BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

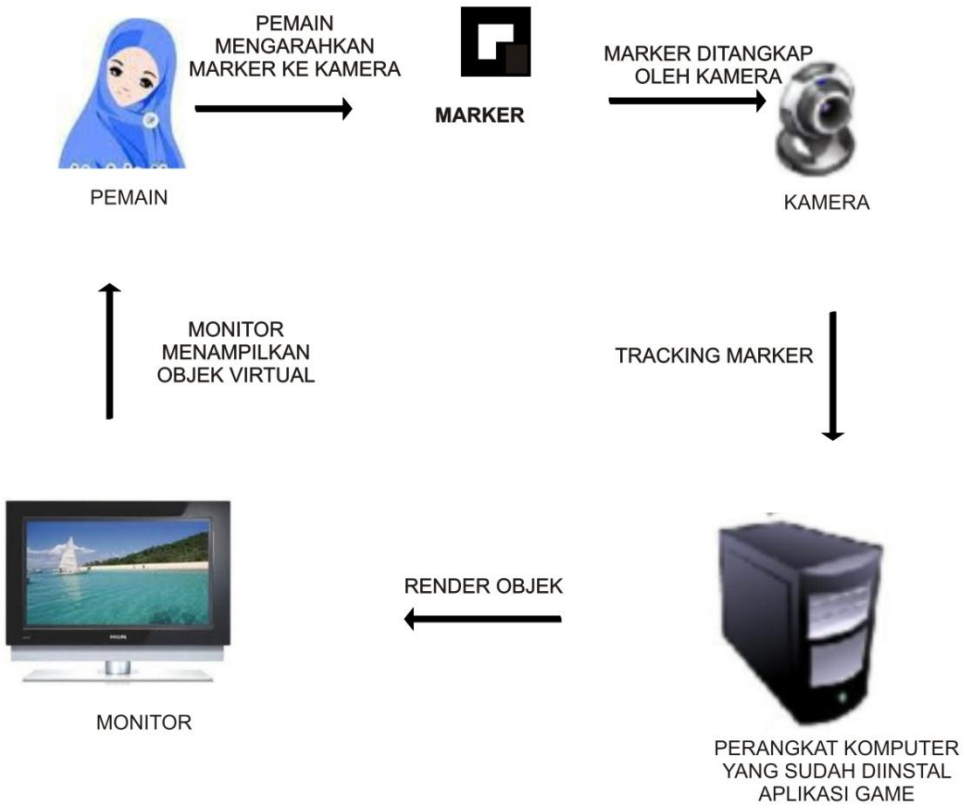
Analisis sistem dapat didefinisikan sebagai penguraian dari suatu sistem informasi yang utuh kedalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan, kesempatan-kesempatan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya. Sistem yang dibuat merupakan permainan berbasis *augmented reality* yaitu permainan menggelindingkan bola. Permainan yang dibuat seolah-olah pengguna dapat berinteraksi langsung dengan obyek-obyek 3 dimensi yang dibuat.

Permainan ini dibuat dengan mengambil latar dilingkungan nyata yang kemudian gabungkan dengan obyek-obyek 3D melalui kamera. Orientasi dan posisi *marker* akan dideteksi lewat *frame-frame* yang ditangkap oleh kamera. Setelah *marker* terdeteksi oleh kamera, maka akan didapatkan matriks transformasi yang dapat digunakan untuk transformasi seluruh obyek yang ada dalam permainan.

3.1.1 Analisis Arsitektur Sistem

Pada arsitektur permainan yang akan dibangun terdiri dari beberapa komponen, yaitu ; pemain adalah *user* yang menggunakan aplikasi permainan berbasis *augmented reality*, pemain mengarahkan *marker*/ penanda sehingga

marker dapat tertangkap olah kamera. Kemudian dari gambar yang didapat dari kamera sistem komputer melakukan *tracking marker* untuk mengidentifikasi *marker* yang digunakan oleh pemain. Komputer melakukan render obyek-obyek 3D yang digunakan dalam permainan. Pemain dapat melihat hasil manipulasi system melalui layar komputer/ monitor. Gambaran arsitektur sistem dapat dilihat pada gambar 3.1.



Gambar 3.1 Arsitektur Aplikasi

3.1.2 Deskripsi Masalah

Deskripsi masalah adalah suatu gambaran masalah yang diangkat dalam penulisan skripsi tentang mengimplentasikan teknologi *augmented reality* pada

permainan yang dapat melatih syaraf motorik halus anak. salah satu permainan yang dapat melatih syaraf motorik halus anak usia 4 tahun keatas adalah permainan menggelindingkan bola, maka permainan yang akan diimplementasikan kedalam teknologi *augmented reality* adalah permainan menggelindingkan bola. Seperti hal yang sama saat melakukan permainan menggelindingkan bola didunia nyata, permainan menggelindingkan bola ini didesain agar pemain dapat melatih keterampilannya dalam menggelindingkan bola kearah yang ditentukan.

Keterampilan menggelindingkan bola ini hanya dapat dicapai jika kontrol permainan ini tidak menggunakan kontrol *keyboard* atau *joystick* dalam memainkannya. Untuk memainkan permainan ini, pemain dapat mengontrol permainan melalui marker yang telah disediakan. Cara mengontrol permainan menggunakan marker yang telah disediakan adalah dengan cara mencondongkan marker kearah depan, belakang, kiri dan kanan. Agar kontrol menggunakan marker ini dapat tercapai, maka dalam pemogramannya dibutuhkan matriks *euler* sebagai representasi matriks rotasi 3D.

Matriks *euler* adalah representasi matriks rotasi grafik 3D yang terdiri dari sumbu x, y, dan z. Untuk proses kontrol *game* dengan *marker* dibutuhkan sumbu yang sama dengan sumbu *tracking marker*, maka terjadi tumpukan sumbu. Dengan *matrix euler* hal tersebut dapat dilakukan dalam proses kontrol permainan dengan menggunakan *marker*.

Konsep permainan yang dibangun adalah terdapat sebuah obyek labirin virtual, yang dibangun menggunakan obyek 3D *cube* (persegi). Kemudian

didalam labirin virtual tersebut terdapat bola virtual yang dapat bergerak secara menggelinding ke segala arah. Agar tujuan dari menggelindingkan bola ini memiliki arah, maka bola virtual ini memiliki untuk misi menabrak bola virtual lain. Jika bola virtual bertabrakan dengan bola virtual yang bergerak, maka bola virtual tersebut akan menghilang. Untuk membuat permainan lebih intuitif maka permainan ini juga harus menerapkan konsep deteksi tabrakan. Deteksi tabrakan yang dirancang adalah deteksi tabrakan antara bola dengan dinding labirin dan deteksi tabrakan antara bola dengan bola.

3.1.3 Analisis Masalah

Permainan ini akan memanfatkan *papervision3d* dalam membentuk obyek-obyek 3D. *Papervision3d* memiliki fungsi membentuk beberapa obyek primitif seperti *cube* (persegi), *sphere* (bola) dan sebagainya. Untuk mendesain obyek kotak-kotak virtual yang membentuk labirin permainan ini memanfaatkan fungsi *cube* dari *library papervision3d*. Sedangkan untuk obyek virtual bola akan memanfaatkan fungsi *sphere* pada *papervision3d*. Agar permainan ini lebih intuitif maka permainan ini harus menerapkan konsep deteksi tabrakan antara *sphere* dan *cube* juga *sphere* dan *sphere.* Deteksi ini dimaksudkan agar dalam menggelindingkan bola, bola akan terbatasi didalam labirin yang telah didesain dengan menggunakan fungsi *cube* dari *papervision3d*. Deteksi tabrakan ini dapat dilakukan dengan memanfaatkan *library JiglibFlash*.

Dewasa ini library untuk pengembangan AR sudah sangat berkembang dengan beberapa jenis bahasa pemograman. Beberapa *library* saat ini yang sedang berkembang dapat dilihat pada table 3.1.

Tabel 3.1 Perbandingan library untuk AR

Parameter		AR Library/ Toolkit			
		AR Toolkit	**Flartoolkit**	**D'Fusion Studio**	**IN2AR**
1	Lesensi	Gratis	Gratis	Gratis	Gratis
2	Bahasa pemograman	C++	ActionScript		ActionScript
3	Dukungan marker yang digunakan	Berbingkai hitam	Berbingkai hitam	Hampir semua Jenis gambar	Hampir semua Jenis gambar
4	Authoring / non authoring	Non authoring	Non authoring	Authoring	Non authoring
5	Format model 3D yang dapat ditampilkan	wrl	*.dae, *.obj, *.3ds	Seenetta	*.dae, *.obj, *.3ds
6	Dukungan platform	Multiplatform	Multiplatform	Multi platform	Multiplatform
7	Distribusi kepada pengguna	Harus install ARtoolkit di komputer pengguna	.swf	Menggunakan Dfusion@Home DesktopPlayer	*.swf

3.1.4 Analisis Pemecahan Masalah

Masing-masing library AR diatas memiliki kelebihan dan kekurangan tersendiri, adapun kelebihan IN2AR dibandingkan dengan library yang lain adalah:

1. Library IN2AR menyediakan lesensi berbayar (2212,50 EURO per proyek), juga menyediakan lisensi gratis. Perbedaan dari kedua lisensi ini adalah jika aplikasi dibuat dengan lisensi gratis maka harus menyertakan logo IN2AR dalam aplikasi yang dibangun.

2. Pengembang sangat aktif dalam menjawab pertanyaan seputar teknis IN2AR dalam forum resmi IN2AR.
3. IN2AR dapat menggunakan hampir semua desain gambar sebagai marker. Berbeda dengan Flartoolkit dan ARToolkit yang hanya dapat menggunakan desain gambar bingkai hitam sebagai marker.
4. Aplikasi yang dibuat dengan menggunakan IN2AR akan berekstensi *.swf, sehingga dapat ditampilkan di halaman web. Pengguna dapat menggunakan aplikasi yang dibangun tanpa harus menginstal aplikasi ini terlebih dahulu.
5. Library IN2AR menggunakan *ActionScript3* yang dapat dikomparasikan dengan library obyek 3D, seperti *papervision3d* dan jiglibflash yang menggunakan bahasa *ActionScript3.*

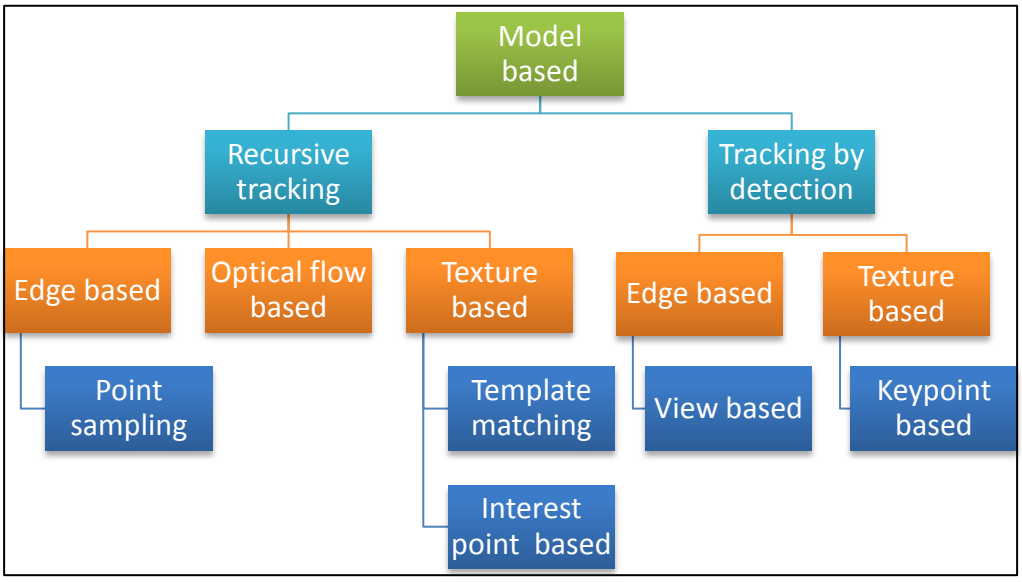
Adapun beberapa kekurangan yang dimiliki oleh library IN2AR adalah sebagai berikut:

1. Pengguna library IN2AR menggunakan *ActionScript3* oleh sebab itu untuk dapat menggunakan library ini pengguna dituntut untuk dapat memahami bahasa pemograman ActionScript3. Aktivitas di forum IN2AR akan sangat membantu jika pengguna mengalami kesulitan.
2. Desain gambar yang akan dijadikan sebagai marker dikonversi terlebih dahulu ke ekstensi *.ASS, untuk library yang menggunakan lisensi gratis desain gambar harus dikirimkan terlebih dahulu ke *teamsuport* IN2AR.

Teknik yang digunakan dalam proses pelacakan atau *tracking marker* yang dilakukan oleh library IN2AR adalah teknik berbasis model yang dapat dibagi menjadi 2 kategori yaitu:

1. Pelacakan rekursif (*Rekursif Tracking*), yaitu sebelum obyek ditampilkan dilakukan perhitungan perkiraan terlebih dahulu.
2. Pelacakan dengan pendeteksian (*Tracking by detection*), yaitu klasifikasi dilakukan perhitungan pelacakan tanpa perkiraan terlebih dahulu, yang memungkinkan otomatis inisialisasi dan pemulihan dari kegagalan.

Skema teknik pelacakan berbasis model dapat dilihat pada gambar 3.2.



Gambar 3.2 *Model Bases Online Monucular Markerless AR Taxonomy*

3.1.4.1 Deteksi Tabrakan (*Collision Detection*)

Collision Detection adalah fungsi yang dapat mendeteksi tabrakan antara 2 obyek atau lebih. *Collision detection* juga berguna untuk menentukan posisi dari

satu objek dengan objek yang lain sehingga tidak ada objek yang saling menembus. Sehingga game yang akan dibuat memiliki kesamaan dengan realita yang ada. Deteksi tabrakan adalah salah satu fungsi yang telah dimiliki oleh JiglibFlash. Perancangan permainan ini menggunakan obyek *sphere* yang sebagai bola yang bergerak dan bola serta *cube* sebagai tembok labirin. Teknik *collision detection* pada permainan antara *sphere* dan *cube* menggunakan teknik deteksi *Sphere-Plane collision detection*, dan tabrakan antara *sphere* dan *sphere* menggunakan deteksi *Sphere collision detection.* Teknik yang digunakan dalam obyek yang berbentuk bola atau *sphere*. *Collision detection* yang dilakukan adalah sebagai berikut:

1. *Sphere-Plane Collision Detection*

   Pada deteksi *Sphere-Plane Collision Detection* ini menghitung nilai radius atau titik tengah dari obyek simetri *sphere* dan *cube*. Nilai radius atau titik tengah dari obyek simetri *sphere* dan *cube* nanti yang digunakan dalam menghitung perpotongan obyek. Jika jarak dari pusat *sphere* ke sebuah obyek kurang dari atau sama dengan jari-jari bola, maka tabrakan telah terjadi.

$$d = (C - P0).n \; ; D = -P0.n \tag{3.1}$$

$$\text{Maka } d = n.C + D \tag{3.2}$$

Keterangan :

C = titik pusat obyek *sphere*

P0 = setiap titik pada *cube*

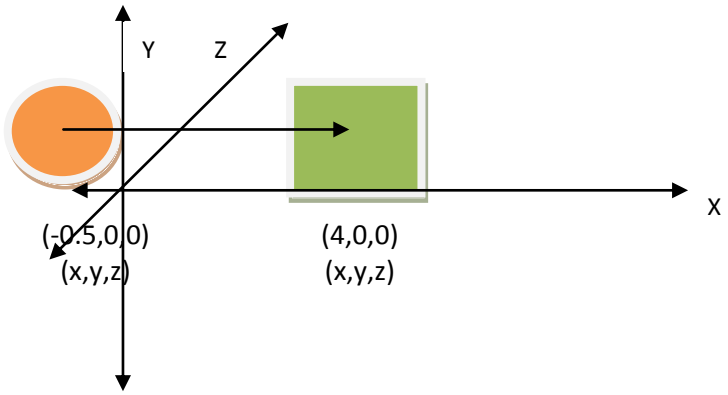
n = unit normal pada *cube*

persamaan *plane/cube* dengan vector normal N= (A,B,C) melalui titik (x,y,z) adalah

$A(x - x_0) + B(y - y_0) + C(z - z_0) = 0$ atau $Ax + By + Cz + D = 0$, maka nilai $D = -Ax_0 - By_0 - Cz$ dengan demikian besar unit normal vector dengan notasi n adalah:

$$n = \frac{N}{||\, N \,||} = \frac{(A,B,C)}{\sqrt{A^2+B^2+C^2}} \tag{3.3}$$

Adapun simulasi tabrakan antara *cube* dan *sphere* dapat dilihat pada gambar 3.3:



Gambar 3.3 Perhitungan Deteksi Tabrakan Antara *Sphere* Dan *Cube*

Posisi kotak untuk sumbu x,y,z adalah (4,0,0) dan posisi bola pada sumbu x,y,z yaitu (-0.5,0,0). Radius dari bola = *geometry sphere* (0.8), ukuran kotak = *geometry cube* (1.5, 1.5, 1.5), titik P_0 = (-0.5+(1.5/2)) = 0.25, N=(0.25,0,0)

Persamaan *Plane/cube* $Ax + By + Cz + D = 0$

0.25 + D = 0,

Maka D = -0.12.

Unit vektor n adalah (persamaan 3.3):

$$n = \frac{N}{||\ N\ ||} = \frac{(A, B, C)}{\sqrt{A^2 + B^2 + C^2}} = \frac{(0.25)}{\sqrt{0.25^2}} = 1$$

Jarak (persamaan 3.2) d = 4 – 0.25 = 3.75.

Radius dari bola = 0.8, maka d > r, dari hasil contoh posisi kotak dan bola diatas didapatkan bahwa bola belum bertabrakan dengan kotak.

2. *Sphere-sphere Collision Detection*

Deteksi kedua yang dilakukan dalam permainan ini adalah deteksi tabrakan antara *Sphere* (bola) jika bersentuhan dengan obyek *sphere* (bola bergerak). Dalam hal ini deteksi yang dilakukan adalah menentukan posisi dari *sphere* (bola). Kemudian dilakukan penentuan posisi x,y,z dari *sphere* (bola). Penentuan posisi x,y,z dari *sphere* (bola) untuk membuat kondisi jika *sphere* (bola) bersentuhan dengan *sphere* (bola bergerak), maka posisi x dari *sphere* (bola) akan berubah. Perubahan posisi dari *sphere* (bola), mengakibatkan pemanggilan kondisi untuk menghapus obyek *sphere* (bola). Dalam perhitungan deteksi tabrakan menggunakan rumus :

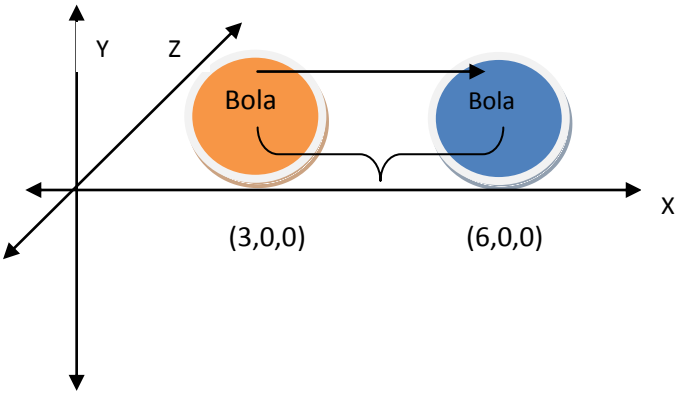
D = (C2-C1) (3.4)

$$d = \sqrt{(D.x)^2 + (D.y)^2} + (D.z)^2 \quad (3.4)$$

Akan terjadi tabrakan jika

d < (r1 + r2)

Simulasi tabrakan antara *sphere* (bola bergerak), *sphere* (bola). Bola dengan r1 = *geometry sphere* (radius 1) berada pada posisi x,y,z (3,0,0), dan posisi bola berada pada sumbu x,y,z (6,0,0) dengan (radius 1). Ilustrasi gambar tabrakan antara bola dan bol dapat dilihat pada gambar 3.4



Gambar 3.4 Perhitungan Deteksi Tabrakan Antara *Sphere* dan *Sphere*

Perhitungan jarak (d) dari kedua obyek:

$$d = \sqrt{(D.x)^2 + (D.y)^2 + (D.z)^2}$$

$$D = (C2 - C1)$$

$$D = (6 - 3) = 3;$$

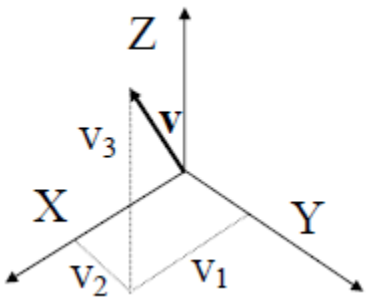
$$r1 + r2 = 0.1 + 0.1 = 0.2$$

Sehingga nilai d adalah:

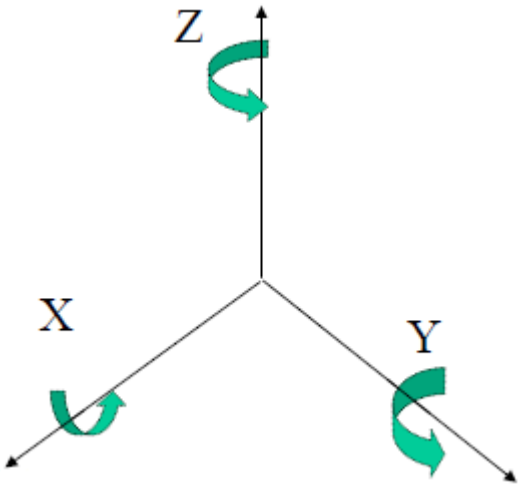
$d = \sqrt{3^2} = 9$ maka 9 > 0.2 atau d > (r1+r2), ini yang disebut belum terjadinya tabrakan.

3.1.4.2 Kontrol Game Menggunakan Matriks Euler

Untuk menggerakan bola sesuai dengan gerakan *marker*, perlu diketahui nilai transformasi *marker* terlebih dahulu. Setelah didapat nilai transformasi 3dimensi berbentuk matriks 4x4 kemudian diubah ke matriks *euler*. Metode ini sudah ada dalam library *papervision3d*. gambar matriks *euler* dalam vektor dapat dilihat pada gambar 3.5 A dan gambar matriks *euler* dalam rotasi dalam ruang 3D dapat dilihat pada gambar 3.5 B



Gambar A



Gambar B

Gambar 3.5 A (Vektor diruang 3D), B (Arah Rotasi Diruang 3D)[11].

Matriks rotasi untuk grafis 3D dengan representasi *euler* terdiri atas rotasi terhadap sumbu x, y, dan z. Karena perkalian matriks bersifat komulati, maka representasi *euler* memiliki 12 urutan rotasi yang dihasilkan, yaitu xyz, xzy, xyx, xzx, yxz, yzx, yxy, yzy, zxy, zxy, zyx, zxz, zyz.

Matriks rotasi tehadap sumbu x:

$$Rx(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & sin\alpha \\ 0 & -sin\alpha & cos\alpha \end{bmatrix} \quad (3.5)$$

Matriks rotasi tehadap sumbu y:

$$Ry(\beta) = \begin{bmatrix} cos\beta & 0 & -sin\beta \\ 0 & 1 & 0 \\ sin\beta & 0 & cos\beta \end{bmatrix} \quad (3.6)$$

Matriks rotasi terhadap sumbu z:

$$Rz(\gamma) = \begin{bmatrix} cos\gamma & sin\gamma & 0 \\ -sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.7)$$

Pada fungsi *rendering* akan didapatkan matriks transformasi dari posisi marker. Kemudian fungsi tersebut melakukan fungsi moveball dengan parameter matriks gerak marker. Fungsi moveball dibuat dengan matriks 3x4 untuk menampung matriks transformasi marker. Kemungkinan posisi bola akan ditentukan oleh rotasi objek berdasarkan perkalian matriks diatas.

3.2 Kebutuhan Non Fungsional *Game*

Analisis kebutuhan non fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan system. Spesifikasi ini juga melipiti elemen ataupun komponen-komponen apa saja yang dibutuhkan system yang akan dibangun sampai system tersebut diimplementasikan. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang dibutuhkan system. Pada analisis kebutuhan system non-fungsional ini dijelaskan mengenai analisis perangkat keras (*hardware*), perangkat lunak (*software*) dan pengguna (*user*) .

Games memecahkan bola merupakan permainan berbasis *Augmented reality*. *Games* memecahkan bola berbasis *Augmented reality* dapat dijadikan alternatif permainan baru untuk melatih gerak otot motorik halus anak maupun dikalangan pencinta permainan. Karena *games* memecahkan bola adalah produk berbasis AR yang menggunakan *marker* sebagai pola pelacakan atau *tracking marker*, maka dibutuhkan computer dan perangkat lunak sebagai pengolah citra dan *marker*. Berikut ini adalah perangkat lunak yang dibutuhkan oleh *games* memecahkan bola.

3.2.1 Analisis Perangkat Keras

Games memecahkan bola adalah program yang membutuhkan grafis video dengan kemampuan relative tinggi. Hal ini karena model-model yang ditampilkan menggunakan teknik *rendering.*

Berdasarkan studi literatur terhadap proyek-proyek pengembangan teknologi AR, maka diperolah spesifikasi minimum perangkat keras untuk teknologi AR seperti berikut ini:

1. *Processor* pentum 4 > 1 GHz atau setara.
2. RAM minimal 2 GB
3. Ruang sisa *hardisk* minimal 4 GB
4. *Motherboard* dengan *chipset* yang kompatibel dengan VGA *card* yang dipakai.
5. Memiliki opsi konektifitas USB 2.0 dan nirkabel.
6. Optimum menggunakan VGA *card* dengan kemampuan me-*render* grafis 3D, seperti GeForce 6xxx atau ATI 1xxx series.
7. *Webcam*

   Webcam dipakai untuk menangkap citra yang kemudian diproses oleh *tracker library*. Hasil dari proses ini akan menghasilkan matriks transformasi *marker* relative terhadap *webcam*. Saat ini di pasaran sudah cukup banyak *webcam* dengan harga yang terjangkau.
8. *Marker*

   Marker dibutuhkan oleh *tracking library* untuk menempatkan model virtual di dunia nyata dengan cara menentukan koordinat *marker*

relative terhadap kamera. Dengan mngetahui koordinat relative tersebut kita dapat berinteraksi dengan AR. *Marker* yang dibutuhkan yaitu, 1 buah *marker* yang telah didesain untuk permainan.

9. *Proyektor*

   Proyektor dibutuhkan untuk jumlah partisipan yang banyak, sehingga partisipan yang tidak perlu melihat ke layar monitor yang kecil. Proyektor sangat berguna dalam kerja kolaborasi permainan ini.

3.2.2 Analisis Perangkat Lunak

Perangkat lunak digunakan dalam sebuah sistem merupakan kumpulan perintah-perintah yang diberikan kepada perangkat keras agar saling berinteraksi untuk melakukan suatu tugas. Perangkat lunak yang dibutuhkan untuk membangun aplikasi adalah sebagai berikut :

1) Sistem operasi Windows Xp Pro2
2) FlashDevelop Version 4.0.0 RC3
3) Adobe Flash CS5
4) Adobe Dreamweaver CS5

3.2.3 Analisis Pengguna

Analisis pengguna dimaksudkan untuk mengetahui siapa saja pengguna yang terlibat system. Pengguna yaitu *public* yang dapat mengerti dan memahami komputer sehingga dapat menggunakan aplikasi yang akan dibangun. Karena permainan ini ditujukan untuk melatih kerja

motorik halus anak, dan usia anak yang menggunakan aplikasi in adalah 4 tahun keatas, oleh sebab itu anak harus didampingi orang tua. Permainan ini juga tidak hanya dapat dimainkan dikalang anak usia 4 tahun keatas akan tetapi dapat digunakan oleh umum yang memahami komputer.

3.3 Analisis Kebutuhan Fungsional

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

3.3.1 Pemodelan Sistem

Pemodelan sistem dilakukan dengan metode berorientasi Obyek dengan menggunakan standarisasi UML, dengan tahap – tahap penentuan siapa saja actor yang terlibat dan perancangan *use case specification, Interaksi diagram (sequence diagram* dan *collaboration diagram).*

3.3.1.1 *Use Case Diagram*

Use case diagram ini digunakan untuk menggambarkan hubungan sejumlah external actor dengan use case yang terdapat dalam system. *Use case* diagram ini hanya menggambarkan keadaan lingkungan system yang dapat dilihat dari luar oleh actor. Pada use case ini terdapat dua actor, yaitu pemain dan system. Penjelasan mengenai dua actor dapat dilihat pada table 3.2

Tabel 3.2 Aktor dalam *Use Case*

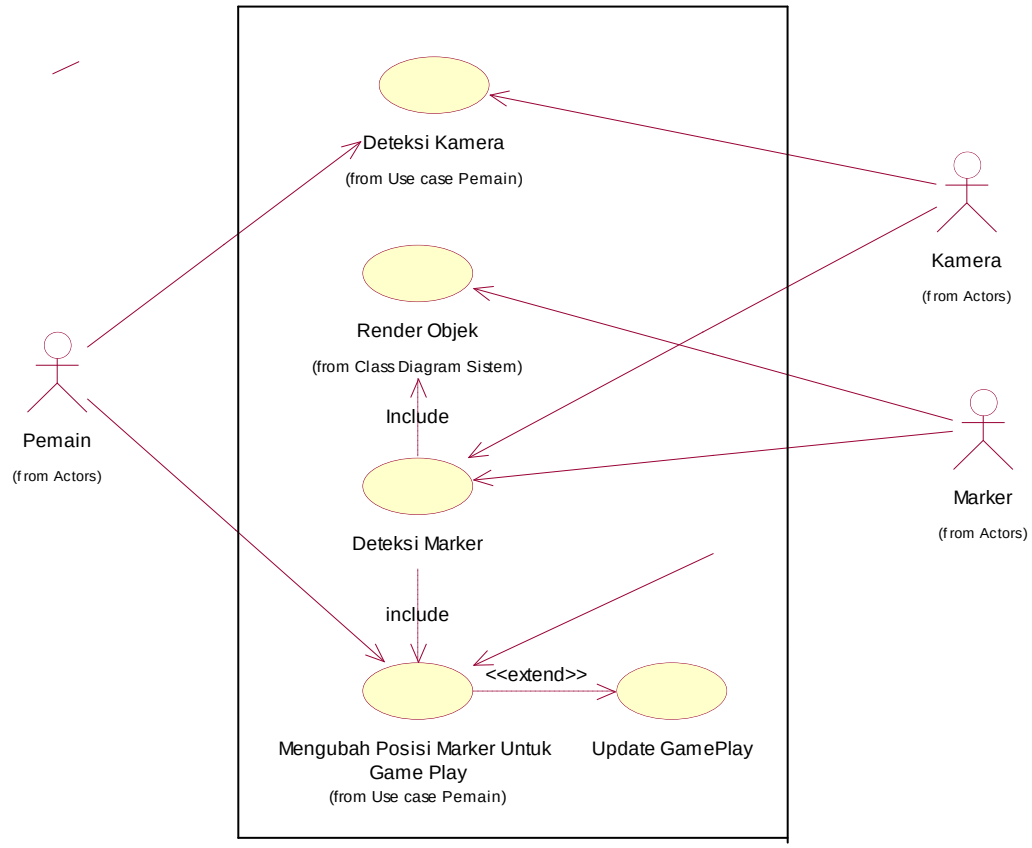
Nama Aktor	Definisi
Pemain	Orang yang berinteraksi dengan system, yaitu pengguna yang mengakses semua fungsi yang disediakan system.

Marker	Marker adalah alat bantu yang digunakan sebagai penanda dimana obyek virtual 3D akan ditampilkan.
Kamera	Kamera adalah alat bantu yang digunakan sebagai pendeteksi keberadaan marker.

3.3.1.2 *Skenario Use Diagram*

Use case pada permainan berhubungan dengan permainan dan penggunaan *augmented reality* pada permaianan. Dimana *augmented reality* digunakan dalam memanipulasi obyek-obyek maya dengan berinteraksi dengan obyek fisik di lingkungan nyata. Agar dapat melihat detail *uses case* dapat lihat pada table 3.3

Diagram *use case* pada gambar 3.6 dilakukan oleh actor dengan Pemain dimana diagram *use case* ini lebih bagaimana actor dapat menjalankan fungsi-fungsi yang terdapat pada permainan.



Gambar 3.6 *Use case Diagram*

Tabel 3.3 Deskripsi *Use Case Diagram*

NO	Kode *Use Case*	Nama *Use Case*	Keterangan
1	UC-P01	Mendeteksi Kamera	Aplikasi Mendeteksi Ketersediaan Kamera.
2	UC-P02	Deteksi Marker	System mendeteksi keberadaan marker melalui gambar yang ditangkap oleh kamera
3	UC-P03	Render Obyek	Sistem akan melakukan render obyek pada awal aplikasi dijalankan .sistem akan merender gambar,tekstur dan model geometri.
4	UC-P04	Mengubah posisi	Pemain menggerakan /

		marker untuk *Game*Play	memiringkan media *marker* kearah kiri dan kanan untuk mengerakan jarum virtual bergerak mendekati bola-bola virtual, hingga bola-bola vitual menghilang karena jarum virtual mengenai bola virtual
5	UC-P05	Update *Game*Play	Setelah sistem mendeteksi perubahan matriks pada gambar *marker*, maka sistem akan melakukan update permainan terutama posisi jarum virtual dan bola virtual

Tabel 3.4 Spesifikasi Skenario : Mendeteksi Kamera

Nama *Use Case*	Mendeteksi Kamera
Nomor	UC-P01
Aktor	Pemain, kamera
Kondisi Awal	-
Kondisi Akhir	Kamera Terdeteksi
Alur Normal	1. Pemain memasang kamera pada perangkat komputer 2. Sistem mendeteksi ketersediaan kamera
Alur Alternatif	
Pengecualian	1.1 terdapat perangkat kamera pada laptop

Tabel 3.5 Spesifikasi Skenario: Deteksi *Marker*

Nama *Use Case*	Deteksi *Marker*
Nomor	UC-P04
Aktor	Pemain, Kamera, Marker

Kondisi Awal	-
Kondisi Akhir	Aplikasi siap mendeteksi *marker*
Alur Normal	1. Inisiasi DirectShowCapture 2. Inisiasi IN2AR*Marker*Tracker 3. Inisiasi *Marker*
Alur Alternatif	-
Pengecualian	1.1 Perangkat kamera tidak ditemukan 2.1 File kalibrasi tidak ditemukan 3.1 *Marker* belum diinisiasi

Tabel 3.6 Spesifikasi Skenario: Render Obyek

Nama *Use Case*	Render Obyek
Nomor	UC-P05
Aktor	Pemain, Kamera, Marker
Kondisi Awal	-
Kondisi Akhir	Semua konten berhasil dirender untuk digunakan dalam permaianan
Alur Normal	1. Render tekstur gambar 2. Simpan semua Tekstur Gambar yang dirender ke dalam *Dictionary* tekstur 3. Render model geometri 4. Simpan semua model geometri yang di render kedalam *dictionary model*
Alur Alternatif	-
Pengecualian	1.1. Filepart yang didefinisikan untuk melakukan render tekstur tidak ditemukan 2.1 Filepart yang didefinisikan untuk melakukan render model geometri tidak ditemukan

Table 3.7 Spesifikasi Skenario : Mengubah Posisi *Marker*

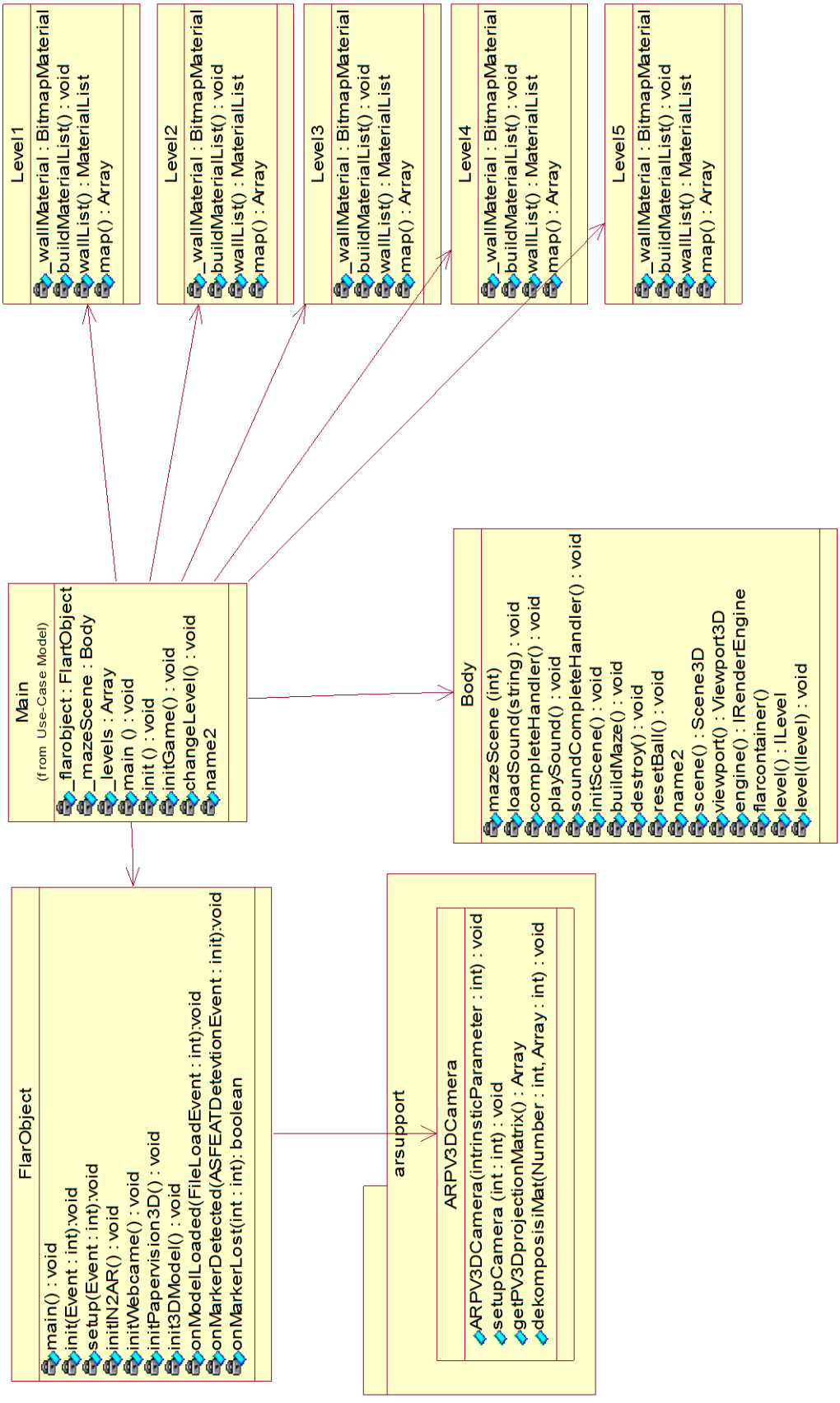
Nama *Use Case*	Mengubah Posisi *Marker* Untuk *Game*Play
Nomor	UC-P06
Aktor	Pemain, Kamera, Marker
Kondisi Awal	-
Kondisi Akhir	Update Transformasi matriks
Alur Normal	1. *Marker* tracker akan mendeteksi *marker* 2. Kalkulasi posisi *marker* dengan cara mendapatkan matriks transformasinya relative terhadap transformasi matriks *marker* permainan. 3. *Marker* node mendapatkan transformasi matriks 4. Update Matriks transformasi terhadap semua obyek yang menjadi child dari *marker* node
Alur Alternatif	-
Pengecualian	-

Table 3.8 Spesifikasi Skenario : *Update GamePlay*

Nama *Use Case*	Update *Game*Play
Nomor	UC-P07
Aktor	Pemain, Kamera, Marker
Kondisi Awal	Sistem mendeteksi perubahan matriks
Kondisi Akhir	Sistem akan melakukan Update *Game*Play
Alur Normal	1. Update Matriks transformasi terhadap semua obyek yang menjadi child dari *marker* node 2. Update seluruh obyek virtual
Alur Alternatif	-
Pengecualian	-

3.3.1.3 *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). Gambar 3.7 adalah kelas diagram dari Implementasi Teknologi Augmented Reality Pada Permainan *Rooling Ball*.

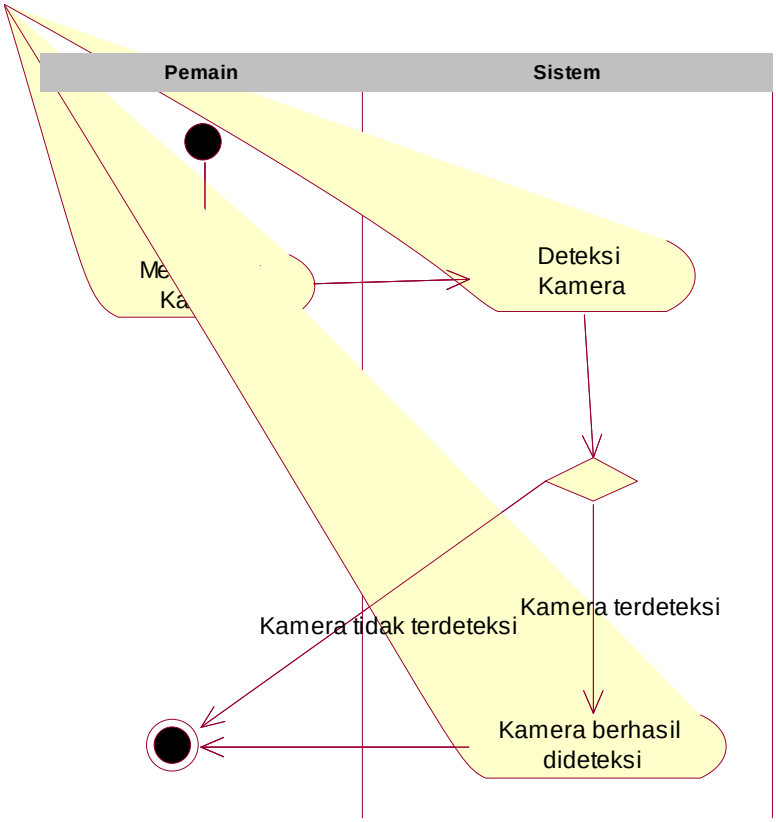


Gambar 3.7 *Class Diagram Game* Permainan

3.3.1.1 *Activity Diagram*

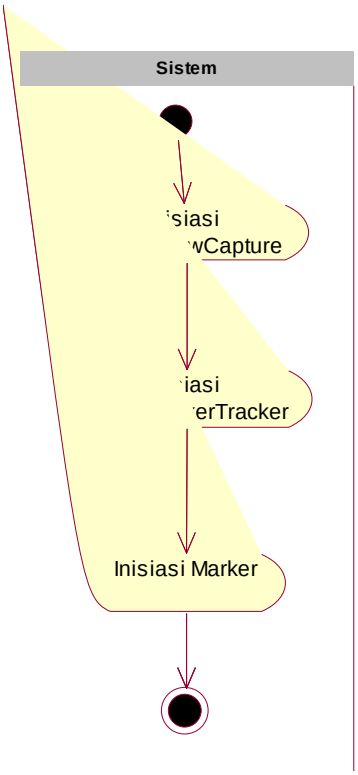
Diagram activity menggambarkan berbagai alir aktivitas dalam sistem yang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi dan bagaimana mereka berakhir. Penggambaran *activity* diagram memiliki kemiripan dengan *flowchart diagram. Activity diagram* memodelkan *event-event* yang terjadi pada *Use Case* dan digunakan untuk pemodelan aspek dinamis dari sistem.

1. *Activity diagram* terkait dengan Skenario Mendeteksi kamera dapat dilihat pada gambar 3.8.



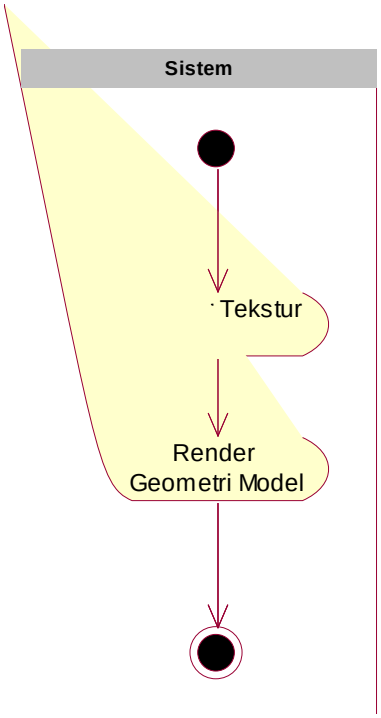
Gambar 3.8 *Activity Diagram* Mendeteksi Kamera

2. *Activity Diagram* terkait dengan *use case* Deteksi *Marker* dapat dilihat pada gambar 3.9.



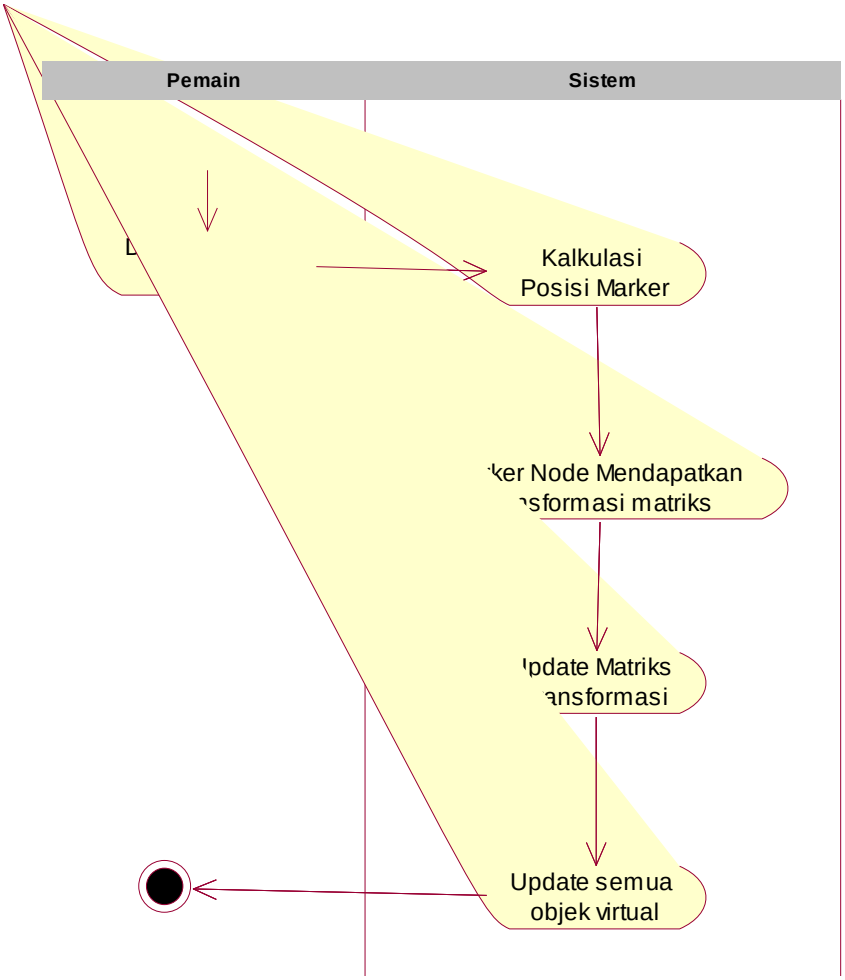
Gambar 3.9 *Activity Diagram* Deteksi *Marker*

3. *Activity Diagram* terkait dengan *use case* Render Obyek dapat dilihat pada gambar 3.10.



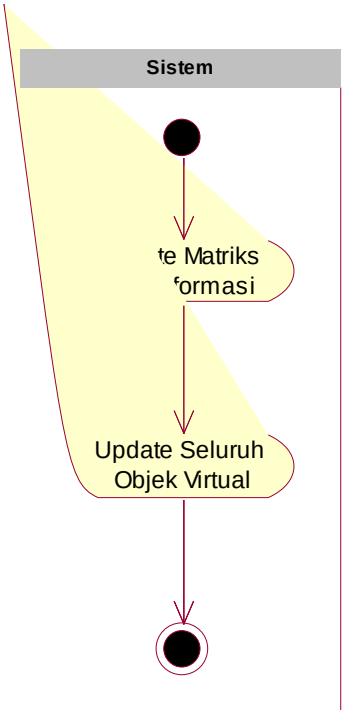
Gambar 3.10 *Activity Diagram* Render Obyek

4. *Activity diagram* terkait dengan *use case* mengubah posisi *marker* untuk *Game*Play dapat dilihat pada gambar 3.11.



Gambar 3.11 *Activity Diagram* Mengubah Posisi *Marker* Untuk *Game*Play

5. *Activity Diagram* terkait dengan *use case* update *Game*Play dapat dilihat pada gambar 3.12.



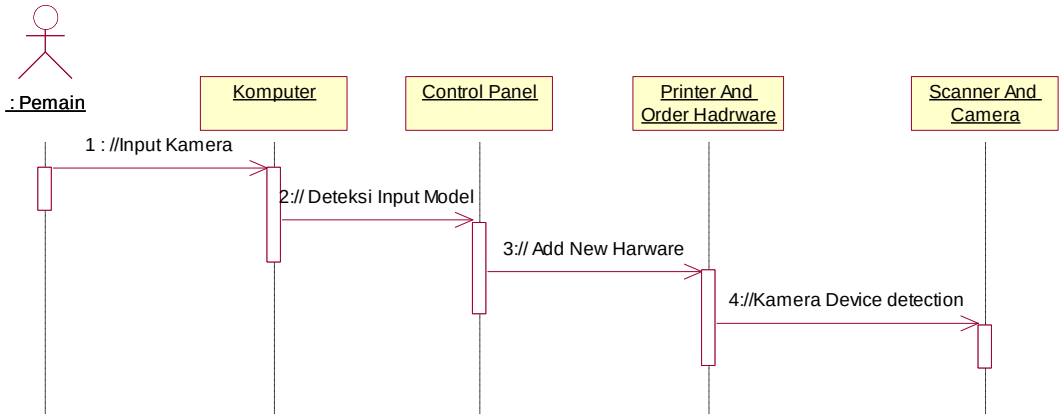
Gambar 3.12 *Activity Diagram* Update *Game*Play

3.3.1.2 *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.

1. *Sequence Diagram* Mendeteksi Kamera

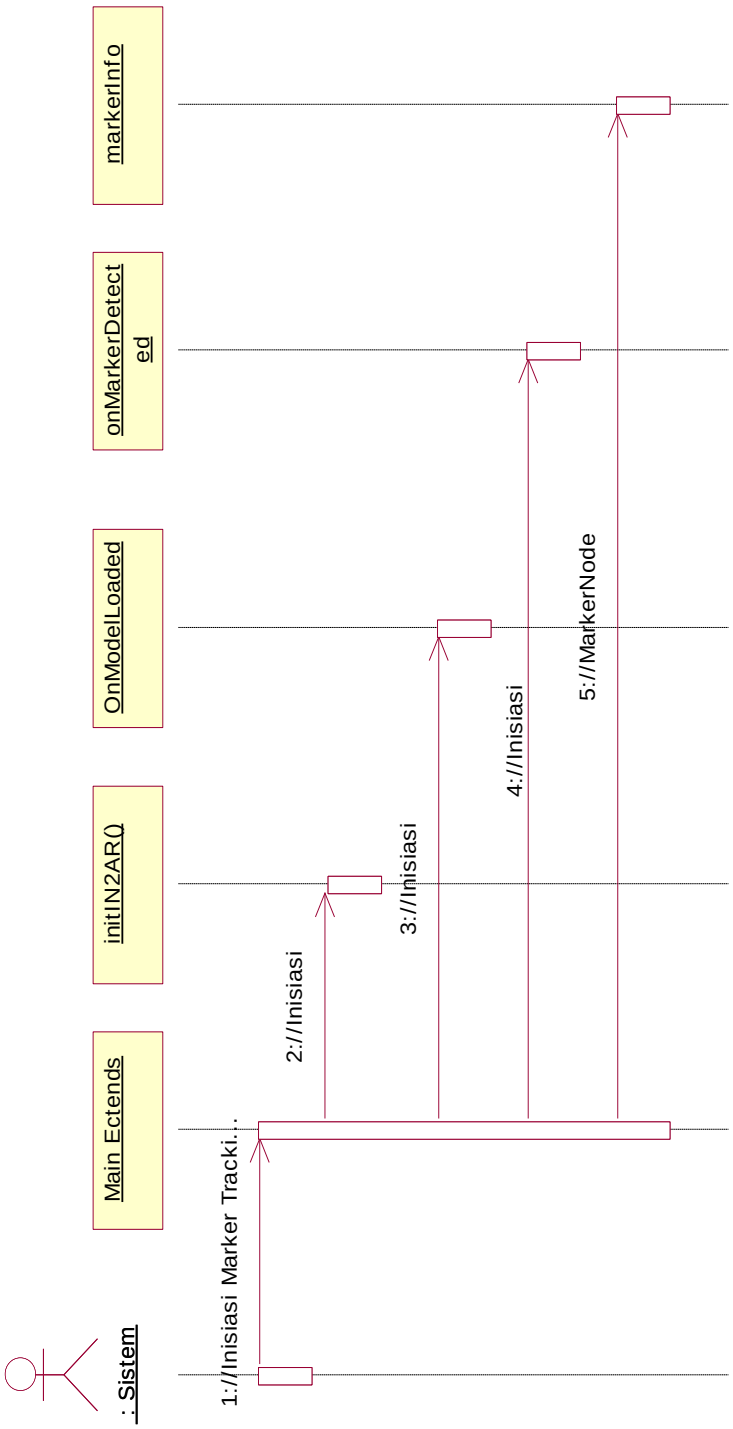
   Gambar *Sequence Diagram* Mendeteksi Kamera dapat dilihat pada gambar 3.13



Gambar 3.13 *Sequence Diagram* Mendeteksi Kamera

2. *Sequence Diagram* Deteksi Marker

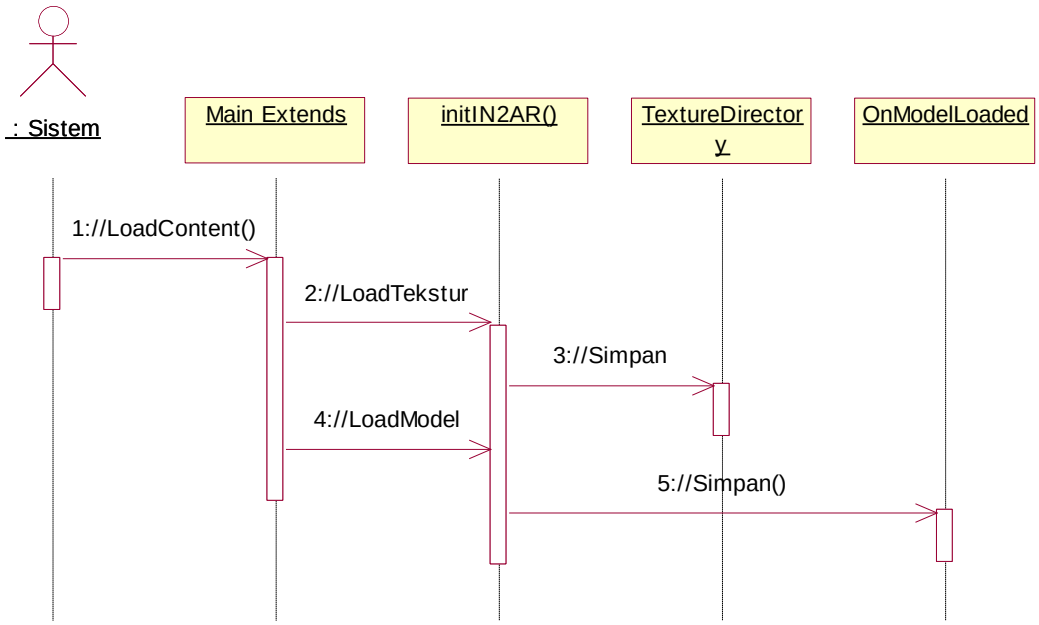
Gambar *Sequence Diagram* Deteksi Marker dapat dilihat pada gambar 3.14



Gambar 3.14 *Sequence Diagram* Deteksi *Marker*

3. *Sequence Diagram* Render Obyek

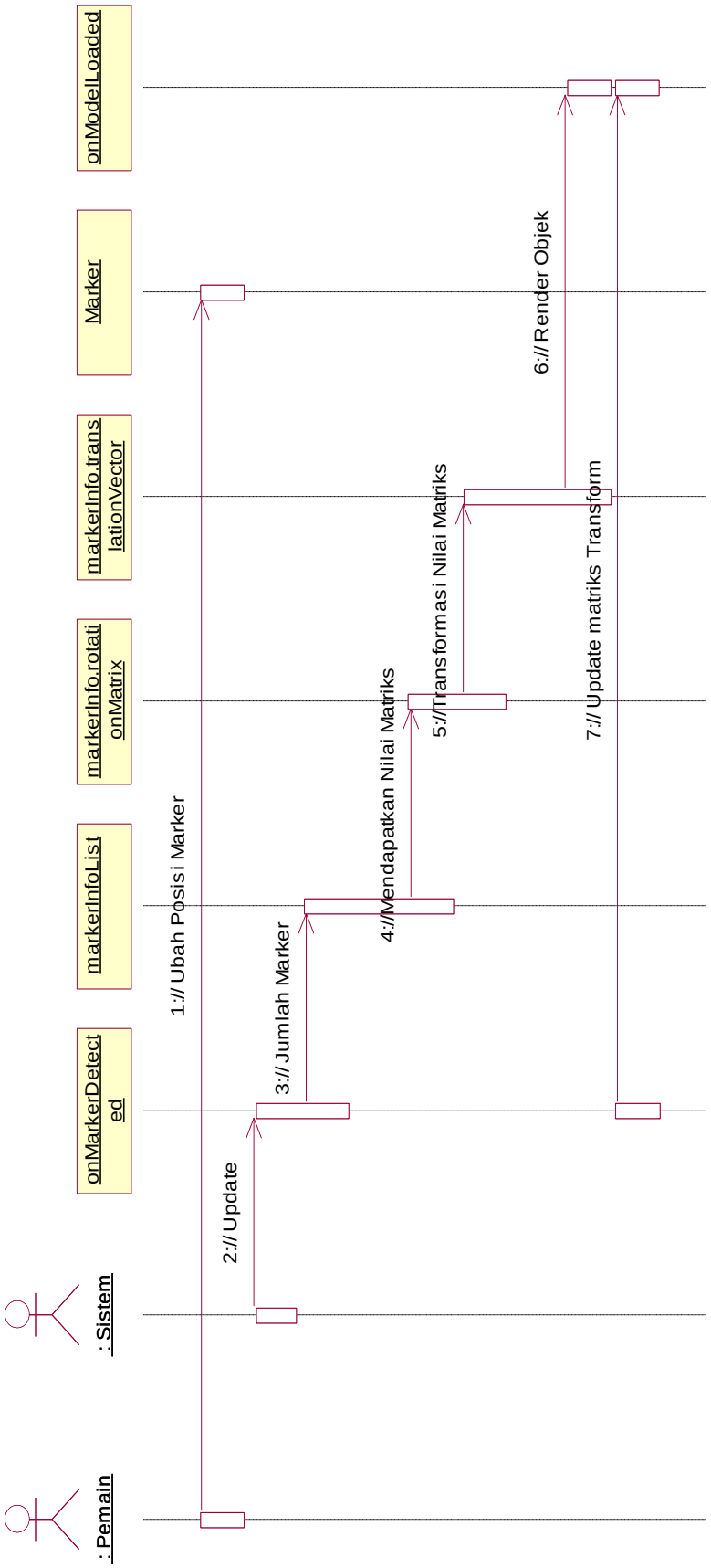
   Gambar *Sequence Diagram* Render Obyek dapat dilihat pada gambar 3.15



Gambar 3.15 *Sequence Diagram* Render Obyek

4. *Sequence Diagram* Mengubah posisi *marker* untuk *GamePlay*

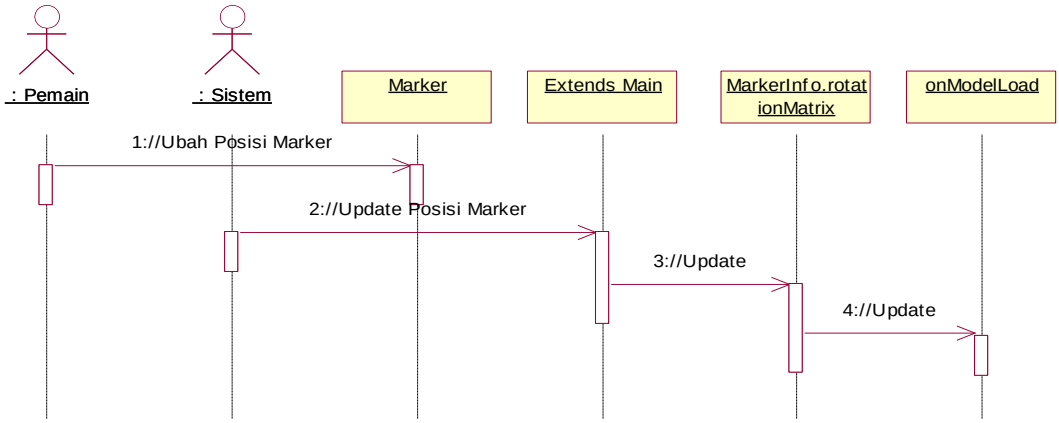
   Gambar *Sequence Diagram* Mengubah posisi *marker* untuk *GamePlay* dapat dilihat pada gambar 3.16.



Gambar 3.16 *Sequence Diagram* Mengubah Posisi *Marker* Untuk *Game* Play

5. *Sequence Diagram Update GamePlay*

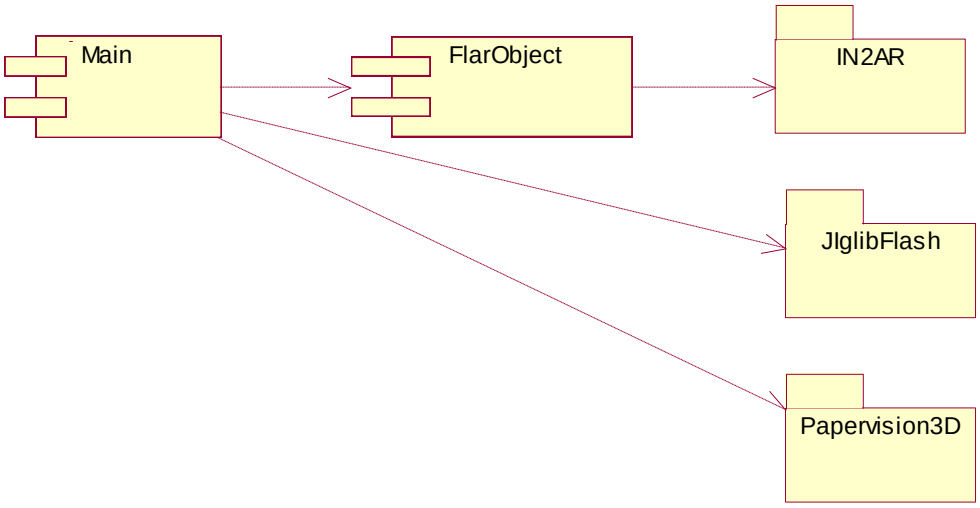
   Gambar *Sequence Diagram Update GamePlay* dapat dilihat pada gambar 3.17.



Gambar 3.17 *Sequence Diagram* Update *Game*Play

3.3.1.3 *Component Diagram*

Component diagram adalah bagian fisik dari sebuah system, karena menetap dikomputer, bukan dibenak para analis. Komponen bisa berupa table, file data, file exe dan lain-lain(dynamic link library), dokumen dan dan lain-lain. *Component* adalah implementasi dari sebuah class. *Component diagram* rancangan aplikasi *game* yang akan dibangun dapat dilihat pada gambar 3.18.

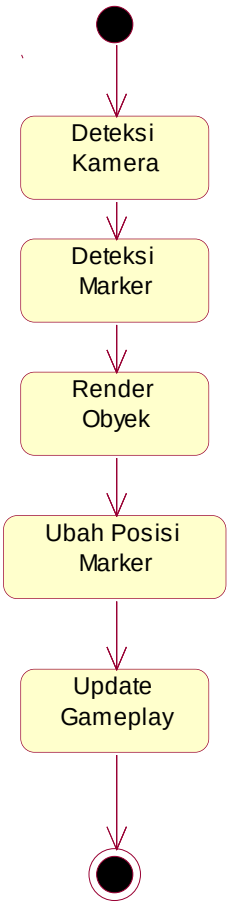


Gambar 3.18 *Component Diagram* Rancangan Aplikasi

Hal terpenting dari *component* adalah *component* memewakili potongan-potongan yang independen yang bisa dipesan dan diperbaharui sewaktu-waktu [12].

3.3.1.4 *State Machine Diagram*

State machine diagram mengekspresikan perilaku sebagai sebuah progresif (perkembangan) melalui sekumpulan *state*, *triggered by events*, dan *action* yang mungkin terjadi. *State machine diagram* juga biasa dikenal sebagai *behavioural state diagram. State machine diagram* biasanya digunakan untuk mendeskripsikan perilaku sebuah *object*. Namun, juga dapat digunakan pada elemen yang besar dari sistem apapun. *State machine diagram* aplikasi *game.* Gambar *State machine diagram* permainan dapat dilihat pada gambar 3.19



Gambar 3.19 *State Machine Diagram Game*

3.4 Perancangan Sistem

Perancangan sistem dibagi menjadi beberapa bagian, yaitu perancangan *augmented reality*, perancangan aplikasi dan perancangan *gameplay*. Pada perancangan *augmented reality* akan dibahas mengenai pokok-pokok pembangun pada permainan yakni cara kerja *augmented reality* ketika diterapkan dalam permainan dan bagaimana interaksi fisik pemain dalam permainan.Dalam perancangan *gameplay* akan dibahas bagaimana bentuk dari konsep permainan

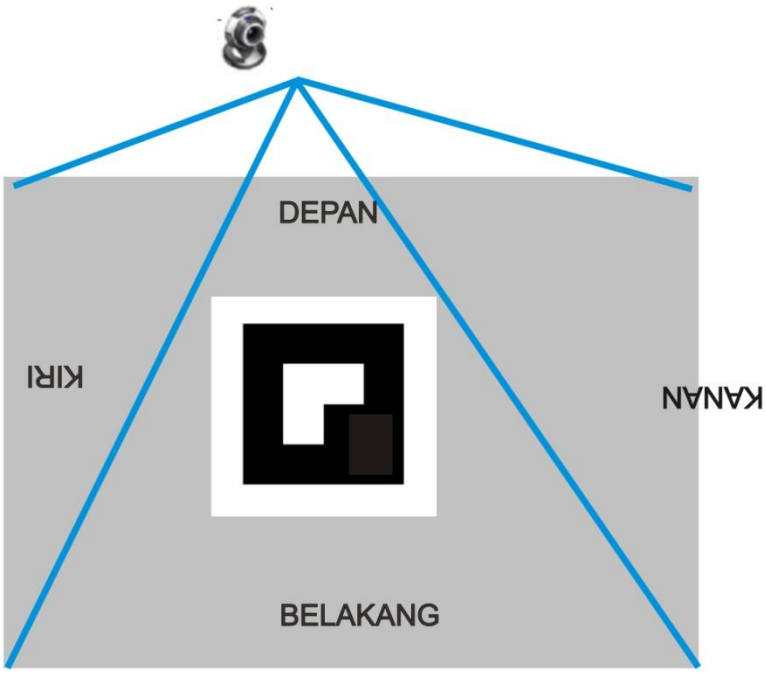
yaitu bagaimana aturan permainan yang akan dijalani ketika memainkan permainan ini.

3.4.1 Perancangan *Marker*

Dalam pembuatan *marker* dapat dilakukan secara sembarang melalui editor grafis atau dapat menggunakan gambar dari obyek tertentu. Gambar yang dihasilkan harus berbentuk *.JPG yang kemudian dimasukan dalam sebuah *.zip yang kemudian dikirimkan melalu halaman IN2AR *support team.*

Setelah pola *marker* berhasil dibuat dan dikirimkan melalui IN2AR *support team* langkah selanjutnya adalah melakukan kalibrasi kamera. Kalibrasi ini hanya dilakukan sekali untuk sebuah perangkat kamera. Untuk desain papan permainan, akan digunakan 1 buah gambar persegi hitam rupiah yang memiliki dimensi 700x306 pixels dengan type JPEG *image* dan size 144 KB. Papan permainan ini nanti akan ditangkap oleh kamera/ *webcam* dan transformasi dari matriks ini akan menjadi acuan dari seluruh obyek virtual seperti pada gambar 3.20.

Untuk dapat memainkan permainan ini, pemain mengunakan gambar *marker* yang diletakkan diatas papan permainan.

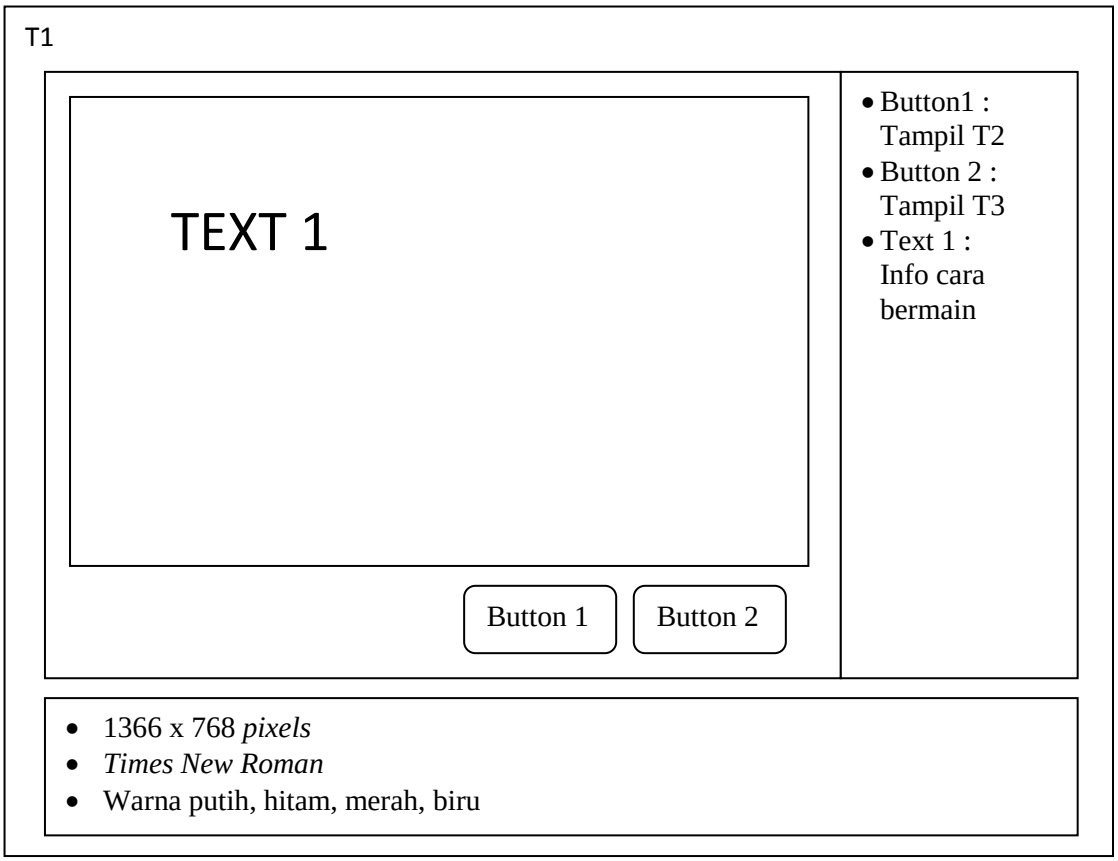


Gambar 3.20 Desain Papan Permainan

3.4.2 Perancangan Aplikasi

3.4.2.1 Perancangan Antarmuka

Desain perancangan antarmuka permaina dapat dilihat pada gambar 3.21



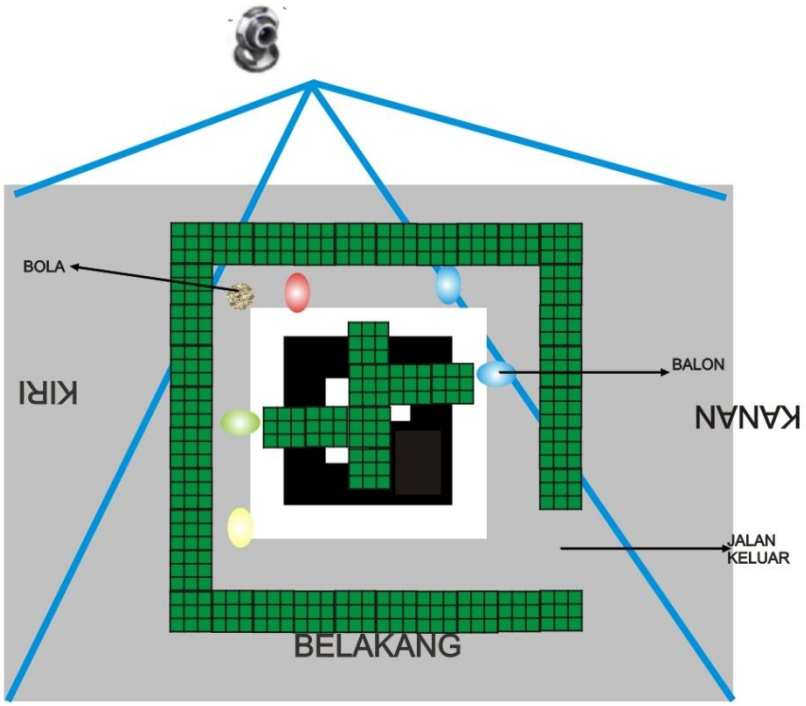
Gambar 3.21 Perancangan Antarmuka Awal

3.4.2.2 Perancangan *GamePlay*

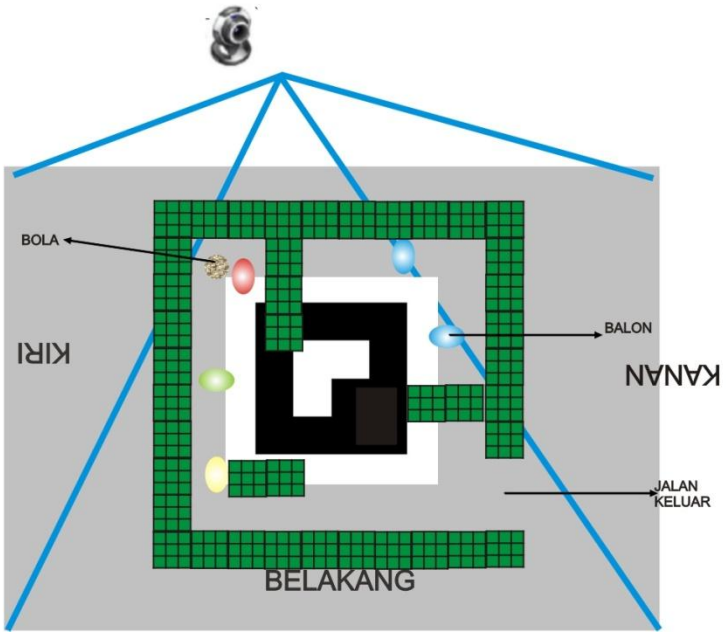
Daerah area permaian berada dalam *marker* yang telah dibuat. Dalam area permainan ini terdapat obyek-obyek bola virtual yang dapat menggelinding, bola virtual yang menjadi target sasaran bola yang dapat menggelinding dan penghalang atau tembok. Permainan ini memiliki misi yaitu, agar bola virtual yang menggelinding dapat mengenai bola-bola virtual diam. Untuk mencapai tujuan itu pemain bertindak sebagai pihak yang mengarahkan bola virtual dengan mencondongkan *marker* ke arah kiri, kanan, depan dan belakang sehingga bola virtual bergerak mendekati bola-bola virtual lain. Ketika bola virtual dapat mencapai bola virtual lain, maka bola virtual lain tersebut akan menghilang.

Area permainan ini berbentuk sebuah labirin dan permainan terdiri dari 5 level permainan, setiap tingkatan level permainan akan menampilkan desain berbeda untuk bentuk labirin. Semakin tinggi tingkat level dari permainan maka desain labirin yang ditampilkan akan semakin rumit untuk dilalui oleh bola virtual. Desain dari masing-masing labirin berdasarkan level permainan dapat dilihat pada gambar 3.22 sampai gambar 3.26.

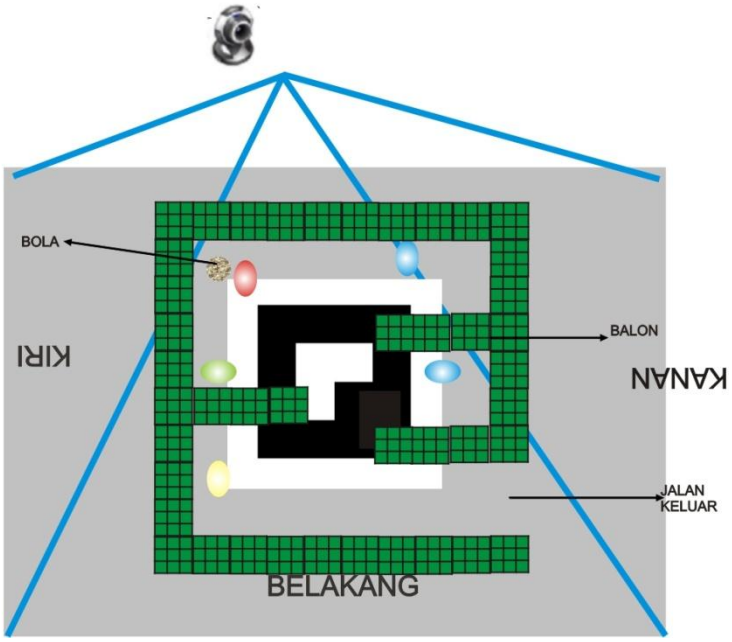
3.4.2.3 Perancangan Cara Bemain



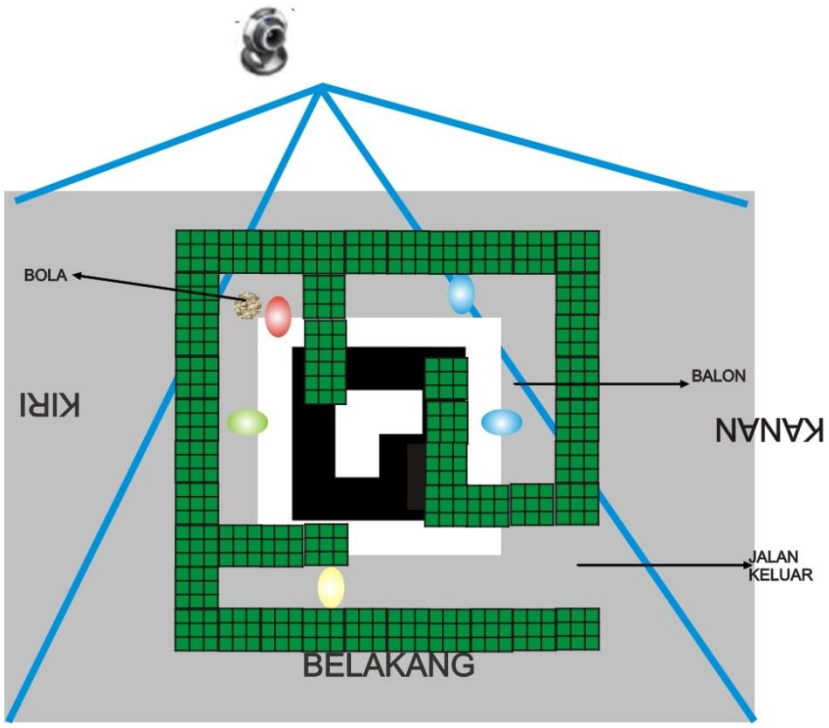
Gambar 3.22 Tampilan Level 1



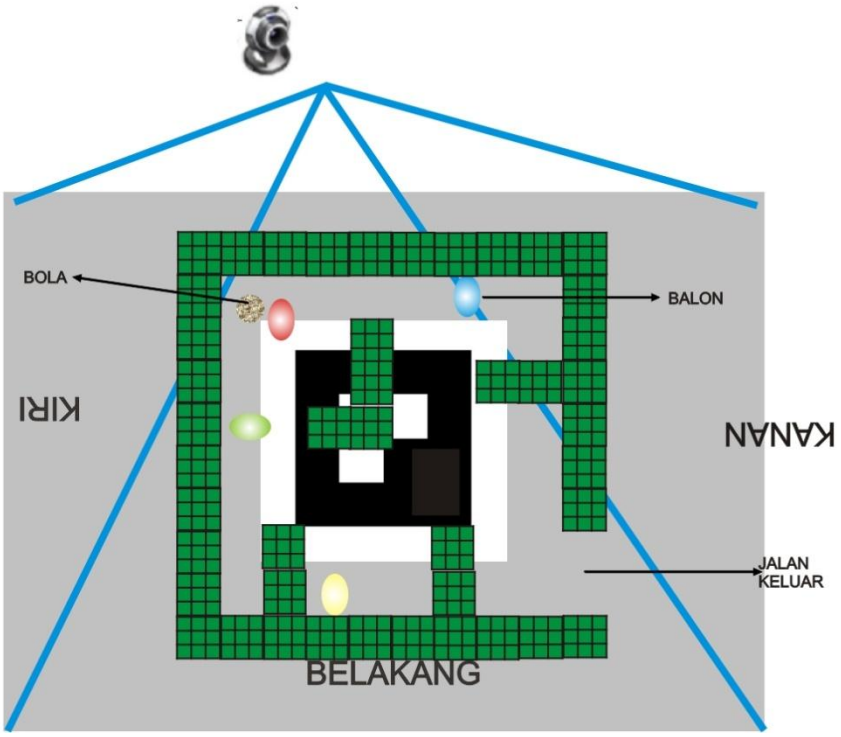
Gambar 3.23 Tampilan Level 2



Gambar 3.24 Tampilan Level 3



Gambar 3.25 Tampilan Level 4



Gambar 3.26 Tampilan Level 5

Cara memainkan permainan ini adalah:

1. Pemain mencondongkan marker kearah kanan, belakang, depan atau kiri sehingga bola menggelinding kearah bola-bola yang terdapat didalam labirin.
2. Untuk dapat melanjutkan ke level selanjutnya pemain harus berhasil menghilangkan seluruh bola yang berjumlah 5 buah, dengan cara ke-1. Kemudian keluar melalui jalan keluar.
3. Jika bola keluar sebelum menyelesaikan misi pada level yang dilalui maka bola akan kembali pada posisi awal.
4. Desain labirin setiap levelnya akan berubah-ubah sesuai dengan level yang sedang dimainkan.
5. Cara 1-3 terus dilakukan dalam memainkan permainan ini.