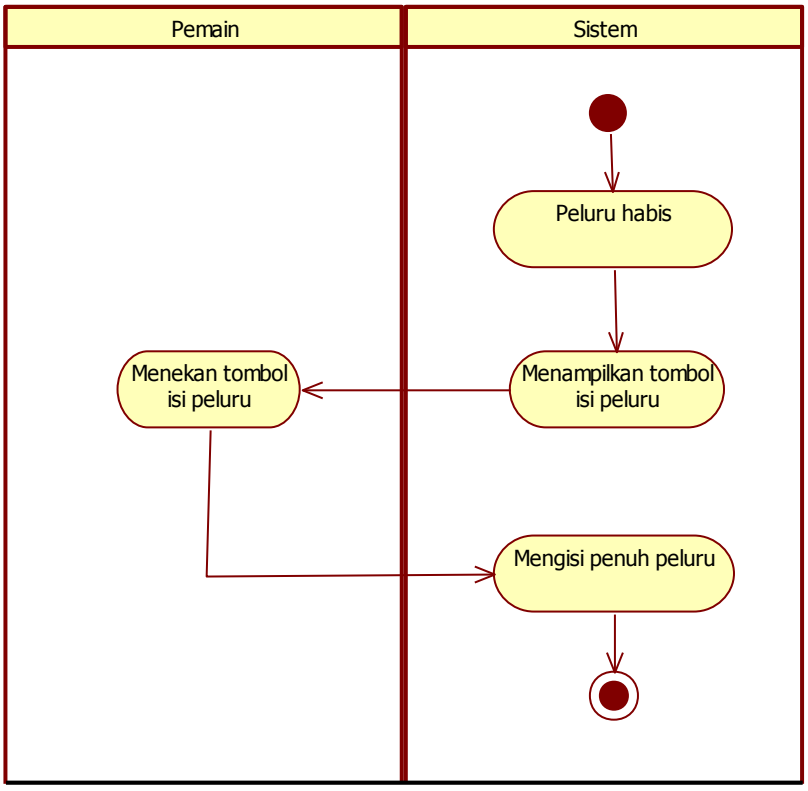
3. *Activity Diagram* Menembak Sasaran



Gambar 3. 16 Activity Diagram Menembak Sasaran

Gambar 3.16 menggambarkan aktivitas ketika pengguna menembak sasaran dalam game. Pengguna menekan tombol tembak dan sistem akan mengecek ketersediaan peluru, jika peluru tersedia maka sistem akan mengeluarkan tembakan dan jika tembakan tersebut mengenai sasaran maka sistem akan menambahkan poin.

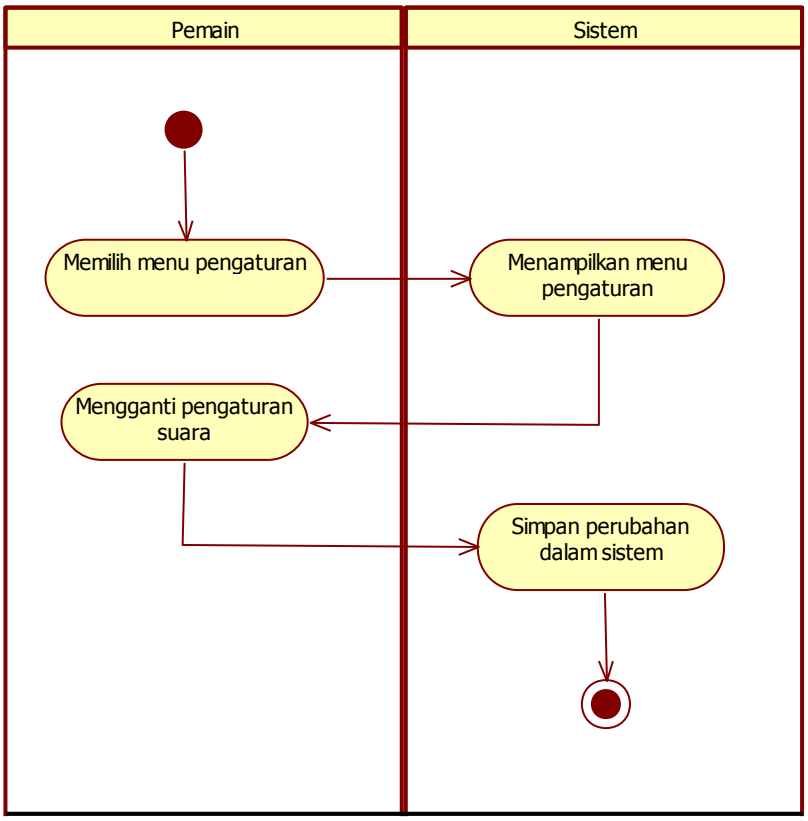
4. *Activity Diagram* Mengisi Peluru



Gambar 3. 17 Activity Diagram Mengisi Peluru

Gambar 3.17 menggambarkan aktivitas ketika pengguna diharuskan untuk mengisi peluru jika peluru habis. Sistem akan menampilkan tombol isi peluru lalu pengguna menekan tombol tersebut untuk melakukan pengisian peluru. Jumlah peluru pada *game* ini sebanyak 5 butir.

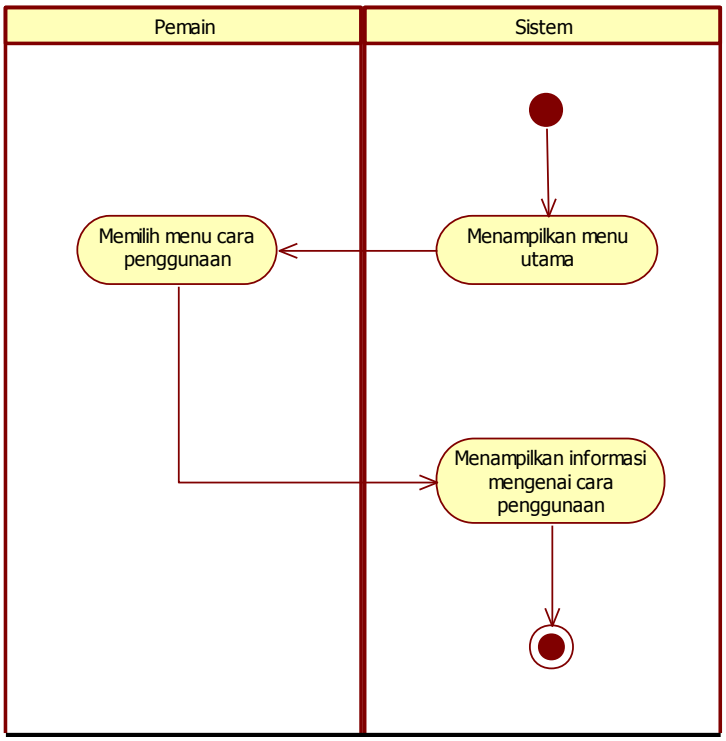
5. *Activity Diagram* Mengatur Suara



Gambar 3. 18 Activity Diagram Mengatur Suara

Gambar 3.18 menggambarkan aktivitas ketika pengguna akan mengatur suara yang ada didalam *game*. Pengguna memilih menu pengaturan dan sistem menampilkannya. Kemudian pada halaman tersebut terdapat tombol untuk mengubah pengaturan suara dengan cara menekan tombol on/off.

6. *Activity Diagram* Melihat Cara Penggunaan



Gambar 3. 19 Activity Diagram Cara Penggunaan

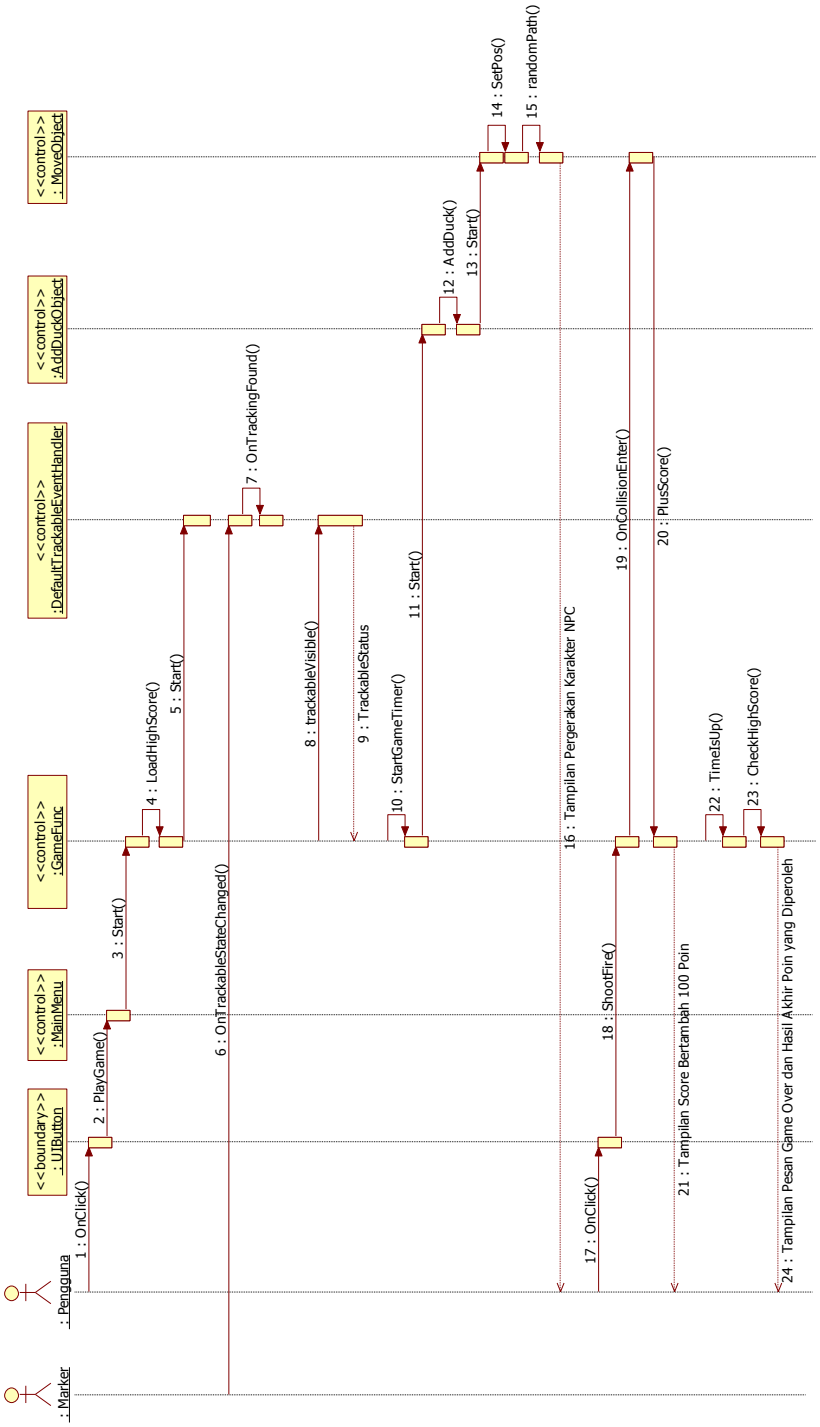
Gambar 3.19 menggambarkan aktivitas ketika pengguna ingin melihat cara menggunakan *game duck hunt* ini. Saat berada di menu utama pengguna dapat memilih menu cara penggunaan untuk melihat petunjuk bagaimana cara menggunakan *game* ini.

3.2.2.3 Sequence Diagram

Sequence diagram menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukkan sejumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini di dalam *use case*. Komponen utama *sequence diagram* terdiri atas obyek yang dituliskan dengan kotak segiempat. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan progres vertikal. Berikut ini adalah *sequence diagram* pada *game duck hunt*.

1. *Sequence* Diagram Memainkan *Game*

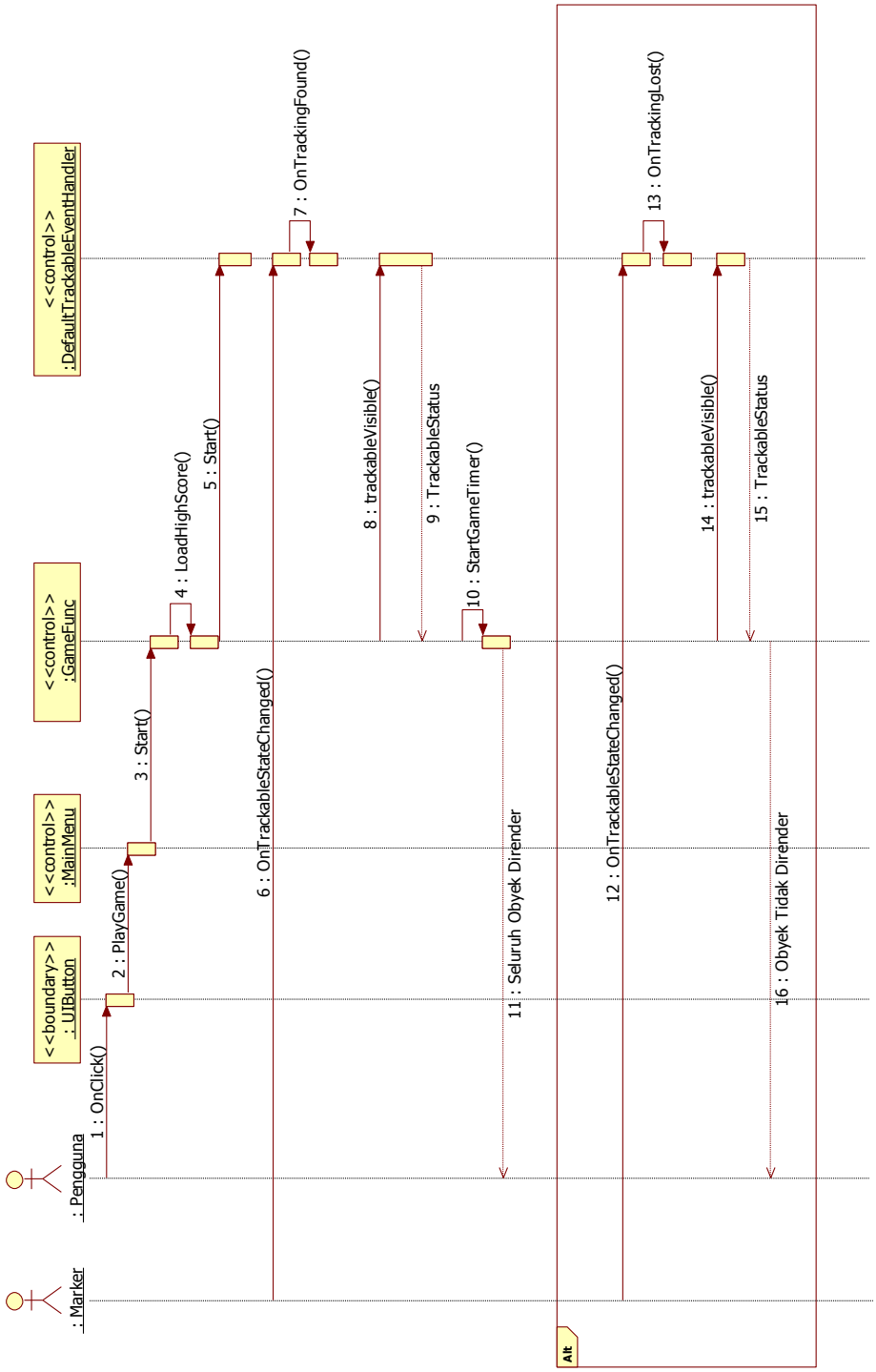
   Sequence diagram ini menunjukan rangkaian interaksi antar obyek pada proses memainkan *game*. Diagramnya dapat dilihat pada gambar 3.20



Gambar 3. 20 Sequence Diagram Memainkan Game

2. *Sequence Diagram* Render Obyek

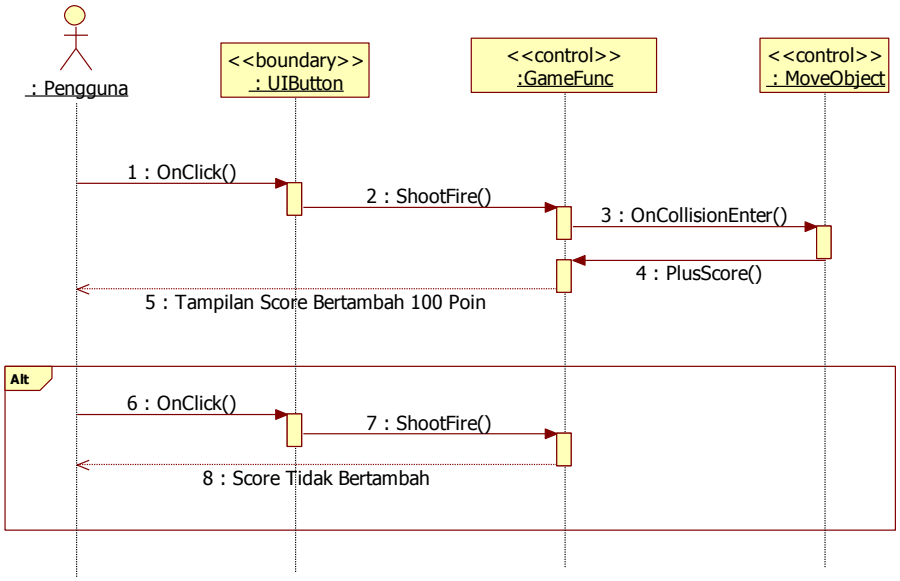
Sequence diagram ini menunjukan rangkaian interaksi antar obyek pada proses render obyek didalam *game*. Diagramnya dapat dilihat pada gambar 3.21



Gambar 3. 21 Sequence Diagram Render Obyek

3. *Sequence Diagram* Menembak Sasaran

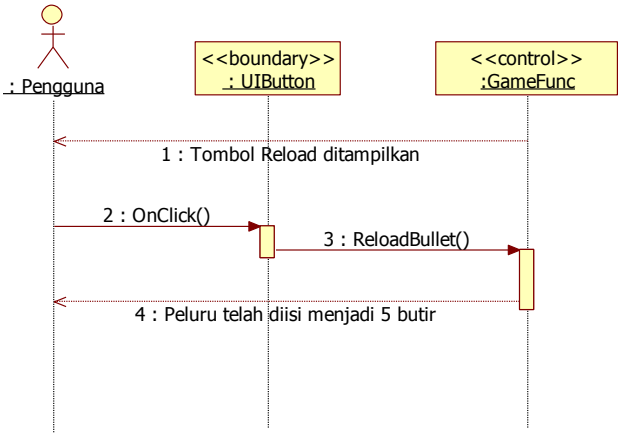
Sequence diagram memilih menu pengaturan menunjukan rangkaian interaksi antar obyek pada proses menembak sasaran. Diagramnya dapat dilihat pada gambar 3.22



Gambar 3. 22 Sequence Diagram Menembak Sasaran

4. *Sequence Diagram* Mengisi Peluru

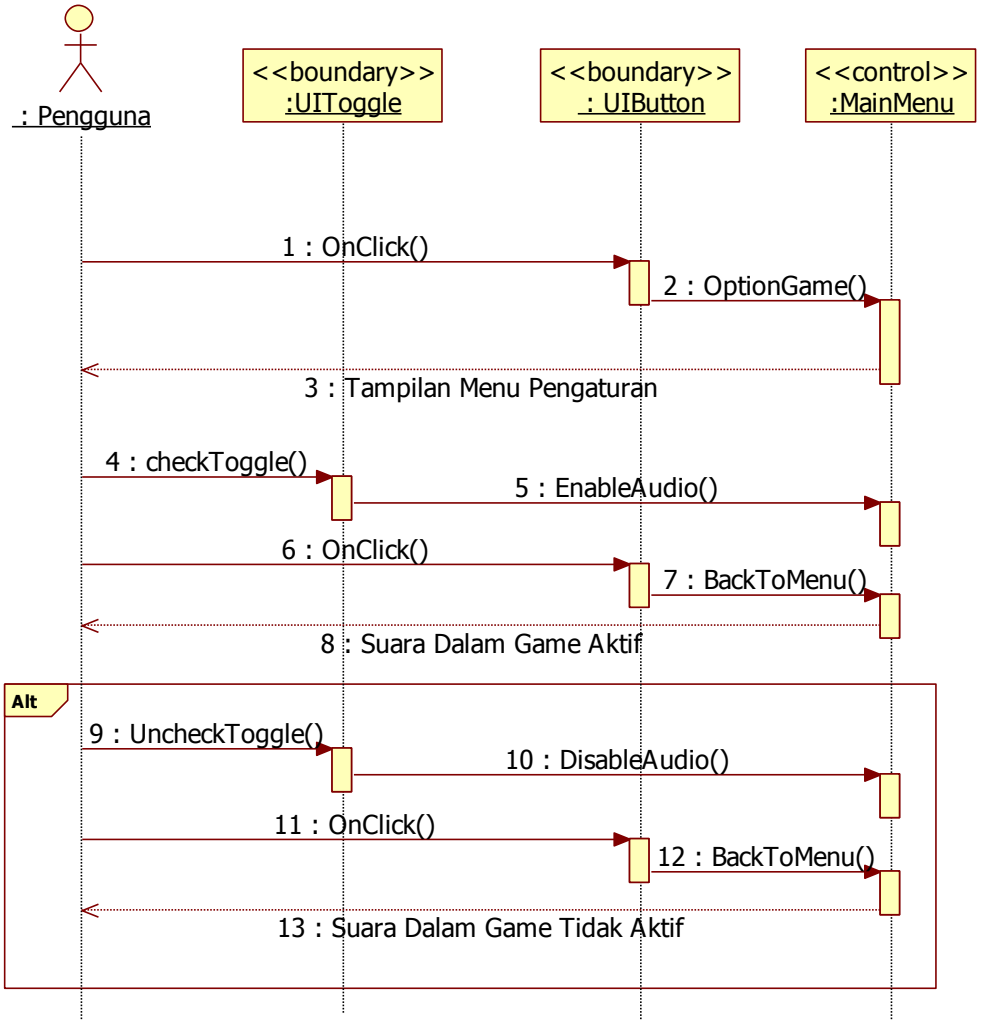
Sequence diagram ini menunjukan rangkaian interaksi antar obyek pada proses mengisi peluru. Diagramnya dapat dilihat pada gambar 3.23



Gambar 3. 23 Sequence Diagram Mengisi Peluru

5. *Sequence Diagram* Mengatur Suara

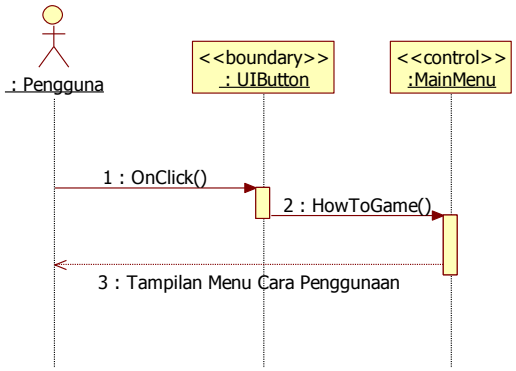
Sequence diagram ini menunjukan rangkaian interaksi antar obyek pada saat proses mengatur suara pada *game*. Diagramnya dapat dilihat pada gambar 3.24



Gambar 3. 24 Sequence Diagram Mengatur Suara

6. Sequence Diagram Lihat Cara Penggunaan

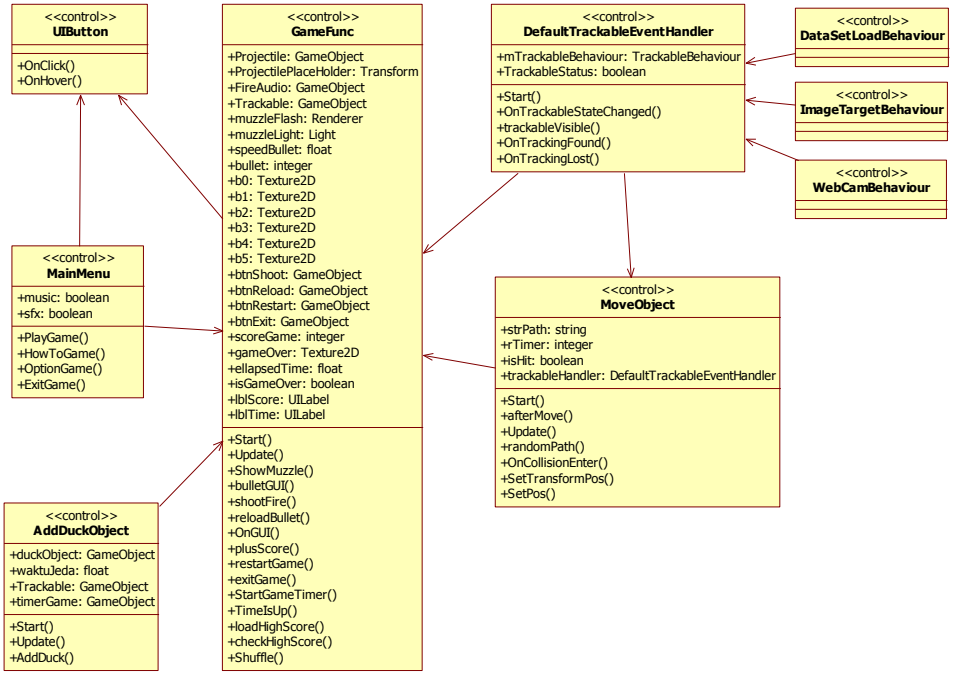
Sequence diagram ini menunjukan rangkaian interaksi antar obyek pada saat proses saat memilih menu cara penggunaan. Diagramnya dapat dilihat pada gambar 3.25



Gambar 3. 25 Sequence Diagram Cara Penggunaan

3.2.2.4 Class Diagram

Class diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (*attribut* atau *property*) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda atau fungsi). Class diagram yang terdapat pada Implementasi Teknologi *Augmented Reality* Pada *Game Duck Hunt* dapat dilihat pada gambar 3.26



Gambar 3. 26 Class Diagram Game Duck Hunt

3.3 Perancangan Sistem

Tahap perancangan merupakan kelanjutan dari proses analisis dimana dilakukan perubahan-perubahan terhadap sistem yang sedang berjalan. Hal ini dilakukan untuk mengatasi kekurangan yang ada, memudahkan pekerjaan yang dilakukan oleh orang yang terlibat dan menghemat waktu pekerjaan.

Perancangan *Game duck hunt* meliputi perancangan komponen permainan, perancangan *marker*, perancangan struktur menu, perancangan antarmuka, jaringan semantik, perancangan prosedural. Semua perancangan yang telah disebutkan diatas harus saling berhubungan agar menciptakan suatu sistem yang baik

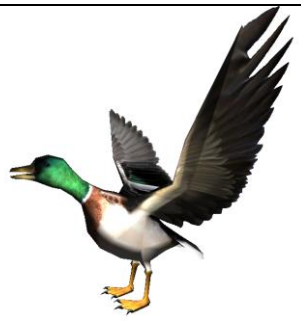
3.3.1 Perancangan Komponen Permainan

Perancangan komponen permainan merupakan bagian pendeskripsian setiap komponen yang ada pada *Game* baik karakter, bangunan, dan *story board*. Perancangan komponen permainan terdiri dari perancangan karakter dan perancangan *story board*.

3.3.1.1 Perancangan Karakter

Perancangan karakter adalah bagian dimana karakter pada *Game* dijelaskan ciri-cirinya baik itu secara fungsionalitas peran maupun secara desain. Pada *Game duck hunt*, karakter merupakan model 3D yang terdapat pada *Game*. Perancangan karakter pada *Game* ini dijelaskan pada tabel 3.15

Tabel 3. 15 Perancangan Karakter Game

Karakter	Keterangan
	Merupakan model 3D karakter NPC yang berbentuk bebek pada game duck hunt. Karakter ini akan menjadi target sasaran dalam game. Pemain harus menembak target ini untuk mendapatkan poin. Setiap target yang terkena tembakan pemain akan mendapatkan 100 poin.

3.3.1.2 Perancangan *Storyboard*

Storyboard merupakan sketsa gambar yang disusun secara berurutan berdasarkan naskah untuk menggambarkan lingkungan pada *Game*. Pada *Game duck hunt, storyboard* merupakan sebuah lingkungan/*environment* yang akan digunakan untuk bermain dan diterapkan pada sebuah *stage*. *Environment* pada *Game* ini digambarkan pada gambar 3.27

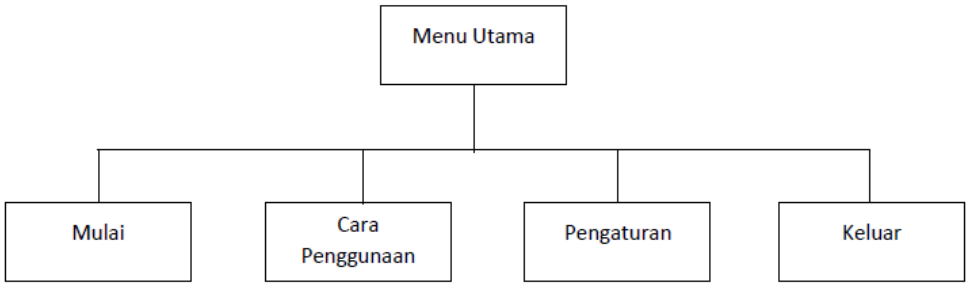


Gambar 3. 27 Storyboard Game Duck Hunt

Storyboard pada gambar 3.27 menceritakan seorang pemburu yang sedang berada dihutan ingin menangkap bebek dengan menggunakan sebuah senapan. Pemain dalam game ini akan menjadi seorang pemburu yang mempunyai misi untuk menangkap bebek sebanyak-banyaknya. Pemain harus mengamati setiap pohon yang ada dilingkungan sekitar untuk dapat memburu bebek, karena bebek-bebek akan mulai terbang dari satu pohon ke pohon lainnya.

3.3.2 Perancangan Struktur Menu

Dalam pembangunan aplikasi diperlukan sebuah struktur menu untuk menunjukan keterkaitan dan tingkatan antara menu utama dengan sub menunya, sehingga pengguna tidak mengalami kesulitan dalam memilih menu-menu yang diinginkan. Perancangan struktur menu *game duck hunt* datap dilihat pada gambar 3.28

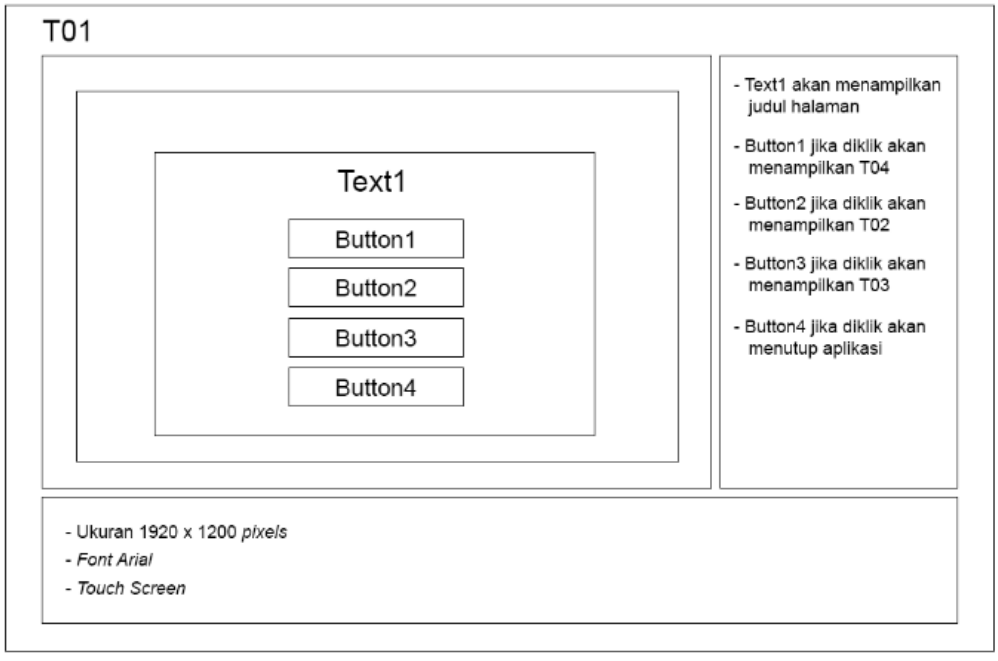


Gambar 3. 28 Struktur Menu Game

3.3.3 Perancangan Antarmuka

Perancangan antarmuka merupakan salah satu bagian penting dalam perancangan sistem karena nantinya antarmuka tersebut akan menjadi fasilitas yang menjembatani interaksi manusia dengan sistem.

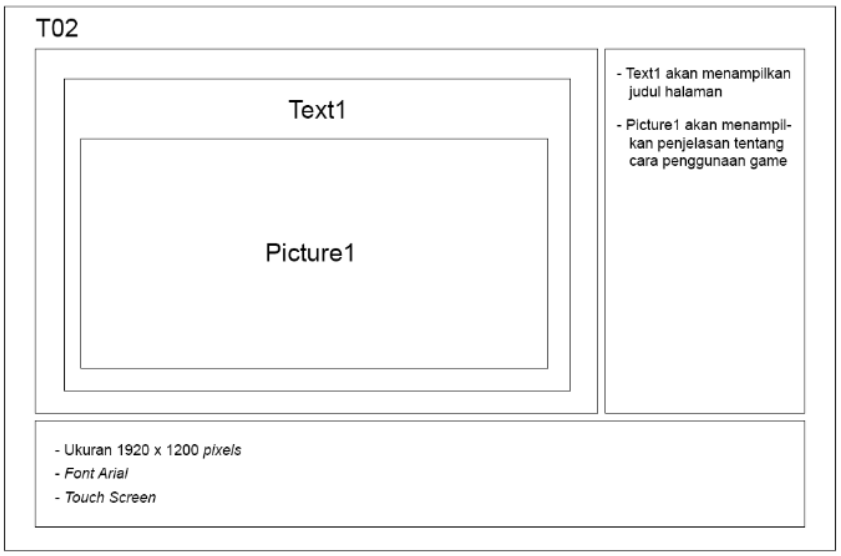
1. Perancangan Antarmuka Main Menu



Gambar 3. 29 Antarmuka Main Menu

Gambar 3.29 menggambarkan perancangan tampilan main menu pada *game* duck hunt.

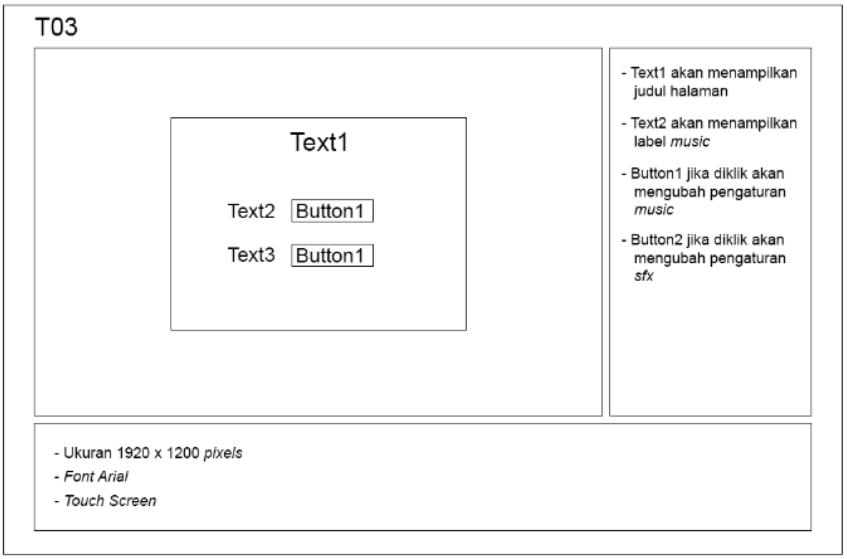
2. Perancangan Antarmuka Cara Penggunaan



Gambar 3. 30 Antarmuka Cara Penggunaan

Gambar 3.30 menggambarkan perancangan tampilan menu cara penggunaan yang ada di dalam *game* duck hunt.

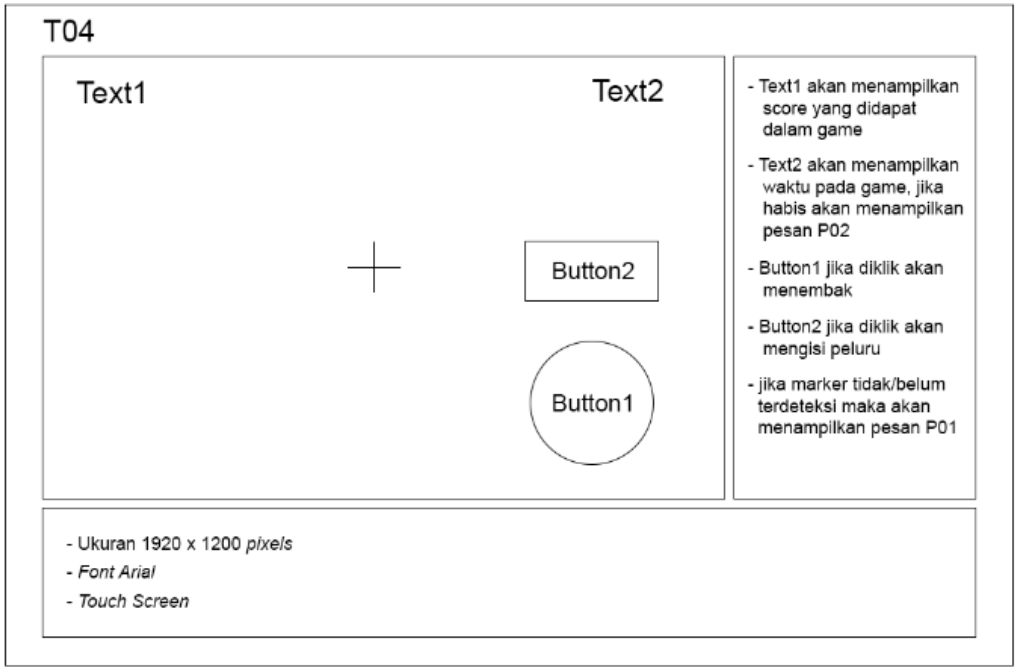
3. Perancangan Antarmuka Pengaturan



Gambar 3. 31 Antarmuka Pengaturan

Gambar 3.31 menggambarkan perancangan menu pengaturan pada *game* duck hunt

4. Perancangan Antarmuka Dalam Game



Gambar 3. 32 Antarmuka Dalam Game

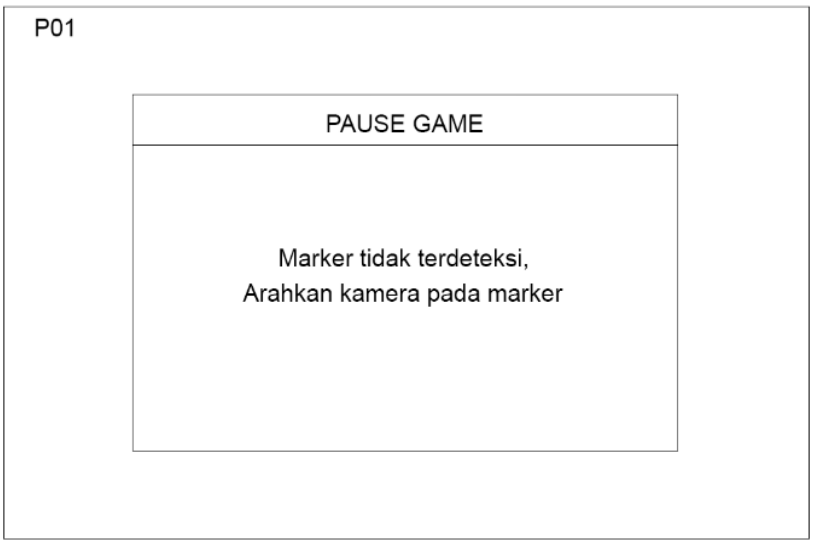
Gambar 3.32 menggambarkan perancangan antarmuka di dalam game duck hunt.

3.3.3.1 Perancangan Pesan

Perancangan pesan merupakan salah satu bagian dari perancangan sistem yang akan menunjukan pesan-pesan di dalam game duck hunt kepada pengguna. Perancangan pesan dalam game duck hunt digambarkan pada gambar 3.33 sampai dengan gambar 3.34

1. Perancangan Pesan Pause Game

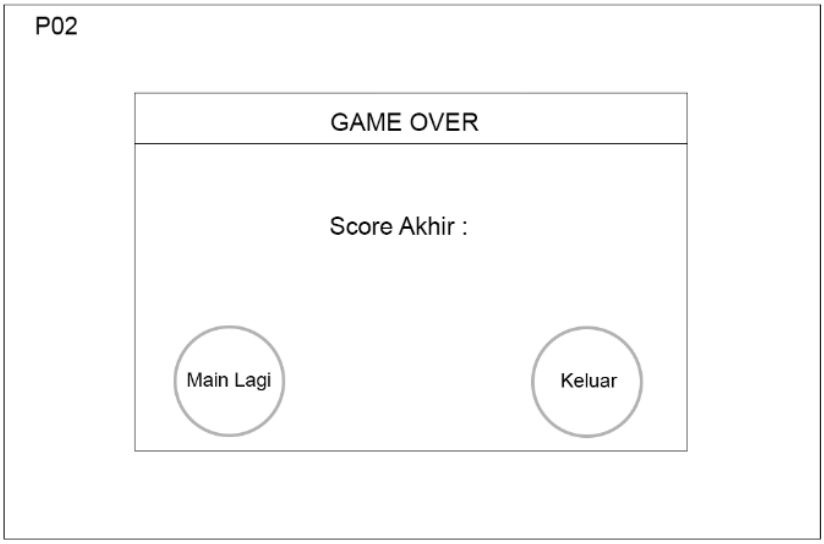
Pesan pause game ditampilkan ketika marker tidak terdeteksi oleh sistem pada saat memainkan game. Perancangan pesan pause game digambarkan pada gambar 3.33



Gambar 3. 33 Pesan Pause Game

2. Perancangan Pesan Game Over

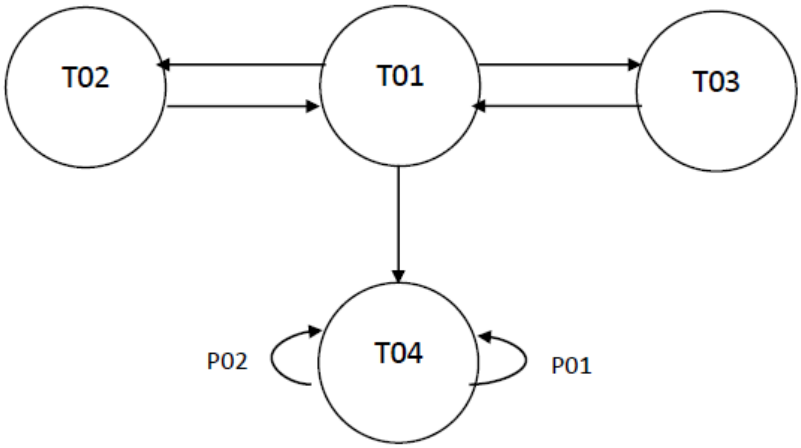
Pesan game over akan ditampilkan pada saat waktu dalam game duck hunt telah habis. Perancangan pesan game over digambarkan pada gambar 3.34



Gambar 3. 34 Pesan Game Over

3.3.4 Jaringan Semantik

Jaringan semantik merupakan gambaran grafis yang menunjukan antar obyek (lingkaran) dan garis yang menggambarkan formasi antar obyek tersebut. Adapun jaringan semantik *game duck hunt* dapat dilihat pada gambar 3.35



Gambar 3. 35 Jaringan Semantik

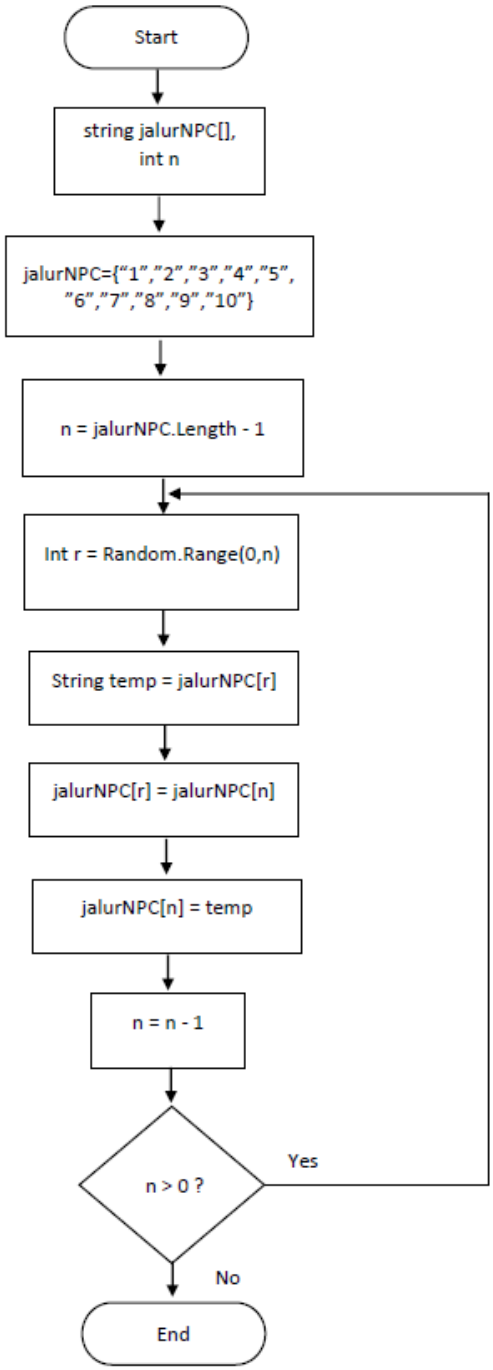
Gambar 3.35 menggambarkan jaringan semantik yang terdapat di dalam *game duck hunt*. Adapun keterangannya adalah sebagai berikut:

a. T01 : merupakan antarmuka untuk tampilan main menu
b. T02 : merupakan antarmuka untuk tampilan menu cara penggunaan
c. T03 : merupakan antarmuka untuk tampilan menu pengaturan
d. T04 : merupakan antarmuka untuk tampilan dalam *game*
e. P01 : merupakan pesan untuk *pause game*
f. P02 : merupakan pesan untuk *game over*

3.3.5 Perancangan Method

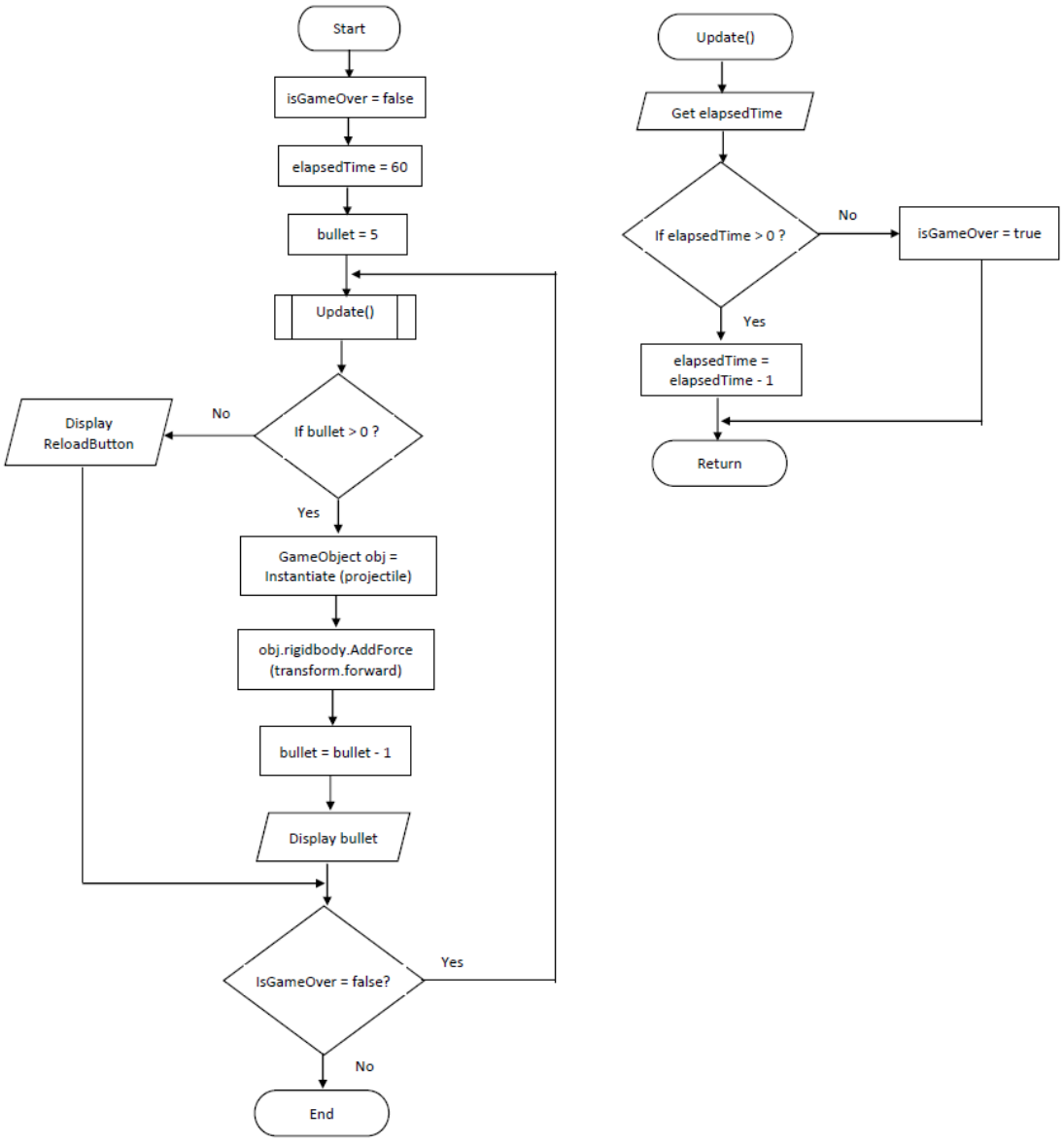
Perancangan *method* merupakan perancangan yang berfungsi untuk mendeskripsikan *method* yang berada di dalam *game duck hunt* menggunakan teknologi *augmented reality*. Adapun perancangan *method* yang digunakan dalam *game duck hunt* yang akan dikembangkan beberapa diantaranya adalah:

1. Perancangan *method* GameFunc.Shuffle(), merupakan method yang dijalankan untuk melakukan pengacakan pada jalur pergerakan NPC dalam game duck hunt. Algoritma yang digunakan pada method ini adalah Fisher-Yates Shuffle. Perancangannya dapat dilihat pada gambar 3.36



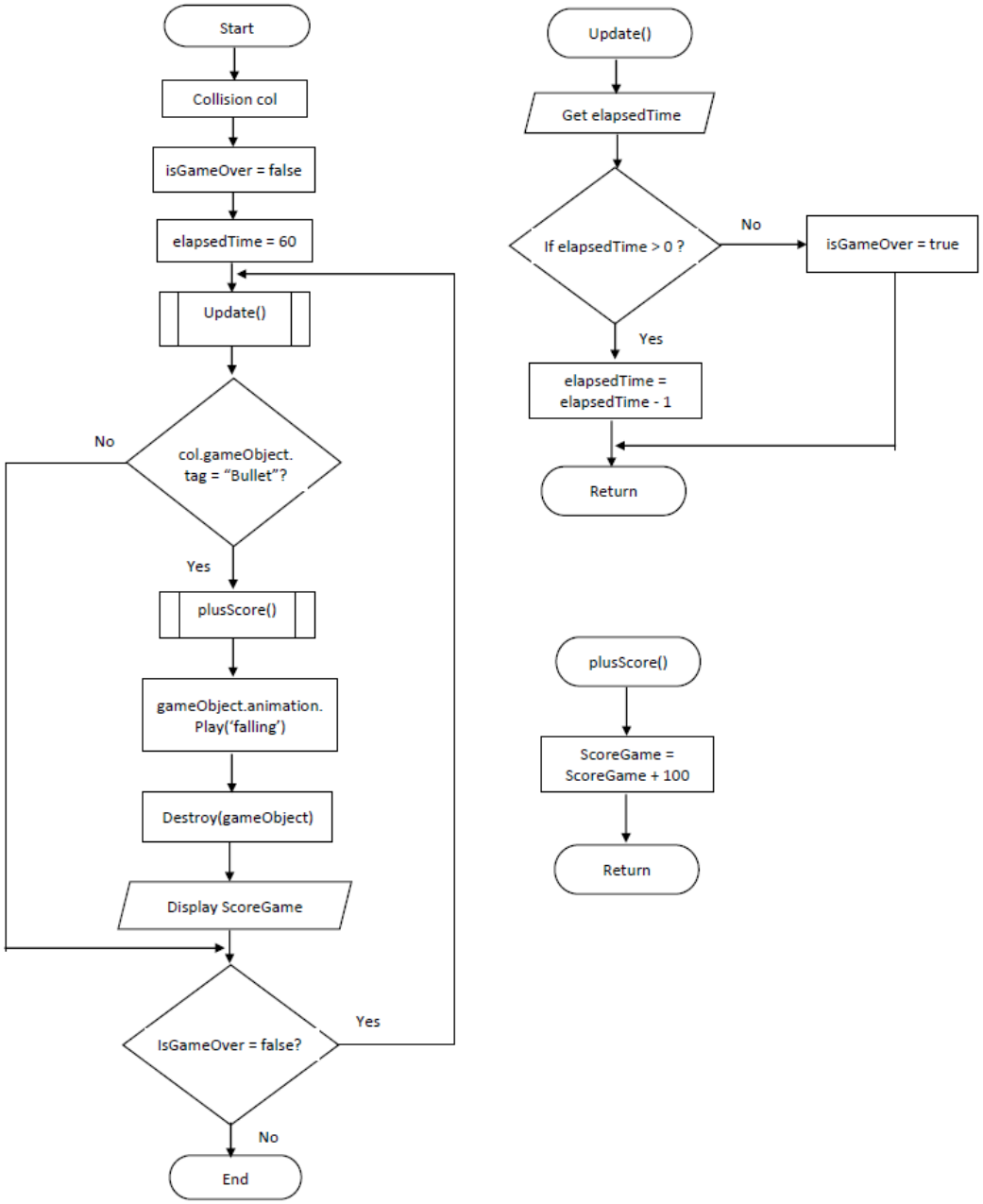
Gambar 3. 36 Flowchart GameFunc.Shuffle()

2. Perancangan *method* GameFunc.shootFire() di kelas GameFunc. *Method* ini dijalankan untuk melakukan tembakan dalam game ketika pemain menekan tombol tembak. Perancangannya dapat dilihat pada gambar 3.37



Gambar 3. 37 Flowchart GameFunc.ShootFire()

3. Perancangan *method* OnCollisionEnter() di kelas MoveObject. *Method* ini dijalankan ketika karakter bebek bertabrakan dengan sesuatu. Perancangannya dapat dilihat pada gambar 3.38



Gambar 3. 38 Flowchart MoveObject.OnCollisionEnter()