BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem merupakan suatu kegiatan untuk penguruaian sistem yang sudah berjalan dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengavaluasi permasalahan-permasalahan dan hambatan yang terjadi dan mendokumentasikan kebutuhan yang akan dipenuhi dalam sistem yang baru, sehingga dapat diusulkan perbaikan-perbaikan. Analisis sistem merupakan tahapan yang paling penting, karena kesalahan pada tahap ini akan menyebabkan kesalahan ditahap selanjutnya. Sistem yang dibangun merupakan sarana media edukasi menggunakan teknologi *Augmented Reality* yaitu media pembelajaran pengenalan organ tubuh manusia dan hewan di Sekolah Dasar (SD) khususnya kelas 5. Aplikasi yang dibangun menciptakan suatu persepsi seolah-olah pengguna dapat berinteraksi langsung dengan objek-objek virtual yang telah dibuat. Aplikasi ini dibangun dengan mengambil latar secara *real-time* yang kemudian digabungkan dengan objek 3D melalui kamera *webcam*.

3.1.1 Analisis Masalah

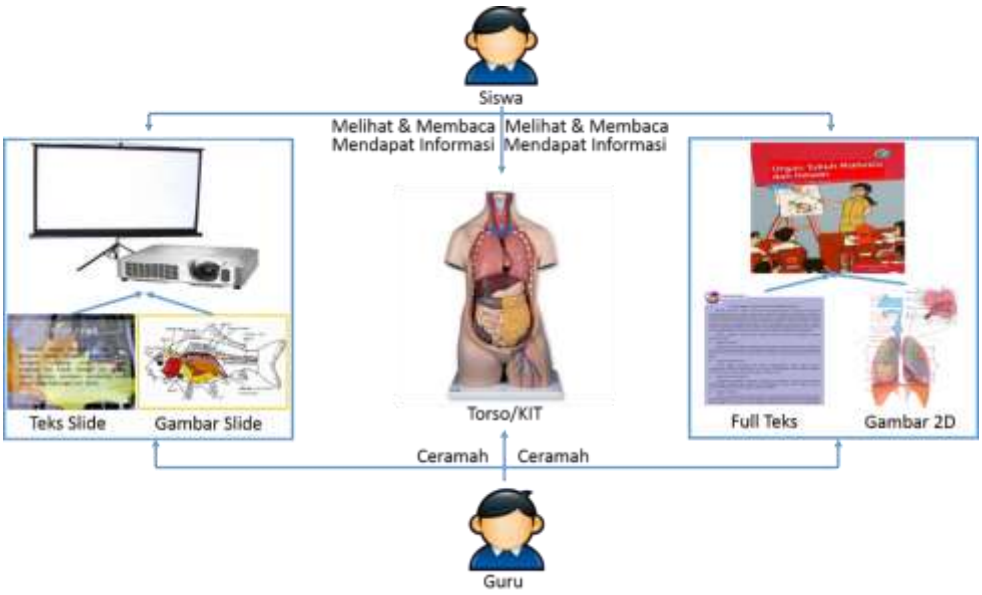
Analisis masalah merupakan suatu gambaran masalah yang diangkat dalam penulisan penelitian ini tentang mengimplementasikan teknologi *Augmented Reality.* Analisis masalah yang dimaksud disini adalah pada sistem pembelajaran tematik terpadu kurikulum 2013 sekolah dasar (SD), metode pembelajaran yang masih konvensional berupa buku bacaan dan ceramah yang selama ini digunakan masih kurang efektif untuk siswa dalam memahami materi yang disampaikan karena daya tangkap dan imajinasi setiap siswa berbeda-beda. Hal ini didukung berdasarkan hasil survey kuesioner penelitian dan wawancara penulis terhadap 20 guru sekolah dasar negeri dilingkungan kota Bandung sebagai responden dengan tanggapan media pembelajaran saat ini kurang memadai 45% serta sarana alat peraga yang tidak lengkap. Proses belajar mengajar yang dikemas kurang menarik dan kurang inovatif disebabkan karena media pembelajaran yang

digunakan kurang efektif seperti menggunakan gambar, ceramah, *slide* dan kit. Hal ini berdasarkan tanggapan responden tentang kesulitan guru dalam menyampaian materi ajar cukup sulit 65%. Media tersebut masih belum mampu menjadi media pengenalan organ tubuh manusia dan hewan yang menarik untuk anak kelas 5 SD.

Dengan teknologi yang berkembang saat ini, khususnya teknologi *Augmented Reality* yang dapat menambah nilai dari penyampaian informasi menjadi lebih tinggi dan mendukung animasi 3D serta suara dapat menjadi sarana untuk menutupi kekurangan yang terdapat pada media pembelajaran saat ini. Dengan merujuk dari kekurangan yang ada maka dibangunlah aplikasi pengenalan organ tubuh manusia dan hewan ini menggunakan teknologi *Augmented Reality*. *Content* dari aplikasi tentang organ tubuh manusia dan hewan ini berdasarkan buku tematik terpadu kurikulum 2013 di Sekolah Dasar. Dengan menggunakan teknologi *Augmented Reality* ini dapat memungkinkan pengguna melihat objek organ tubuh manusia dan hewan secara lebih nyata dengan memproyeksikan objek dalam bentuk tiga dimensi.

3.1.2 Analisis Sistem yang Berjalan

Analisis sistem yang berjalan merupakan tahapan yang memberi gambaran tentang sistem yang sedang berjalan sekarang. Analisa ini bertujuan untuk memberi gambaran yang lebih detail bagaimana cara kerja sistem yang saat ini digunakan. Prosedur pada proses media yang sedang berjalan sekarang dapat dilihat pada gambar 3.1.



Gambar 3.1 Alur Sistem Yang Sedang Berjalan

1. Siswa adalah objek yang ingin mendapatkan informasi yang ada pada buku, *slide* dan torso/kit.
2. Guru adalah objek yang menyampaikan informasi kepada siswa melalui buku, *slide* dan torso/kit.
3. Media buku memberikan informasi dengan menampilkan teks dan gambar dalam bentuk dua dimensi (2D).
4. Media *Slide Show* memberikan informasi dengan menampilkan teks dan gambar dalam bentuk dua dimensi (2D).
5. Media Torso/KIT memberikan informasi dalam bentuk objek organ tubuh.

Dari gambaran tersebut jelas bahwa media yang digunakan untuk mengakses informasi menggunakan buku, *slide* dan torso/kit, media yang selama ini digunakan cukup membuat siswa tertarik pada tahap 75%, namun kekurangan media-media tersebut pada tampilan gambar berbentuk 2D kurang jelas dan media alat torso hanya mengandalkan alat torso organ tubuh manusia serta torso hewan yang jarang terdapat di sekolah. Dimana siswa kesulitan dalam memahami materi tentang organ tubuh manusia dan hewan pada tahap cukup sulit 80% dan guru mengalami kesulitan dalam menyampaikan materi ajar pada tahap cukup sulit 65%.

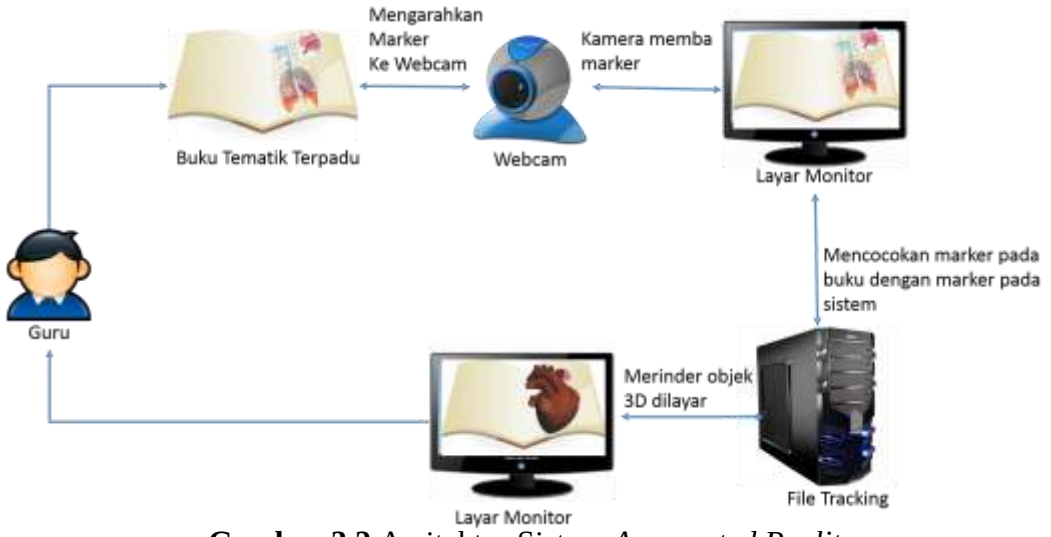
Maka diperlukan media yang dapat menarik kreatiftas siswa berupa gambar animasi 3D, hal ini berdasarkan hasil kuesioner penelitian penulis kepada 20 guru sekolah dasar yang menyatakan sangat setujuh 100% dibutuhkan media inovasi baru yang mampu memudahkan dalam proses belajar mengajar berupa gambar animasi 3D.

Teknologi *Augmented Reality* bisa dijadikan sebagai media alternatif untuk mendapatkan informasi dengan kelebihan:

1. Media lebih fleksibel dan efesien dapat digunakan kapan saja.
2. Tidak membutuhkan biaya besar untuk pengadaan media.
3. Dapat menampilkan objek maya ke dunia nyata sehingga lebih interaktif dan menarik.
4. Dapat berinteraksi langsung dengan objek sacara dinamis.
5. Dapat menampilkan seluruh konten dengan mudah.

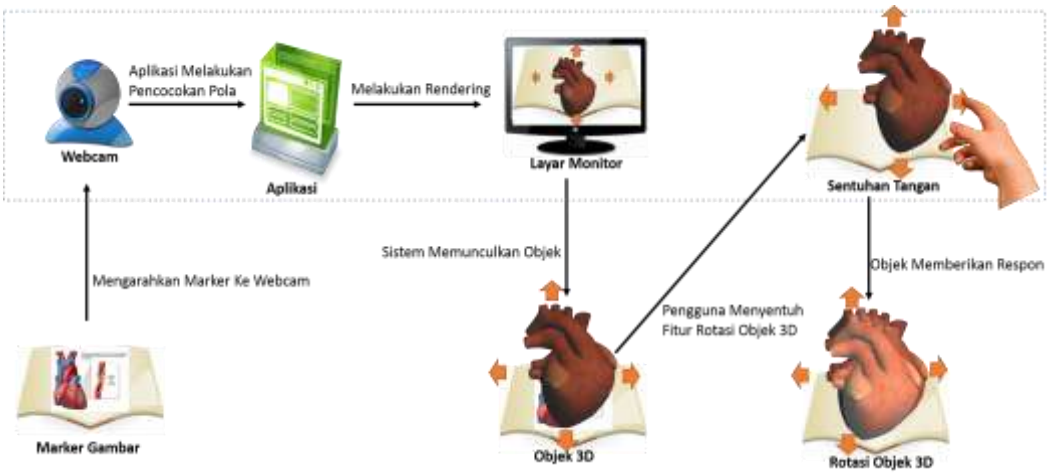
3.1.3 Analisis Arsitektur Sistem

Pada arsitektur sistem yang akan dibangun terdiri dari beberapa komponen yaitu: Guru yang menggunakan sistem pengenalan organ tubuh manusia dan hewan, guru mengarahkan buku yang sudah dilengkapi *marker* gambar (objek AR) ke komputer yang sudah tersedia/terpasang *webcam*, dari *marker* gambar yang di dapat dari *webcam* sistem komputer akan di lakukan pelacakan dan pencocokan *marker* gambar guna mengidentifikasi *marker* gambar yang digunakan guru, jika pola sama seperti yang terdapat pada sistem maka komputer me-*render* objek-objek 3D yang digunakan dalam sistem. Guru dapat melihat hasil melalui layar komputer dan proyektor. Gambaran arsitektur sistem dapat dilihat pada gambar 3.2.



Gambar 3.2 Arsitektur Sistem *Augmented Reality*

Sistem yang akan dibangun adalah suatu aplikasi berbasis *desktop* dengan menggunakan teknologi *Augmented Reality*. Keunggulan dari teknologi *Augmented Reality* sangat menonjolkan dari segi tampilan yang lebih nyata dan dijalankan secara *real-time* dan didukung dengan *motion detection* dalam berinteraksi dengan objek virtual berupa gerakan tangan yang seolah-olah menyentuh fitur objek yang muncul pada layar. Sehingga dapat diterapkan dalam aplikasi agar dapat menambah nilai dari suatu informasi yang dibutuhkan dan juga dapat menjadi solusi dari permasalahan yang ada. Proses aplikasi yang akan dibangun dengan menggunakan teknologi *Augmented Reality* dapat dilihat pada gambar 3.3 yang menggambarkan alur sistem yang akan dirancang.



Gambar 3.3 Sistem Yang Akan Dibangun

3.1.4 Analisis Kebutuhan Tool/Library Augmented Reality

Tools dan *library* untuk *Augmented Reality* telah banyak dikembangkan. Perbandingan beberapa *tools* dan *library* untuk *Augmented Reality* dapat dilihat pada Tabel 3.1.

Tabel 3.1 Perbandingan *Tools* dan *Library* AR

No	Parameter	Tools/Library			
		ARToolkit	**Flartoolkit**	**D'Fusion**	**IN2AR**
1	Lisensi	Gratis	Gratis	Gratis	Gratis
2	Dukungan bahasa pemrograman	C	ActionScript	LUA scripting	ActionScript
3	Dukungan marker	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	sccnctta	.dae, obj, .3ds
6	Dukungan platform	Multiplatform	Multiplatform	Multiplatform	Multiplatform
7	Distribusi kepada pengguna	Harus instal ARToolkit dikomputer pengguna	.swf	Menggunakan D'Fusion @HomeDesktopPlayer	.swf

Kelebihan *IN2AR* jika dibandingkan dengan *toolkit* atau *library* lain adalah:

1. Selain menyediakan lisensi berbayar (2212,50 EURO per proyek), juga terdapat lisensi gratis. Perbedaan kedua lisensi tersebut adalah bahwa aplikasi AR yang dibuat dengan lisensi gratis harus selalu menyertakan logo *IN2AR* di dalam aplikasi.
2. Dukungan pengembang di forum *IN2AR* sangat kuat. Pengembang aktif menjawab pertanyaan seputar teknis *IN2AR*.
3. *IN2AR* dapat mengenali hampir semua jenis gambar sebagai *marker*. Berbeda dengan *ARToolkit* dan *FLARToolkit* yang hanya mengenali gambar dengan bingkai hitam sebagai *marker*.
4. Distribusi aplikasi yang dibuat dengan *IN2AR* cukup mudah.

Aplikasi didistribusikan menggunakan file .swf yang dapat ditampilkan di halaman *web*. *User* yang memiliki *web browser* dengan *flash player* dan komputer *webcam* serta terkoneksi internet akan dapat mengakses aplikasi AR tersebut.

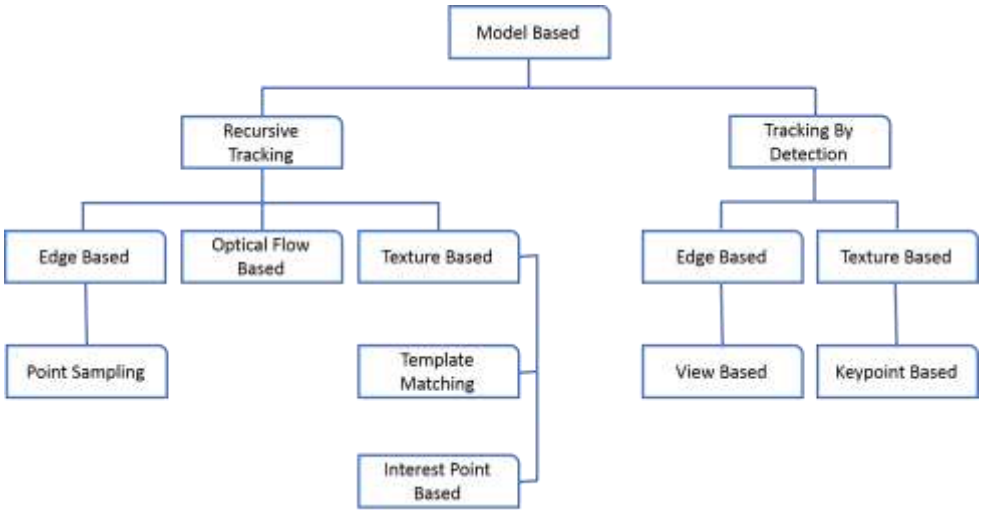
Selain memiliki beberapa kelebihan, *IN2AR* juga memiliki kelemahan yaitu:

1. Pengguna harus mengenal bahasa pemrograman, terutama bahasa pemrograman *Action Script 3*.0. Hal ini karena segala macam perubahan dalam penggunaan *IN2AR* adalah dengan cara mengetikkan kode dalam bahasa *pemrograman Action Script 3.0*. *IN2AR* tidak memiliki dokumentasi yang lengkap dan sistematis. Pengguna *IN2AR* harus melakukan eksplorasi sendiri untuk dapat menggunakan *IN2AR* sesuai kebutuhan. Aktivitas di forum *IN2AR* akan sangat membantu jika pengguna mengalami kesulitan.
2. Pembuatan *marker* untuk lisensi gratis harus melalui tim, Gambar yang ke tim *IN2AR* gambar yang akan dijadikan *marker* harus dikirim terlebih dahulu ke tim *IN2AR* untuk dibatkan file .ass.

3.1.5 Analisis *Markerless Augmented Reality Tracking System*

Sifat pelacakan yang digunakan dengan teknik berbasis model *(model based techniques)* dapat diklasifikasikan dalam 2 bagian seperti pada Gambar 3.5.

1. Pelacakan rekursif *(Recursive tracking*), dimana pada saat sebelum objek dimunculkan terlebih dahulu dilakukan perkiraan untuk menghitung pelacakan yang terjadi.
2. Pelacakan dengan pendeteksian *(Tracking by detection)*, pada klasifikasi ini dilakukan perhitungan pelacakan tanpa perkiraan sebelumnya, yang memungkinkan otomatis inisialisasi dan pemulihan dari kegagalan.



Gambar 3.4 *Diagram Teknik Online Monocular*

Pelacakan rekursif menjadi teknik model yang digunakan oleh pihak *IN2AR* untuk melakukan teknik *markerless tracking system*. Dalam penelitian ini menggunakan teknik *markerless* karena dalam proses *tracking marker*, dimana proses *tracking* ini menggunakan tekstur gambar yang disimpan dalam sistem sebagai sumber referensinya dan membandingkan tekstur yang tertangkap oleh kamera dengan tekstur gambar yang ada pada sistem.

Tekstur gambar yang digunakan untuk *tracking* beradasarkan gambar yang terdapat pada buku Tematik Terpadu Kurikulum 2013 yang membahas mengenai lingkup organ tubuh manusia dan hewan.

3.1.6 Analisis Metode Terhadap Kasus

Analisis metode terhadap kasus merupakan analisis yang mendeskripsikan bagaimana proses *Augmented Reality IN2AR* dari awal Inisialisasi, *Tracking marker*, sampai dengan proses *Rendering objek 3D* dengan metode *Markerless Tracking* yang diterapkan oleh *IN2AR*. Dalam perancangan aplikasi dengan teknologi *Augmented Reality*, seolah-olah menggabungkan objek virtual dengan objek nyata, dalam hal ini objek virtual berupa objek 2D atau 3D dan objek nyatanya berupa gambar dengan pola tertentu (*markerless*). Sistem *Augmented Reality Display* yang digunakan adalah teknik *spatial display* dengan *screen display* (bisa menggunakan *monitor* ataupun *proyektor*).



Gambar 3.5 Alur *Tracking Augmented Reality*

Secara keseluruhan, proses sistem *Augmented Reality* dapat digambarkan dengan diagram aliran seperti pada gambar 3.5. Aplikasi melakukan inisialisasi terlebih dahulu sebelum melakukan *tracking marker*, *marker* dideteksi dari masukan video *webcam*, jika *marker* terdeteksi maka objek 3D di *render*. Secara garis besarnya, dalam perancangan ada tiga bagian utama yaitu sebagai berikut :

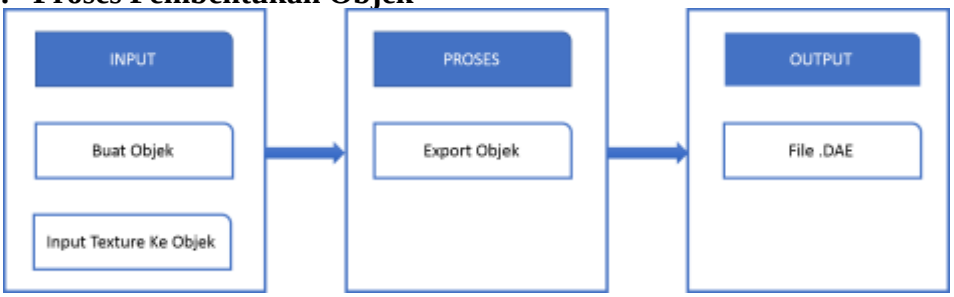
3.1.6.1 Inisialisasi

Pada tahap ini ditentukan *marker* yang akan digunakan, sumber *input* video nya, dan objek 3D yang akan digunakan .Pada bagian inisialisasi ini, objek 3D diinisialisasi terlebih dahulu karena *loading* objek 3D memerlukan waktu yang cukup lama.

3.1.6.1.1 Inisialisasi Pembentukan Model 3D

Model 3D yang akan ditampilkan di *load* terlebih dahulu. Agar aplikasi dapat menampilkan objek 3D tertentu tanpa merubah atau membangun ulang aplikasi, diperlukan sebuah *file* konfigurasi untuk menentukan objek 3D yang akan di *load* sesuai dengan pola *marker* yang dideteksi.

1. **Proses Pembentukan Objek**



Gambar 3.6 Proses Pembentukan Data Objek 3D



Gambar 3.7 Proses *Export* Objek

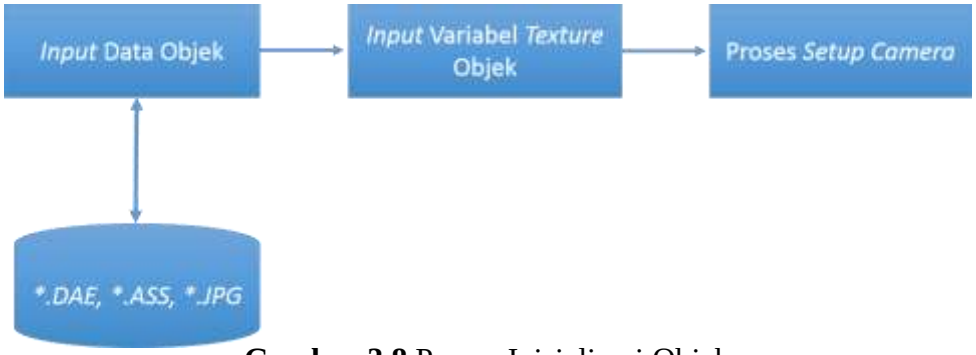
Dalam proses pemodelan objek terdiri dari 3 langkah yaitu:

1. Menyesuaikan objek 3D dengan animasi atau bentuk yang akan dibuat.
2. Memasukan *teksture* sesuai dengan objek 3D.
3. Meng*export* objek yang sudah dirancang dan dibuat kedalam format *collada* (*.DAE).

Pada tahapan ketiga yaitu meng*export* objek kedalam format *collada* (*.DAE), tidak bisa dilakukan secara manual dengan menggunakan *export* bawaan dari aplikasi 3D pembuat objek. *Export* objek harus terlebih dahulu meng-install applikasi *OpenCOLLADA* agar objek dapat ditampilkan dan sesuai dengan yang dibuat.

3.1.6.1.2 Inisialisasi Objek

Pada tahap ini ditentukan proses dimana *IN2AR SDK* bekerja untuk mengambil gambar sebagai *input*-an yang akan diproses. Inisialisasi merupakan tahap awal yang digunakan sebagai media dalam mengangkap gambar dari *webcam*.



Gambar 3.8 Proses Inisialisasi Objek

1. *Input* Data Objek

Proses *input* data objek merupakan proses dimana seluruh komponen objek yang dijadikan sebagai data masukan dan keluaran sistem akan di *embed* pada suatu *class* agar dapat di insialisasi oleh sistem. Proses *tersebut* menambahkan data objek 3D yang berekstensi *.DAE* di dalam suatu kode *script* pada sebuah *class ModelDAE*, menambahkan data *file tracking marker* yang telah di konversi menjadi *.ASS* di dalam suatu kode *script* pada sebuah *class MarkerASS* dan menambahkan objek *teksture* gambar yang menjadi objek *tracking* berekstensi *.JPG* di dalam suatu kode *script* pada sebuah *class TekstureJPG* seperti pada *pseudocode* berikut:

```
[Embed(source <- "../ass/organ.ASS", mimeType <-
"application/octet-stream")]
kamus: MarkerASS:Class
[Embed(source <- "../dae/organ.DAE", mimeType <-
"application/octet-stream")]
kamus:  ModelDAE:Class
[Embed(source <-
"../dae/images/organ_rotation_rotation.JPG")]
kamus:  TeksturJPG:Class
```

Pada *pseudocode* tersebut semua *file* objek yang dijadikan *file tracking* di *embed*, objek 3D model dan *marker* teksture gambar di *embed* sesuai dengan lokasi file data objek, dimana data *marker* .ASS disimpan dalam sebuah folder bernama *'ass'* dan data objek 3D model disimpan pada sebuah folder bernama *'dae'* sedangkan data objek teksture gambar disimpan dalam sebuah folder bernama *'dae'* dan didalam sebuah folder bernama *'images'* yang masing-masing *file* di tampung oleh *class* masing-masing.

Hal ini bertujuan agar sistem dapat mengenali semua *file tracking* pada sistem dan membandingkan pola *file* tersebut dengan hasil tangkapan kamera *realtime* saat melakukan *tracking* terhadapap *marker*.

Tabel 3.2 *File* Data Objek

FILE	**FUNGSI**
.DAE	Berfungsi untuk memanggil bentuk objek 3D yang telah di-*export*.
.ASS	Berfungsi untuk memanggil bentuk *marker* yang telah di konversi oleh tim *IN2AR*.
.JPG	Berfungsi untuk memanggil bentuk *teksture* gambar yang dijadikan *marker tracking*.

2. *Input* Variabel Teksur Objek

Proses ini berfungsi untuk menambahkan tekstur pada objek, tekstur pada objek tidak bisa digunakan atau muncul sebelum ada penambahan variabel dan pengaturan kode yang ditambahkan pada *function init3Dmodel* agar tekstur tersebut sesuai dengan model yang telah dibuat.

1. Penambahan Variabel *Teksture*

   Penambahan variabel teksture disesuaikan dengan jumlah dan nama material *id* pada *file .DAE* yang digunakan pada objek.

2. Penambahan Kode C*lass* Variabel *Teksture*

   Penambahan kode *class* pada gambar atau tekstur yang ditambah disesuaikan dengan variabel *teksture*.

```
kamus:
bitmapTexture:Bitmap
bitmapMaterialTexture:BitmapMaterial
algoritma:
bitmapTexture <- new TeksturJPG_organtubuhmanusia()as
Bitmap
bitmapMaterialTexture <- new
BitmapMaterial(bitmapTexture.bitmapData, true)
model.materials.addMaterial(bitmapMaterialTexture,
"organtubuhmanusia")
```

Pada *pseudocode* tersebut tekstur perlu *assigned* secara manual pada *material id* yang sesuai dengan *file dae embedded*. Karena model 3D tersebut

tidak diambil dari *external file* ketika aplikasi sedang berjalan, maka teksturnya tidak secara otomatis *assigned* dan harus pasangkan dengan *material id* yang sesuai secara manual.

```
<material id<-"material001" name="organtubuhmanusia">
```

Nama dari *material id* (pada *script* tersebut adalah "organtubuhmanusia") dapat diketahui dengan membuka *file* objek 3D model *.dae* yang sebelumnya telah di inisialisasi yang terdapat pada folder '*dae'*. Proses pemasangan nama *material id* ini dilakukan dengan meng-*click* objek 3D dan akan ditampikan *source code* dan perubahan *material id* dilakukan pada *<library material> <material id>*.

3. Proses *Setup Camera*

Proses ini dilakukan pada saat objek di *compile* menjadi aplikasi dan data objek beserta tekstur objek akan di inisialisasi. Proses pertama yang akan dilakukan dengan mengaktifkan *webcam* dan menampilkan halaman video *realtime* dengan pengaturan kode pada *function initWebcam* agar kamera dapat menginisialiasai objek *tracking*.

```
kamus:
camera:Camera
algoritma:
bitmapData <- new BitmapData(WIDTH, HEIGHT)
camera <- Camera.getCamera()
camera.setMode(WIDTH, HEIGHT, FPS)
video <- new Video(WIDTH, HEIGHT)
video.attachCamera(camera)
```

Proses pengaktifkan kamera dengan cara memanggil variabel kamera dengan *Camera.getCamera()*, setelah camera dipanggil maka secara otomatis kamera akan mengatur *width, height* dan *fps* lalu menampilkan kamera video secara *realtime* untuk melakukan *tracking*.

Pada proses penyesuaian kamera diperlukan integritas *camera3D* antara *IN2AR* dan *Papervision3D*, karena menggunakan *library papervision3D* sebagai *engine* untuk model objek 3D yang akan ditampilkan dengan mengatur kode pada *function initPapervision3D.*

```
scene3D <- new Scene3D()
viewport3D <- new Viewport3D(WIDTH, HEIGHT)
addChild(viewport3D)

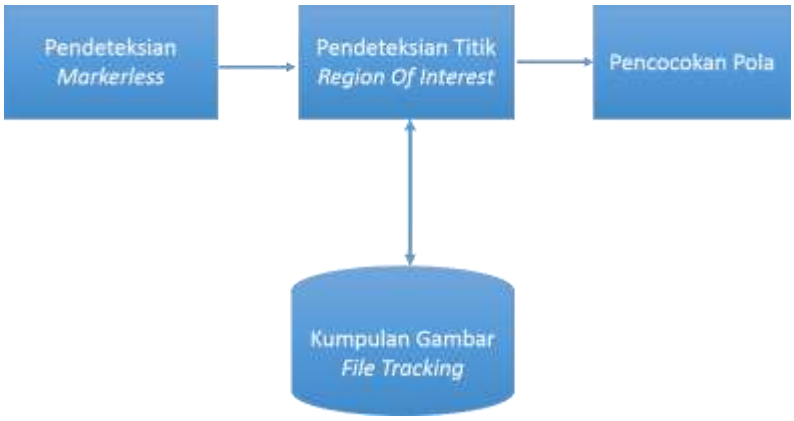
kamus:
params:IntrinsicParameters
algoritma:
params <- asfeat.lib.getIntrinsicParams()
camera3D <- new ARPV3DCamera(params, WIDTH, HEIGHT)
renderEngine <- new LazyRenderEngine(scene3D, camera3D,
viewport3D)

pointLight3D <- new PointLight3D()
pointLight3D.x <- 1000
pointLight3D.y <- 1000
pointLight3D.z <- -1000
```

Pada *pseudocode* tersebut terdapat variabel *params:IntrinsicParameters = asfeat.lib.getInterinsicParams()* hal ini diperlukan untuk menyeseuaikan antara camera 3D pada *IN2AR* dan pada *Papervision3D*, pengaturan camera3D *papervision* dengan cara memanggil *class ARPV3Dcamera.as* pada folder bernama *'src'* lalu folder bernama *'arsupport'* dan mengatur 3 parameter untuk objek 3D pada sumbu X,Y dan Z yang ditampung pada *PointLight3D*.

3.1.6.2 Pelacakan Dan Pencocokan *Marker*

Pada tahap ini ditentukan proses dimana *IN2AR SDK* bekerja untuk pelacakan dan pencocokan pola *marker* sebagai *file tracking*. Dengan proses seperti pada gambar 3.9.



Gambar 3.9 Proses Pelacakan dan Pencocokan *Marker*

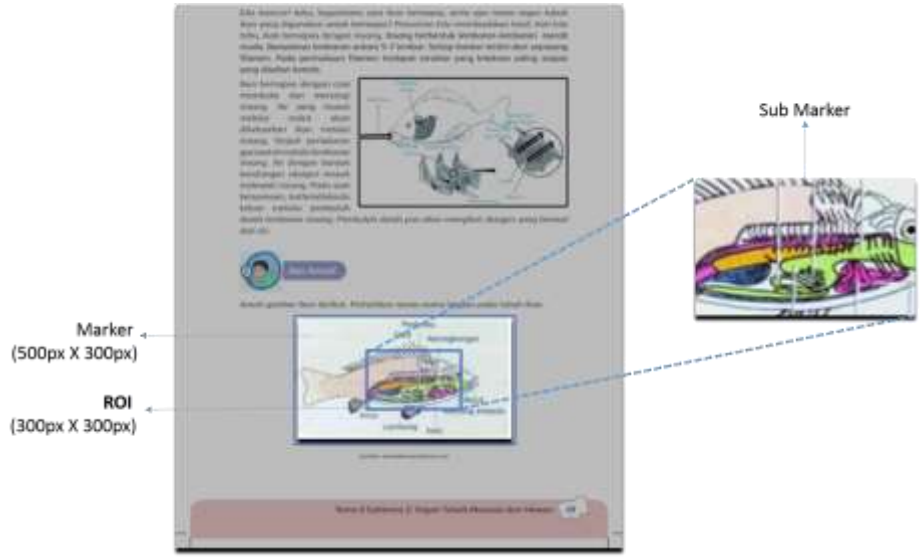
IN2AR memiliki kemampuan untuk mendeteksi gambar dan menghitung posisi gambar tersebut menggunakan *webcam* standar. Informasi posisi yang didapatkan akan dipergunakan untuk menempatkan objek atau model 3D atau animasi ke dalam posisi gambar atau *marker*. Proses yang dilakukan yaitu mengatur berbagai parameter dari *IN2AR*, diantaranya jumlah titik *region of interest* pada setiap *frame* gambar dari *webcam* yang dapat dideteksi serta jumlah *marker* yang digunakan, pengaturan kode dilakukan pada *function initIN2AR*.

```
Kamus:
maxPoints:int
maxMarkers:int
maxTransformError:Number
algoritma:
maxPoints <- 300
maxMarkers <- 1
maxTransformError <- 10 * 10

asfeat.lib.init(WIDTH, HEIGHT, maxPoints, maxMarkers,
maxTransformError, stage)
asfeat.lib.setupIndexing(12, 10, true)
asfeat.lib.setUseLSHDictionary(true)
```

```
asfeat.lib.addReferenceObject(ByteArray(new
MarkerASS_organtubuhmanusia))
asfeat.lib.addListener(ASFEATDetectionEvent.DETECTED,
onMarkerDetected)
asfeat.lib.setMaxReferencesPerFrame(1)
```

Seperti pada potongan *pseudocode* tersebut *maxPoints* yang digunakan sebesar 300, pemberian nilai *maxPoints* tersebut merupakan nilai standar berdasarkan ketentuan IN2AR setiap *marker* yang telah di *generate* memiliki minimum 550 x 100 *pixel*, dengan begitu sistem akan mampu mendeteksi titik *region of interest* sebesar 300 dari setiap *frame marker* gambar *tracking* yang nantinya akan dilakukan pencocokan. Area *region of interest* dapat dilihat pada gambar 3.10. Pada *maxMarkers* yang digunakan berjumlah 1, maka sistem akan mengenali maksimal 1 buah *marker* dan posisi transformasi *error marker* tersebut sebesar 10x10. *Marker* yang di kenali oleh sistem berupa *marker* yang sudah di referensikan berdasarkan *addReferenceObject* serta ketentuan maksimal penggunaan referensi *marker*-nya di atur pada *setMaxReferencesPerFrame.*



Gambar 3.10 Contoh Area *Region Of Interest*

Jika terdapat *marker* yang dapat terdeteksi melalui video *webcam* secara *realtime*, maka model 3D akan ditampilkan diatas *marker* tersebut berdasarkan matriks transformasi dari *marker IN2AR* yang telah di petakan ke matriks transformasi *Papervision3D*. Pengkodean pendeteksian *marker* yang ditemukan diatur pada *funcution onMarkerDetected.*

```
Kamus:
markerInfoList:Vector.<ASFEATReference>
markerInfo:ASFEATReference
n:int
i:int
rot:Vector.<Number>
trans:Vector.<Number>
algoritma:
markerInfoList <- e.detectedReferences
n <- markerInfoList.length

for (i <- 0, i < n, i++)
   markerInfo <- markerInfoList[i]
   rot <- markerInfo.rotationMatrix
   trans <- markerInfo.translationVector
   PV3D_MATRIX:Matrix3D <- new Matrix3D()
      if (mirror <- false)
            PV3D_MATRIX.n11 <- rot[1]
            PV3D_MATRIX.n12 <- rot[0]
            PV3D_MATRIX.n13 <- rot[2]
            PV3D_MATRIX.n14 <- trans[0]
            PV3D_MATRIX.n21 <- rot[4]
            PV3D_MATRIX.n22 <- rot[3]
            PV3D_MATRIX.n23 <- rot[5]
            PV3D_MATRIX.n24 <- trans[1]
            PV3D_MATRIX.n31 <- rot[7]
```

```
            PV3D_MATRIX.n32 <- rot[6]
            PV3D_MATRIX.n33 <- rot[8]
            PV3D_MATRIX.n34 <- trans[2]
      else
            PV3D_MATRIX.n11 <- rot[1]
            PV3D_MATRIX.n12 <- -rot[0]
            PV3D_MATRIX.n13 <- -rot[2]
            PV3D_MATRIX.n14 <- -trans[0]
            PV3D_MATRIX.n21 <- -rot[4]
            PV3D_MATRIX.n22 <- rot[3]
            PV3D_MATRIX.n23 <- rot[5]
            PV3D_MATRIX.n24 <- trans[1]
            PV3D_MATRIX.n31 <- -rot[7]
            PV3D_MATRIX.n32 <- rot[6]
            PV3D_MATRIX.n33 <- rot[8]
            PV3D_MATRIX.n34 <- trans[2]
     endif
    if (arrayModelContainer[markerInfo.id].visible <-
    false)
    arrayModelContainer[markerInfo.id].visible <- true
    arrayModelContainer[markerInfo.id].transform <-
    PV3D_MATRIX
        markerFound <- true
        renderEngine.render()
    endif
endfor
```

Pada *pseudocode* transformasi matriks *IN2AR* ke *Papervision* tersebut ditampung pada suatu variabel *markerInfoList: Vector. <ASFEATReference> = e.detectedReferences,* transformasi matriks yang digunakan berupa transformasi matriks rotasi menggunakan variabel

`rot:Vector.<Number> = markerInfo.rotationMatrix` dan transformasi matriks translasi menggunakan variabel `trans:Vector.<Number>=markerInfo.translationVector.`

Jika *`markerFound`* sama dengan *`true`* maka objek akan di-*render* berdasarkan hasil transformasi matriks dengan menampilkan objek 3D ke layar komputer.

Proses pendeteksian ada atau tidaknya *marker* dilakukan pada setiap *frame* gambar yang diperoleh dari *webcam*, jika *`markerFound`* bernilai *`false`* maka akan memanggil *function onMarkerLost* dimana jika *marker* tidak dapat terdeteksi maka sistem akan menghilangkan objek 3D tersebut, pengaturan terhadap pendeteksian *marker* dengan mengatur pada *function onEnterFrame* dan *function onMarkerLost.*

```
markerFound <- false
bitmapData.draw(video)
asfeat.lib.detect(bitmapData)
     if (!markerFound)
          onMarkerLost(0)
     endif
```

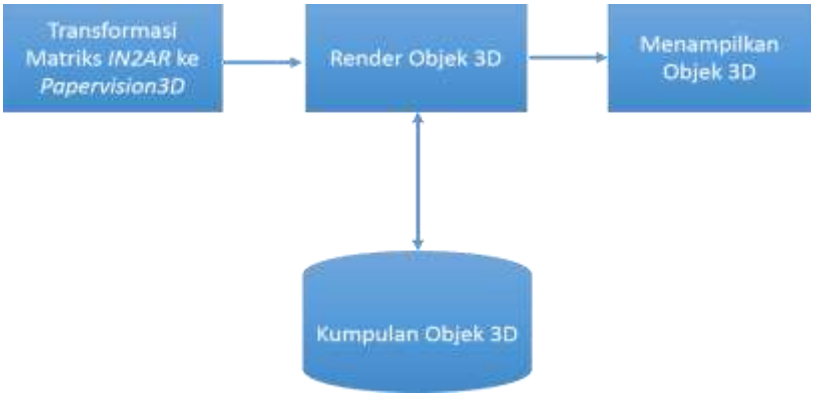
```
arrayModelContainer[i].visible <- false
renderEngine.render()
```

Pada proses pencocokan pola, mula-mula sistem mendeteksi *marker* melalui perspektif kamera *realtime*, jika *marker* terdeteksi kamera maka sistem akan mengecek pola *marker* tersebut dengan membandingkannya dengan *teksture* gambar *marker* yang sudah di *embed*. jika pola *marker* sesuai dengan yang terdapat pada data *file tracking* maka sistem akan memanggil objek 3D berdasarkan *class ModelDAE* dan mengecek kesesuaian *material id* nama pada struktur objek **.DAE* dan melakukan *rendering* dengan memanggil *function onModelLoaded.*

```
kamus:
bitmapTexture:Bitmap
bitmapMaterialTexture:BitmapMaterial
algoritma:
bitmapTexture <- new TeksturJPG_ rangkatubuhmanusia ()
as Bitmap
bitmapMaterialTexture <- new
BitmapMaterial(bitmapTexture.bitmapData, true)
model.materials.addMaterial(bitmapMaterialTexture,
"rangkatubuhmanusia")
model <- new DAE(true, "model", true)
model.addEventListener(FileLoadEvent.LOAD_COMPLETE,
onModelLoaded)
model.load(new ModelDAE_rangkatubuhmanusia() as
ByteArray)
```

3.1.6.3 *Rendering* Objek 3D

Pada tahap ini ditentukan proses dimana *IN2AR SDK* bekerja untuk me-*render* objek 3D berdasarkan pendeteksian dan pencocokan pola *marker* sebelumnya. Dengan proses seperti pada gambar 3.11.



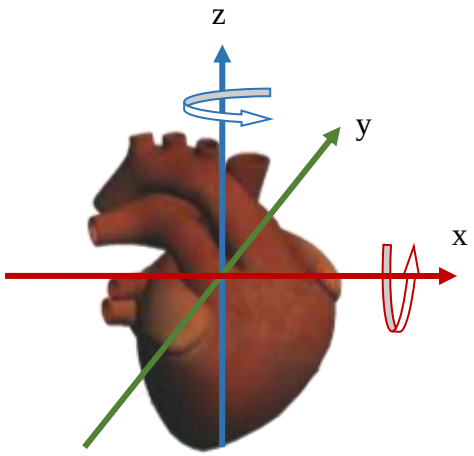
Gambar 3.11 Proses *Rendering* Objek 3D

Proses *rendering* merupakan tahapan akhir menempatkan objek 3D pada lingkungan dunia nyata yang ditangkap melalui kamera, setelah pola *marker*

dikenali dan cocok dengan pola yang telah di *embed* pada *file tracking*, berdasarkan kesesuaian antara *bitmapTeksture* dan *bitmapData* bernilai *true* dan berdasarkan kesesuaian terhadap nama *material id* objek 3D *.DAE* dengan memanggil sebuah *function onModelLoaded*. Sistem akan menampilkan objek 3D dan memposisikan objek 3D tersebut berada ditengah *marker* .

```
model.removeEventListener(FileLoadEvent.LOAD_COMPLETE,
onModelLoaded)
model.rotationX <- -90
model.rotationY <- 0
model.rotationZ <- 90
model.scale <- 1.5
```

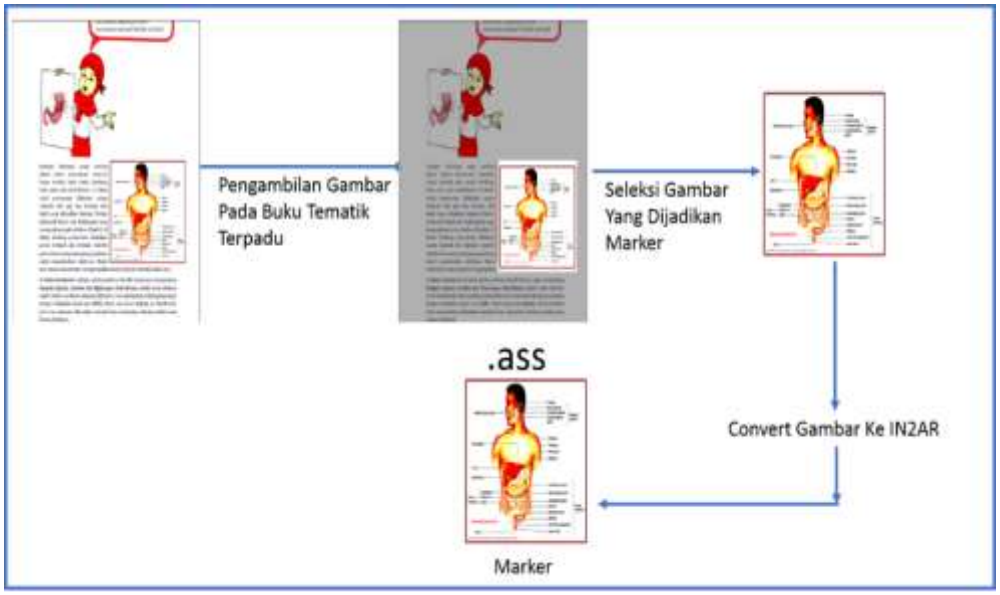
Model yang telah di-*render* akan diatur rotasi dan skalanya, dimana rotasi sumbu X sebesar -90, rotasi sumbu Y sebasar 0, rotasi Z sebesar 90 dan dengan skala 1.5.



Gambar 3.12 Rotasi Objek 3D

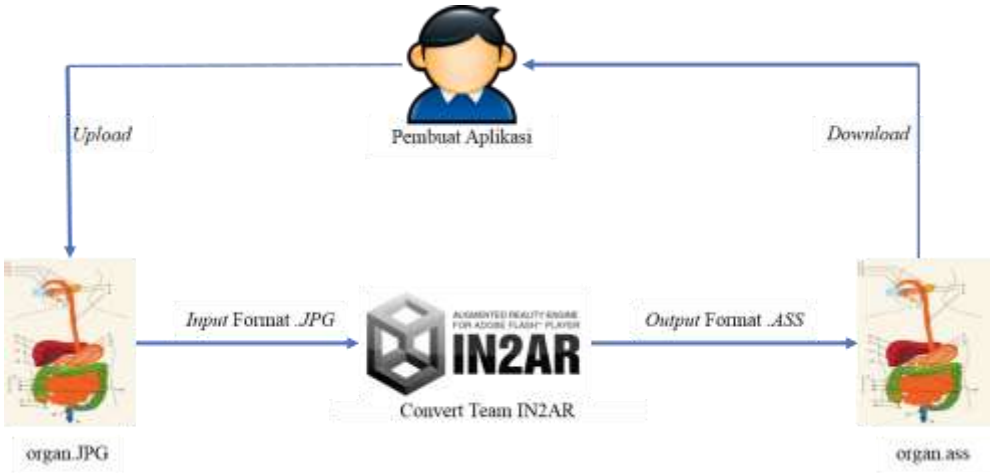
3.1.6.3.1 *Marker* Yang Digunakan

Marker yang digunakan pada analisis ini didapatkan dari hasil gambar pada buku tematik terpadu kurikulum 2013 yang sudah ada. Adapun proses pembentukan *marker* dapat dilihat pada gambar 2.13.



Gambar 2.13 Pembentukan *Marker*

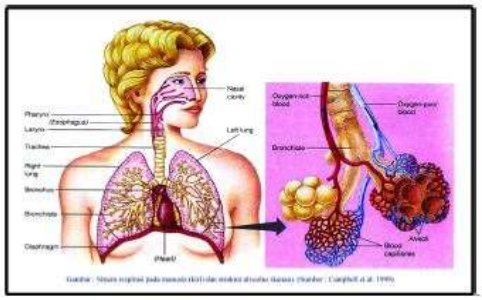
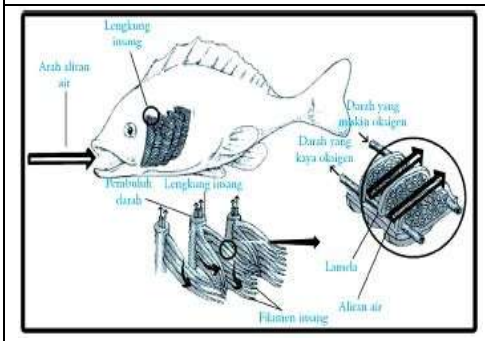
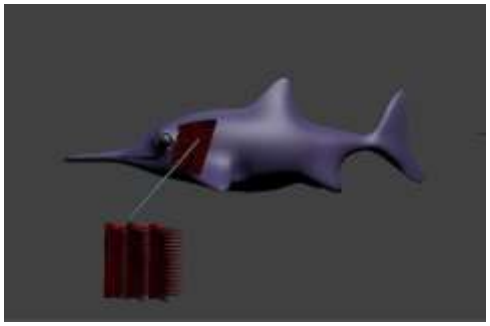
Proses pembuatan gambar *marker* menjadi *marker* dilakukan oleh pihak *IN2AR* dengan cara mengirimkan gambar yang akan di-*convert*, melalui *marker engine* yang disediakan oleh pihak *IN2AR*. Setelah gambar di *convert* menghasilkan *file* dengan format *.ass*. File tersebut kemudian dijadikan masukan pada *coding* untuk mendeteksi gambar yang dijadikan *marker*.

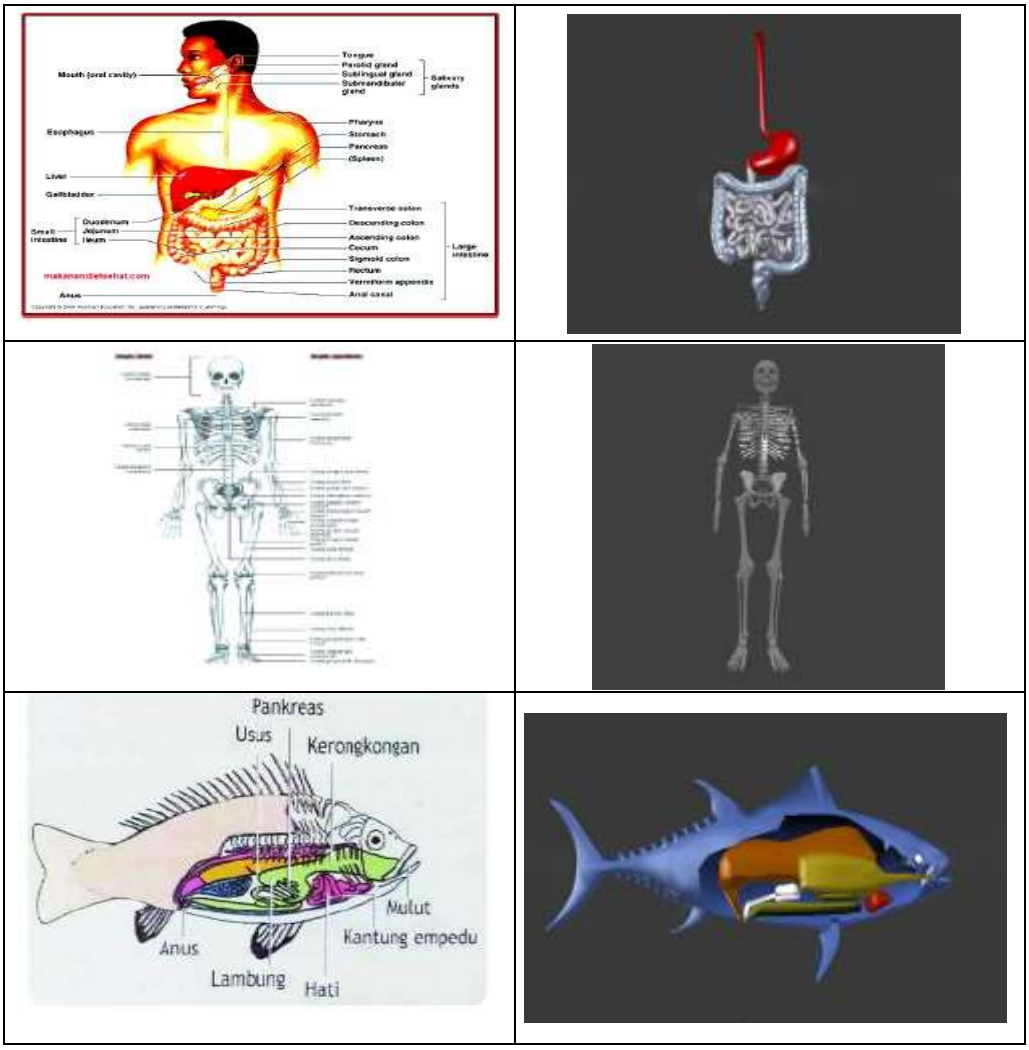


Gambar 2.14 Proses Pembuatan *Marker*

Langkah-langkah dalam pembuatan *marker* pada *IN2AR* dapat dilihat pada gambar 2.14 dengan penjelasan sebagai berikut:

1. Pembuat aplikasi mempersiapkan gambar yang akan dijadikan sebagai *marker* dengan ketentuan format gambar harus .JPG ukuran minimal 550 *pixel.*
2. Pembuat aplikasi mengisi data pembuat *marker* pada sistem *IN2AR* berupa Nama, *E-mail*, dan meng-*upload file marker* berupa .JPG dalam bentuk satuan atau dalam bentuk grup yang tergabung dalam format .ZIP maksimal 10 gambar berformat .JPG dan mengirim *file* gambar tersebut.
3. Proses *generate free marker* dalam beberapa detik bahkan hari bergantung terhadap kecepatan koneksi internet dan banyaknya *file* gambar yang akan di *generate* menjadi format .ass.
4. Setelah proses *generate* selesai, sistem *IN2AR* akan memberikan konfirmasi bahwa *file marker* sudah di *generate* menjadi format .ass dan dapat di *download* dan dapat diuji apakah *marker* dapat mendeteksi atau tidak pada sistem *IN2AR.*

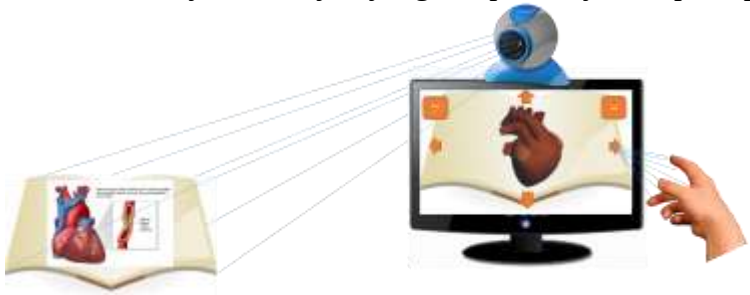
File Marker Tracking	
Marker	**Objek 3D**
	
	



Gambar 3.15 *Marker* Yang Digunakan

3.1.7 Analisis Metode *Motion Detection*

Proses interaksi dengan objek 3D menggunakan *motion detection* dimana proses berinteraksi dengan metode tersebut dengan cara menggerakan tangan pada kamera *realtime* dan menyentuh objek yang tampil dilayar. Seperti pada gambar 3.16.



Gambar 3.16 Interaksi *Motion Detection*

Proses pendeteksian gerakan dimulai dengan pengaktifan kamera yang berguna untuk menangkap setiap *frame* tangkapan kamera untuk dideteksi oleh sistem, terjadi pergerakan atau tidak terjadi pergerakan, pengkodean aktif kamera diatur pada kode *function initWebcam.*

```
Kamus:
camera:Camera
algoritma:
camera <- Camera.getCamera()
camera.setMode(WIDTH, HEIGHT, FPS)
video <- new Video(WIDTH, HEIGHT)
video.attachCamera(camera)

if (mirror <- true)
   video.scaleX <- -1
   video.x +<- video.width
endif
addChild(video)

mt <- new CMotionTacker(video)
view <- new Bitmap(new BitmapData(width, height)
addChild(video)
view <- new Bitmap(new BitmapData(640, 480))

box1 <- new Box(0x00000A)
addChild(box1)
PernafasanManusia <- new Loader()
PernafasanManusia.load(new
URLRequest("../resources/button/PERNAFASAN
MANUSIA.png"))
PernafasanManusia.x <- 500
PernafasanManusia.y <- 5
```

```
this.addChild(PernafasanManusia)
video.addEventListener(Event.EXIT_FRAME, loop)
```

Pada *pseudocode* tersebut *Camera.getCamera* sebagai fungsi pemanggilan parameter kamera agar dapat di akses dan saat kamera aktif pengaturan mode kamera/video berdasarkan *width*, *height* dan *fps*. Parameter yang digunakan sebagai *pointer* gerakan tangan ditampung pada sebuah variabel *box1*. Parameter yang digunakan pada PernafasanManusia.x dengan nilai sebesar 500px hal ini guna untuk mengatur posisi sumbu x *button* menu pada layar dan PernafasanManusia.y dengan nilai 5px mengatur sumbu y dari posisi *button*. Pengenalan terhadap pendeteksian pergerakan diatur pada *class CmotionTacker*.

Pendeteksian gerakan berdasarkan terhadap citra masukan yang memungkinkan sistem mengenali gerakan, gerakan akan terdeteksi akan ditandai dengan sebuah *pointer* berbentuk *box*.

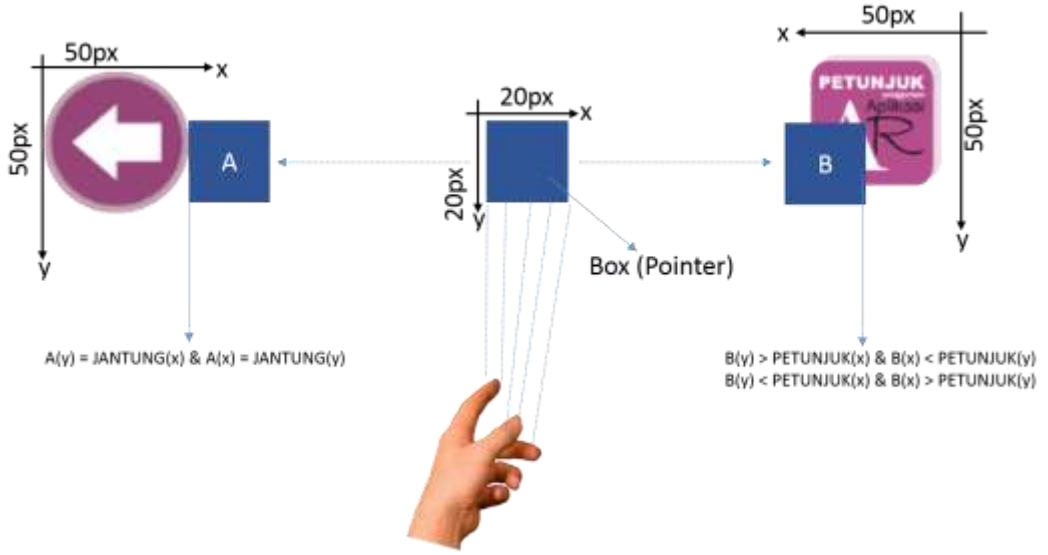
```
Kamus:
P:Point
Algoritma:
P <- new Point()
if (mt.track())
     p.x <- 640-(mt.x + view.x)
     p.y <- mt.y + view.y

     box1.x <- p.x - view.x
     box1.y <- p.y
endif
box1.addEventListener(Event.ENTER_FRAME, check)
```

Pada *pseudocode* tersebut *pointer* ditampung pada sebuah variabel *p: Point = new Point()*jika terjadi pergerakan pada kamera sistem akan melakukan *tracking* terhadap gerakan tersebut dengan menandai titik pergerakan

dengan sebuah *pointer* yang memiliki parameter dimensi X dan Y dan model objek 3D ditampung pada sebuah variabel `model:DAE <- new DAE().`

Tumbukan antar objek diatur melalui parameter fungsi milik *actionscript* yaitu menggunakan fungsi *hitTestObject*, pada *pseudocode function check* diatur suatu kondisi tumbukan antar objek melalui hasil deteksi pergerakan yang ditangkap melalui webcam secara *realtime*, kondisi yang menyebabkan tumbukan terjadi apabilah gerakan tangan(*pointer*) menyentuh fitur interaksi *button* model objek 3D yang muncul pada layar, lalu objek tersebut akan memberikan umpan balik (*feedback*) berupa *event* berdasarkan fungsi objeknya masing-masing.



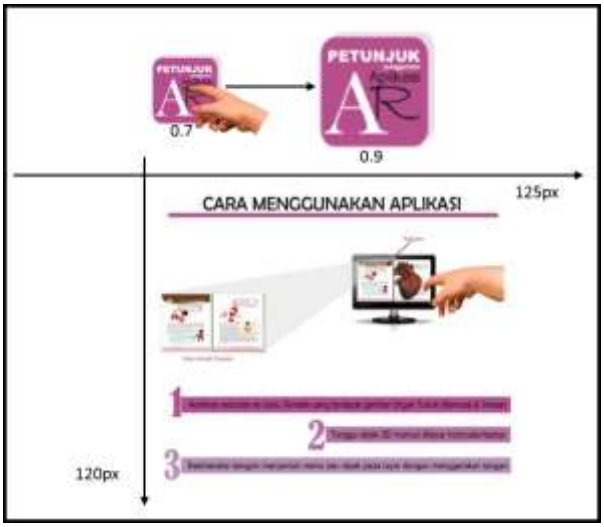
Gambar 3.17 Tumbukan Antar Objek

```
if (e.target.hitTestObject(ButtonPetunjuk))
     ButtonPetunjuk.scaleX <- 0.9
     ButtonPetunjuk.scaleY <- 0.9

     Petunjuk = new Loader()
     Petunjuk.load(new
URLRequest("../resources/button/Petunjuk.jpg"))
     Petunjuk.x <- 125
     Petunjuk.y <- 120
     this.addChild(Petunjuk)
```

```
else
     ButtonPetunjuk.scaleX <- 0.7
     ButtonPetunjuk.scaleY <- 0.7
```

Pada *pseudocode* tersebut parameter yang dijadikan objek tumbukan yaitu *button* `ButtonPetunjuk` dimana jika terjadi tumbukan *button* tersebut akan melakukan pembesaran *button* dalam skala sebesar 0.9 pada sumbu x dan y dan jika sudah tidak terkena tumbukan maka *button* akan *zoomin* ke skala awal sebesar 0.7 pada sumbu x dan y. Sedangkan parameter penampung terhadap objek *event* `ButtonPetunjuk` dari *button* tersebut ditampung pada variabel `Petunjuk` dimana sumbu x sebesar 125px dan y sebesar 120px nya hal ini untuk mengatur penentuan posisi dari event yang terjadi. Ilustrasi dapat dilihat pada gambar 3.18.



Gambar 3.18 Ilustrasi Penempatan *Event*

Pengenalan gerakan dapat terjadi dikarenakan sistem menginsialisasi dua buah parameter penampung citra awal dan citra akhir dari hasil tangkapan *webcam*, pengkodean penampungan citra tersebut di atur pada *function CMotionTacker*.

```
input <- video
areaThreshold <- 50
trackImg <- new BitmapData(WIDTH, HEIGHT)
last <- new BitmapData(WIDTH, HEIGHT)
```

Pada *pseudocode* tersebut parameter yang digunakan untuk menampung citra awal yaitu *trackingImg* dan parameter yang digunakan untuk menampung citra akhir yaitu *last*. Sedangkan parameter yang digunakan untuk melakukan *threshold* pada *input*-an video merupakan `areaThreshold` dimana nilai yang diberikan sebesar 50. Perbandingan citra tersebut diatur pada matriks *filter* yang dikodekan pada *function track.*

```
kamus:
curr:BitmapData
rect:Rectagle
mat:Array
conv:ConvolutionFilter
area:int
algoritma:
curr <- new BitmapData(WIDTH, HEIGHT)
curr.draw(input)
rect<- new Rectagle (0,0, WIDTH, HEIGHT)
trackImg <- curr.clone()
trackImg.draw(last, null, null, BlendMode.DIFFERENCE)
mat <- [0, 0, 0,  0, 15, 0,  0, 0, 0]
conv <- new ConvolutionFilter(3, 3, mat)
conv.divisor <- 1
trackImg.applyFilter(trackImg, rect, p, conv)
trackImg.applyFilter(trackImg,       null,       null,
BlendMode.INVERT)
trackImg.applyFilter(trackImg,     rect,     p,     new
BlurFilter(8, 8))
```

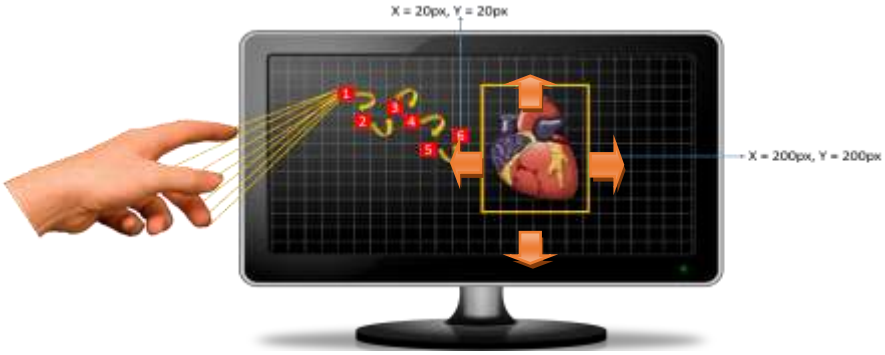
```
area <- brect.width * brect.height
motion <- false
     if (area > areaThreshold)
     motion <- true
     this.x <- brect.x + brect.width / 2
     this.y <- brect.y + brect.height / 2
last.copyPixels(curr, rect, p)
     return motion
     endif
```

Proses pendeteksian gerakan diperhalus dengan menggunakan `BlurFilter` agar proses pendeteksian lebih *smooth* dan lebih halus untuk mendeteksi setiap piksel pergerakannya. Penghitungan jumlah gerakan yang dapat terdeteksi oleh sistem akan *valid* jika lebih dari ambang batas, dimana area tangkapan lebih besar dari *areaThreshold*.

3.1.8 Analisis Interaksi Pengguna Aplikasi

Analisis interkasi pengguna aplikasi merupakan analisis yang menggambarkan bagaimana guru berinteraksi dengan aplikasi *augmented reality* pengenalan organ tubuh manusia dan hewan. Untuk berinteraksi dengan aplikasi, guru menggunakan masukan berupa gerakan tangan pada jangkauan kamera *webcam* komputer/laptop. Setiap perubahan *pixel* dari pergerakan pada tangkapan kamera akan di jadikan masukkan interaksi, dimana aplikasi akan menandai area pergerakan dengan sebuah *box* sebagai *pointer*, *pointer* tersebut akan mengikuti area dimana terjadi gerakan. Guru berinteraksi dengan menggerakan tangan ke arah *button* sebagai fitur interaksi yang terdapat dilayar seperti *button* petunjuk dan *reset*, jika *pointer box* mengenai *button* maka *button* tersebut akan memberikan *event* balikan. Guru juga dapat berinteraksi dengan objek 3D yang tampil dilayar dari hasil *tracking marker*, saat objek muncul dilayar maka guru dapat melepaskan *marker* dari tangkapan *webcam*, karena objek 3D yang sudah tampil otomatis akan di *pause*. Proses interkasi dengan cara menggerakan tangan

dan mengerahkan ke fitur *button* pada layar berupa rotasi persepektif , hal ini bertujuan untuk dapat melihat dari setiap sudut pandang terhadap objek 3D.



Gambar 3.19 Interkasi Pengguna dengan Aplikasi

Pergerakan tangan akan ditangkap oleh *webcam*, pergerakan tangan tersebut berupa *pointer box* seperti pada gambar 3.19, kotak merah berisi angka 1, 2, 3, 4, 5, 6 merupakan *pointer* yang merepresentasikan terjadinya gerakan berdasarkan terjadi perpindahan *pointer* berdasarkan *pixel*, jika *pointer* mengenai fitur *button* objek 3D maka akan menghasilkan suatu *event*.

3.2 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem, diantaranya kebutuhan perangkat keras *(hardware)*, perangkat lunak *(software)*, serta pengguna *(user)* sebagai bahan analisis kebutuhan yang harus dipenuhi dalam perancangan sistem yang akan diimplementasikan.

3.2.1 Analisis Perangkat Keras

Komputer adalah sebuah perangkat keras yang tidak luput dari perangkat lunak sebagai interaksinya. Perangkat lunak memberikan perintah agar perangkat keras berfungsi dengan baik. Dalam pembangunan aplikasi menggunakan perangkat keras sebagai pendukungnya sebagai berikut :

1) *Processor* Intel(R) Core(TM) i3-3110M CPU @ 2.40GHz
2) RAM 2GB
3) *Harddisk* dengan kapasitas 464GB

4) Sistem Operasi *Windows* 8
5) *Motherboard* dengan *chipset* yang kompatibel dengan VGA Card yang digunakan
6) VGA Card dengan kemampuan grafis 3D, seperti NVIDIA GeForce dan ATI Radeon
7) *Mouse* dan *Keyboard*
8) *Webcam* 1.3 Mega Pixel
9) *Printer* Canon IP1980
10) *Speaker* Simbadda

Sedangkan spesifikasi minimum perangkat keras yang dibutuhkan oleh pihak pengguna untuk teknologi AR seperti berkut ini :

1) *Processor* Dual Core minimal 1.7GHz (Quad Core disarankan)
2) RAM minimal 2 GB
3) *Harddisk* dengan kapasitas minimal 50 GB
4) Sistem Operasi Windows 7
5) *Motherboard* dengan chipset yang kompatibel dengan VGA card yang dipakai
6) VGA card dengan kemampuan grafis 3D, seperti NVIDIA GeForce dan ATI Radeon
7) *Webcam* minimal 1 Mega Pixel
8) *Speaker*

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 sistem adalah sebagai berikut :

1) Sistem operasi Windows 8
2) FlashDevelop Version 4.0.0
3) Autodesk 3Ds Max 2010

4) In2ar SDK
5) Papervision3D SDK
6) Astah Profesional

Sedangkan perangkat lunak yang dibutuhkan oleh pihak pengguna untuk aplikasi ini adalah sebagai berikut :

1) Sistem Operasi *Windows* 7
2) Aplikasi AR
3) Adobe Flash Player

3.2.3 Analisis Pengguna

Analisis pengguna dimaksudkan untuk mengetahui siapa saja pengguna yang terlibat dalam proses penggunaan sistem. Pengguna yaitu orang yang dapat mengerti dan memahami komputer sehingga dapat menggunakan aplikasi yang akan dibangun. Pengguna dari sistem yang akan dibagun adalah Guru. Hak akses yang diberikan melihat objek 3D yang dihasilkan oleh sistem serta dapat berinteraksi dengan objek.

Tabel 3.3 Karakteristik Pengguna

Pengguna	Hak Akses Aplikasi	Tingkat Pendidikan	Pengalaman
Guru	Melihat, Berinteraksi dan Menyampaikan Informasi	Guru Bidang IPA SD	Minimal dapat menjalankan 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. Kebutuhan fungsional juga berhubungan dengan fitur perangkat lunak yang akan dibuat. Analisis akan dimodelkan dengan menggunakan UML (*Unified Modelling Language*). Tahap-tahap pemodelannya antara lain identifikasi *actor*, *use case*

diagram, skenario *use case diagram, activity diagram, class diagram, sequence diagram.*

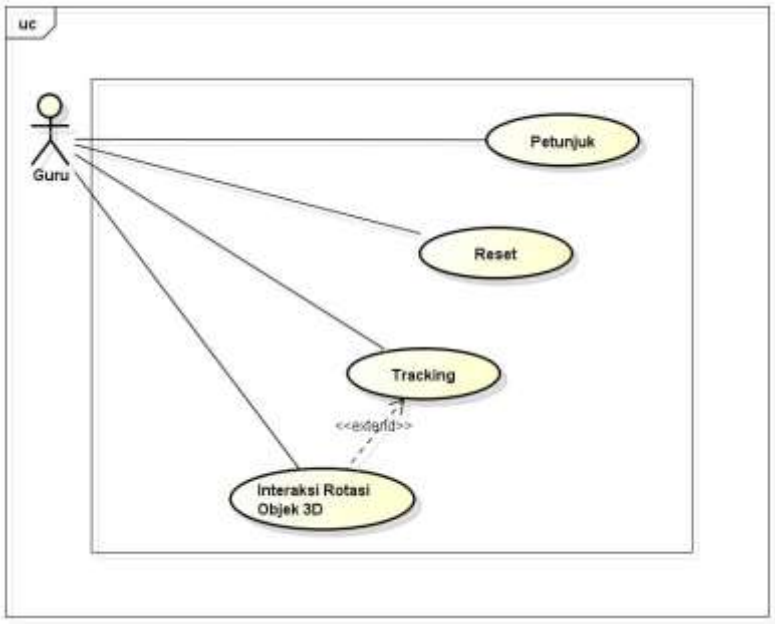
3.3.1 *Use Case Diagram*

Use Case Diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem yang akan dibuat. *Use Case* mendekripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Pada *use case* terdapat 1 aktor yaitu, guru. Penjelasannya lihat pada tabel 3.4.

Tabel 3.4 Aktor dalam *Use Case*

Nama Aktor	Definisi
Guru	Orang yang berinteraksi dengan sistem dan dapat menggunakan semua fungsi yang terdapat pada sistem.

Dari hasil analisa aplikasi yang ada, maka *use case diagram* untuk aplikasi ini dapat dilihat pada gambar 3.20.



Gambar 3.20 *Use Case Diagram*

3.3.1.1 Definisi *Use Case Diagram*

Bagian ini menjelaskan definisi dari masing-masing dari *use case diagram* yang dilakukan oleh guru. Mulai dari proses memilih masuk kedalam aplikasi sampai dengan objek ditampilkan.

Tabel 3.5 Definisi *Use Case Diagram*

No	Kode *Use Case*	Nama *Use Case*	Keterangan
1	UCAR01	*Tracking*	Fungsional untuk membandingkan gambar tangkapan dengan gambar yang terdapat pada *file tracking* dan me-*render* objek 3D.
2	UCAR02	Interaksi Rotasi Objek 3D	Fungsional sebagai sarana interaksi antara guru dan fitur menu *button* serta objek 3D pada sistem.
3	UCAR03	Petunjuk	Fungsional untuk melihat informasi tentang tata cara penggunaan aplikasi.
4	UCAR04	*Reset*	Fungsional untuk membersihkan layar dari hasil *event* menu *button* dan hasil *rendering* objek 3D.

3.3.1.2 Skenario *Use Case Diagram*

Skenario *use case* mendiskripsikan urutan dari tahapan-tahapan dalam proses bisnis, baik yang dilakukan aktor terhadap sistem maupun yang dilakukan sistem terhadap aktor. Skenario *tracking* terkait dengan *use case tracking* dapat dilihat pada tabel 3.6.

Tabel 3.6 Skenario *Use Case Tracking*

Identifikasi	
Nama *Use Case*	*Tracking*
Kode	UCAR01
Aktor	Guru
Tujuan	Mendeteksi dan mencocokan pola *marker* dengan data yang sudah terdapat pada *file tracking* sistem dan me-*render* objek 3D.
Kondisi Awal	Sistem menampilkan gambar *marker* pada aplikasi dilayar komputer.
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Guru mengarahkan *marker* berupa	

gambar pada Buku Tematik Terpadu ke kamera.	
	2. Melakukan pendeteksian dan pencocokan pola *marker* dengan *file tracking*.
	3. Sistem menampilkan status "Marker Terdeteksi" pada layar.
	4. Melakukan proses *rendering* untuk menampilkan objek 3D bersadarkan *marker tracking*.
Skenario Gagal	
Aksi Aktor	**Reaksi Sistem**
	1. Sistem menampilkan status "Marker Tidak Terdeteksi" pada layar.
Kondisi Akhir	Sistem menampilkan hasil *rendering* objek 3D ke layar.

Skenario interaksi rotasi objek 3D terkait dengan *use case* Interaksi Rotasi Objek 3D dapat dilihat pada tabel 3.7.

Tabel 3.7 Skenario *Use Case* Interaksi Rotasi Objek 3D

Identifikasi	
Nama *Use Case*	Interaksi Rotasi Objek 3D
Kode	UCAR02
Aktor	Guru
Tujuan	Berinteraksi menggunakan gerakan tangan dengan *button* rotasi objek 3D.
Kondisi Awal	Sistem menampilkan *button* rotasi dan objek 3D berdasarkan *marker tracking*.
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Guru mengarahkan tangan ke kamera *webcam* untuk dideteksi.	
	2. Sistem akan mendeteksi gerakan tangan masukan.
3. Guru menyentuh *button* rotasi objek 3D yang muncul dilayar.	
	4. Sistem mendeteksi tumbukan *(Collusion Detection)* antara *button* rotasi objek 3D dengan objek tangan (*pointer*).
	5. Sistem memberikan aksi balikan dari *button* rotasi objek pada layar komputer.

Skenario Gagal	
Aksi Aktor	**Reaksi Sistem**
	1. Objek 3D tidak memberikan aksi balikan merotasi.
Kondisi Akhir	Sistem menampilkan aksi objek 3D yang terkena tumbukan gerakan tangan dengan menunjukan objek 3D berputar.

Skenario petunjuk terkait dengan *use case* Petunjuk dapat dilihat pada tabel 3.8.

Tabel 3.8 Skenario *Use Case* Petunjuk

Identifikasi	
Nama *Use Case*	Petunjuk
Kode	UCAR03
Aktor	Guru
Tujuan	Menampilkan informasi tata cara penggunaan aplikasi.
Kondisi Awal	Sistem menampilkan *button* menu Petunjuk dilayar kamera *realtime*.
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Guru mengarahkan tangan ke *webcam* dan menggerakan tangan untuk dideteksi.	
	2. Melakukan pendeteksian terhadap gerakan yang tertangkap oleh *webcam*.
	3. Mendeteksi tumbukan *button* menu Petunjuk dengan objek tangan.
	4. Menjalankan aksi tumbukan dengan *button* Petunjuk.
Skenario Gagal	
Aksi Aktor	**Reaksi Sistem**
	1. *Button* petunjuk tidak menampilkan informasi tata cara penggunaan aplikasi.
Kondisi Akhir	Sistem menampilkan informasi tata cara penggunaan aplikasi.

Skenario *reset* terkait dengan *use case Reset* dapat dilihat pada tabel 3.9.

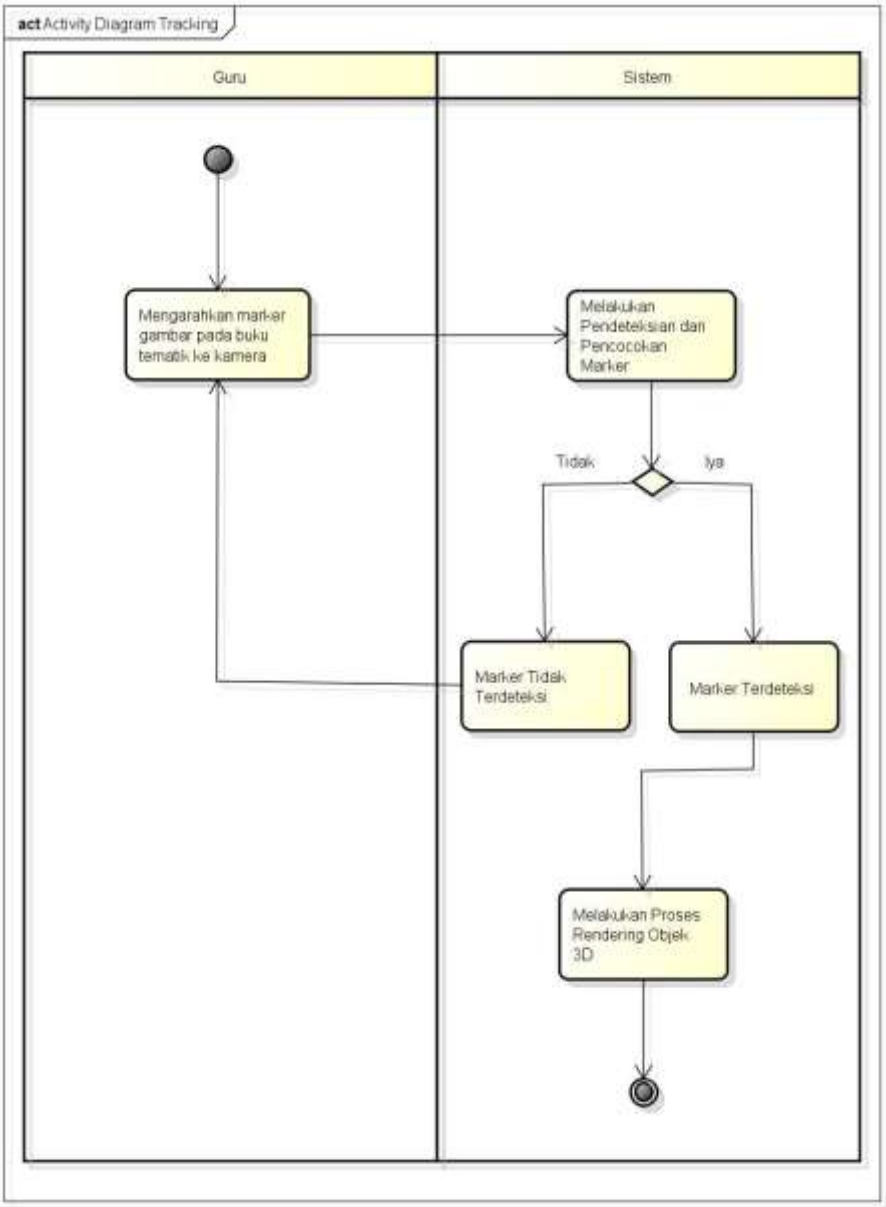
Tabel 3.9 Skenario *Use Case Reset*

Identifikasi	
Nama *Use Case*	*Reset*
Kode	UCAR04
Aktor	Guru
Tujuan	Membersihkan layar dari tampilan informasi pada *button* menu dan Objek 3D.
Kondisi Awal	Sistem menampilkan *button* menu *Reset* dilayar kamera *realtime*.
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Guru mengarahkan tangan ke *webcam* dan menggerkan tangan untuk dideteksi.	
	2. Melakukan pendeteksian terhadap gerakan yang tertangkap oleh *webcam*.
	3. Mendeteksi tumbukan *button* menu *Reset* dengan objek tangan.
	4. Menjalankan aksi tumbukan dengan *button Reset*.
Skenario Gagal	
Aksi Aktor	**Reaksi Sistem**
	1. *Button reset* tidak membersihkan layar dari informasi dan objek 3D dilayar.
Kondisi Akhir	Sistem membersihkan layar dari informasi dan objek 3D yang tampil dilayar.

3.3.2 *Activity Diagram*

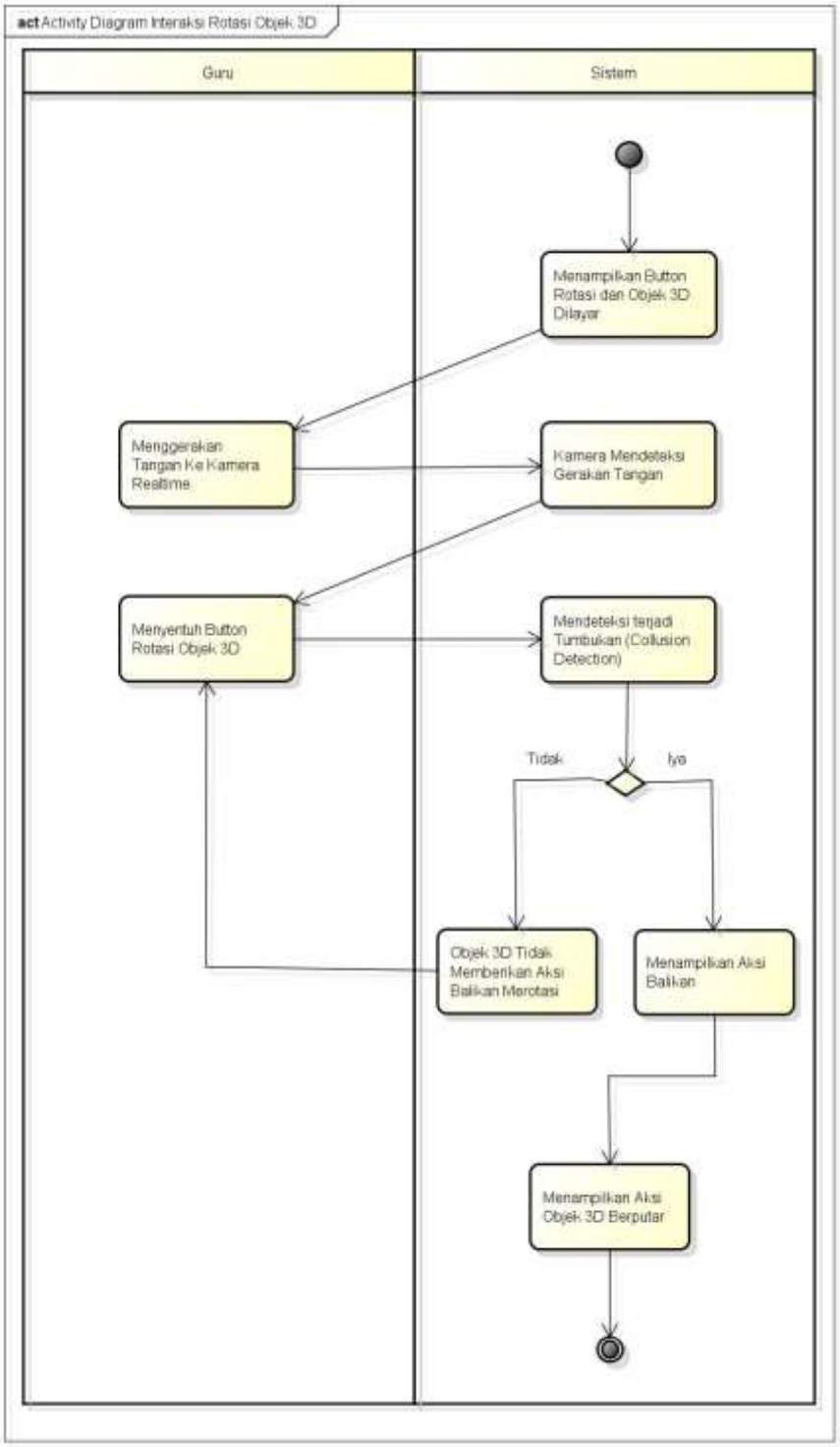
Activity Diagram atau diagram aktifitas menggambarkan *workflow* (aliran kerja) aktor pada saat kamera *webcam* menyorot *marker* agar dapat melakukan *tracking* yang akan dicocokan polanya.

Gambaran *activity diagram* terkait *use case tracking* dapat dilihat pada gambar 3.21



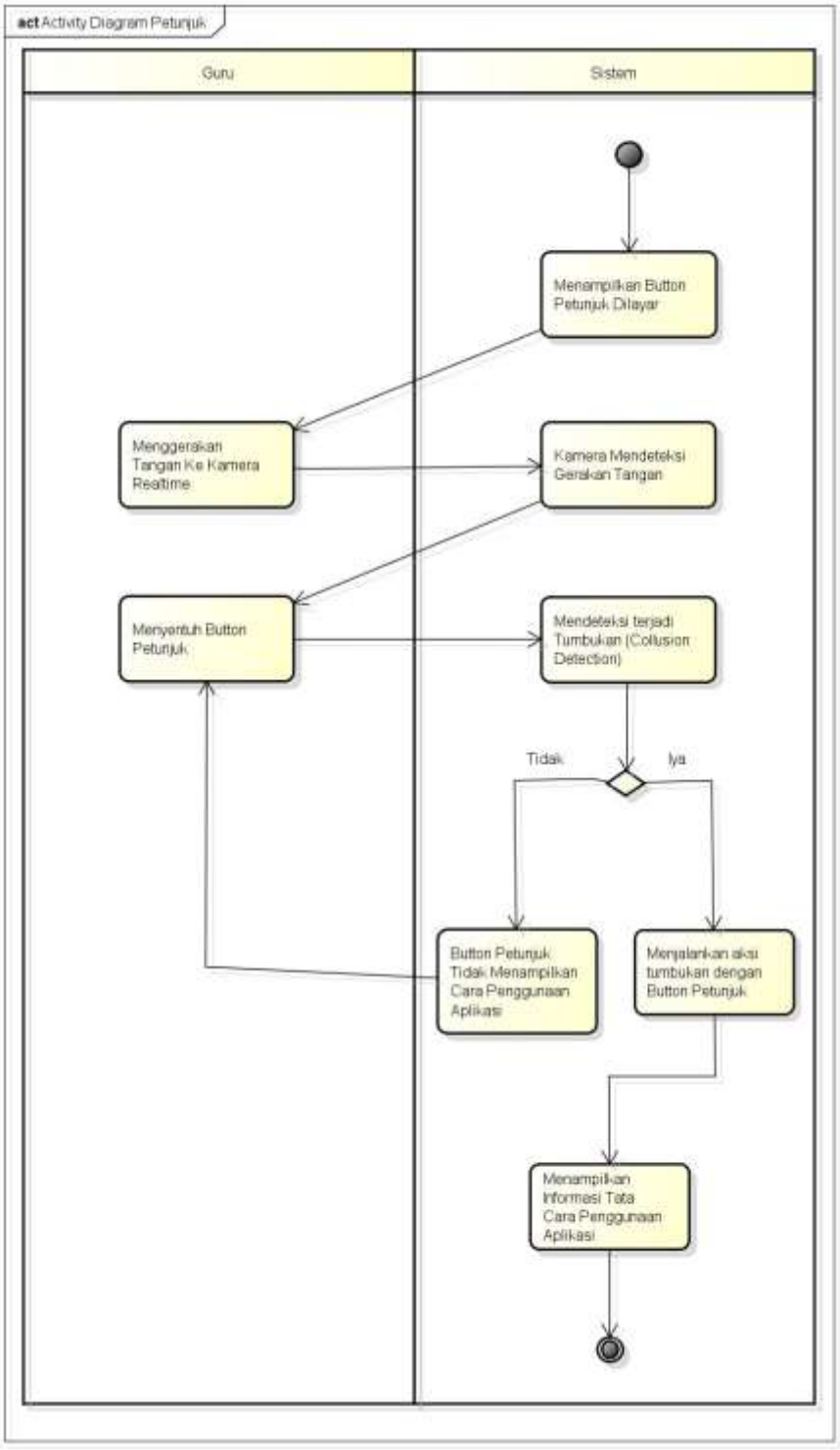
Gambar 3.21 *Activity Diagram Tracking*

Gambaran *activity diagram* terkait *use case* Interaksi Rotasi Objek 3D dapat dilihat pada gambar 3.22.



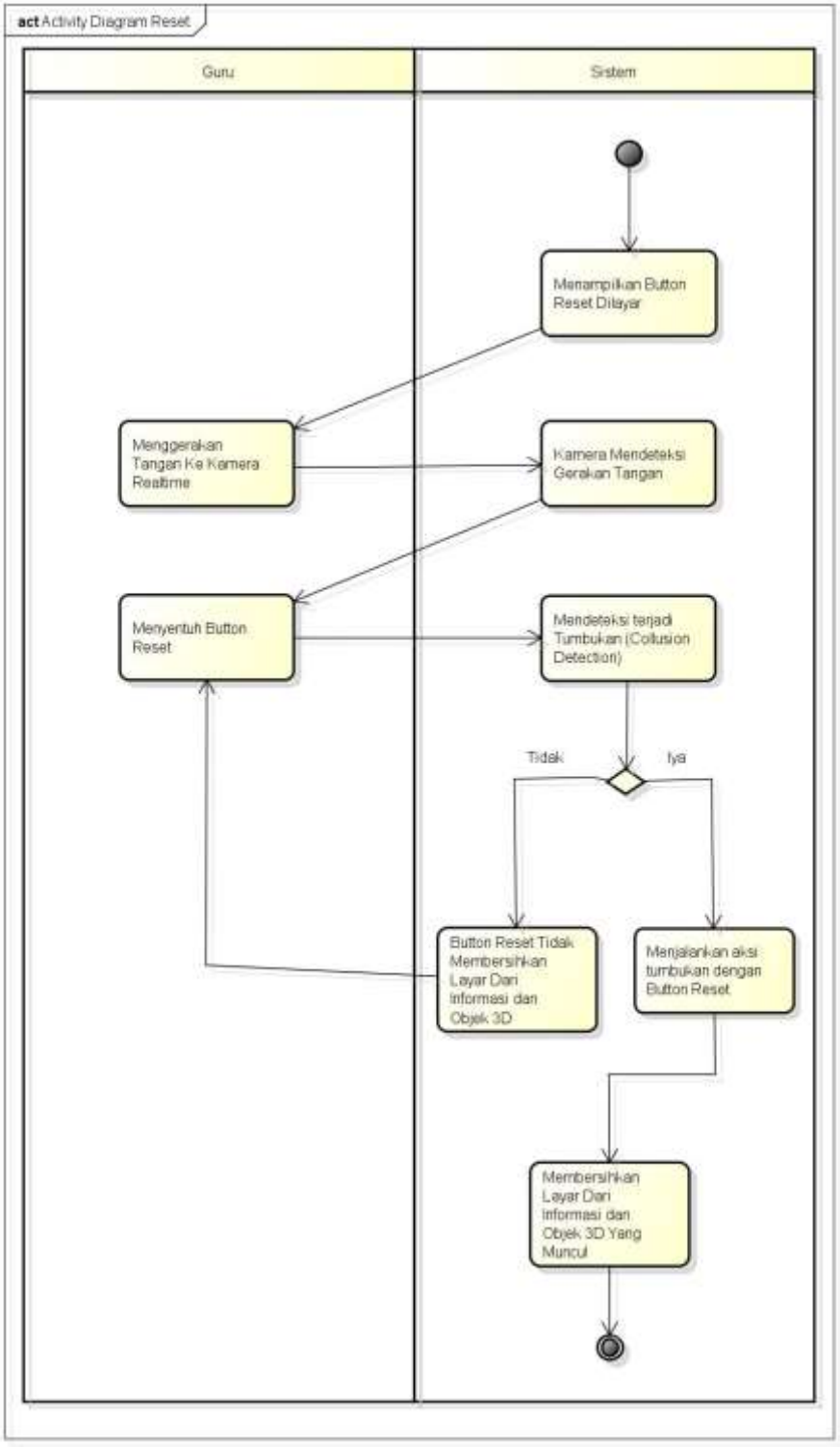
Gambar 3.22 *Activity Diagram* Interaksi Rotasi Objek 3D

Gambaran *activity diagram* terkait *use case* Petunjuk dapat dilihat pada gambar 3.23.



Gambar 3.23 *Activity Diagram* Petunjuk

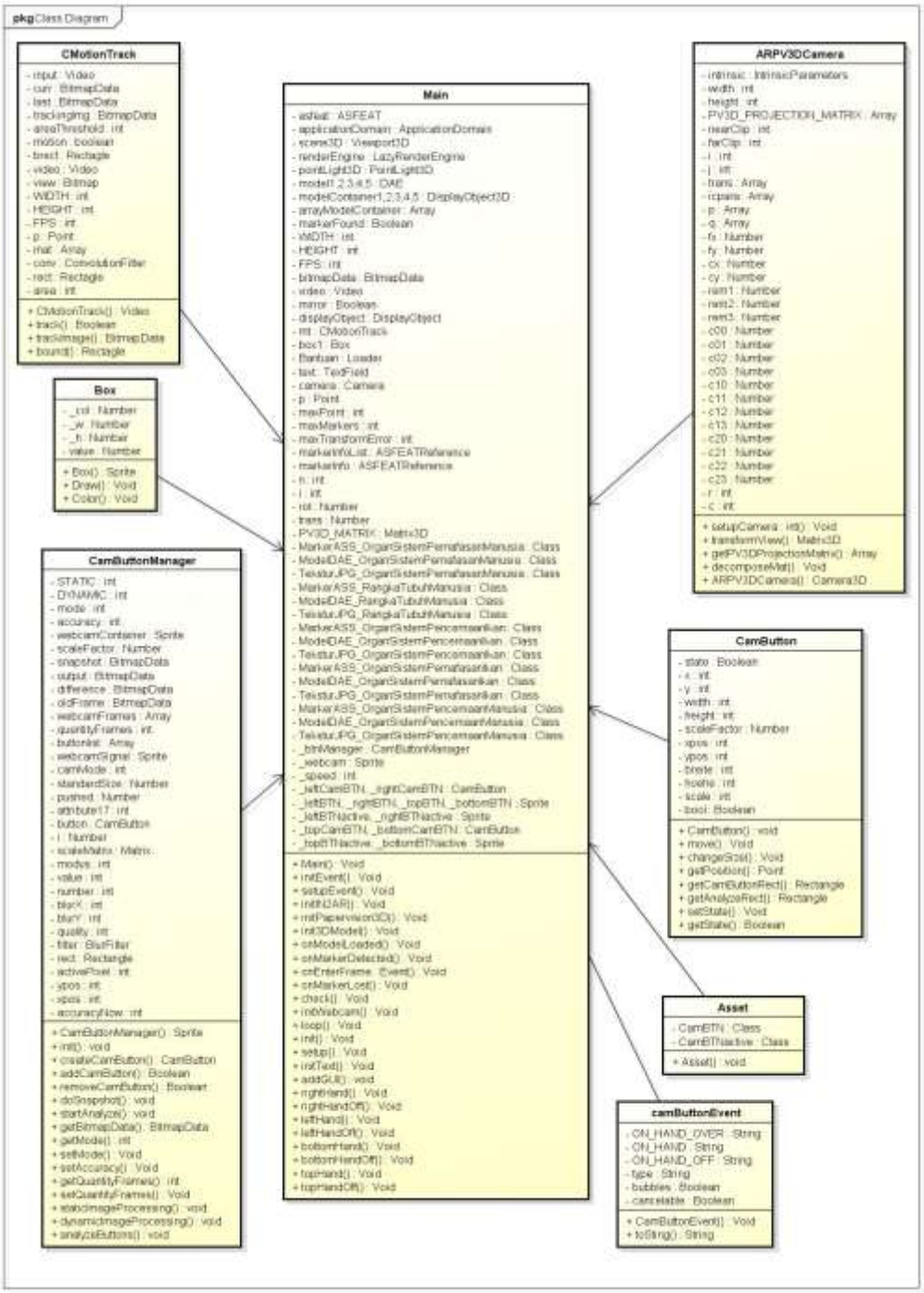
Gambaran *activity diagram* terkait *use case Reset* dapat dilihat pada gambar 3.24.



Gambar 3.24 *Activity Diagram Reset*

3.3.3 *Class Diagram*

Class Diagram merupakan gambaran struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Berikut adalah *class diagram* dari sistem yang dibangun.



Gambar 3.25 *Class Diagram* Sistem

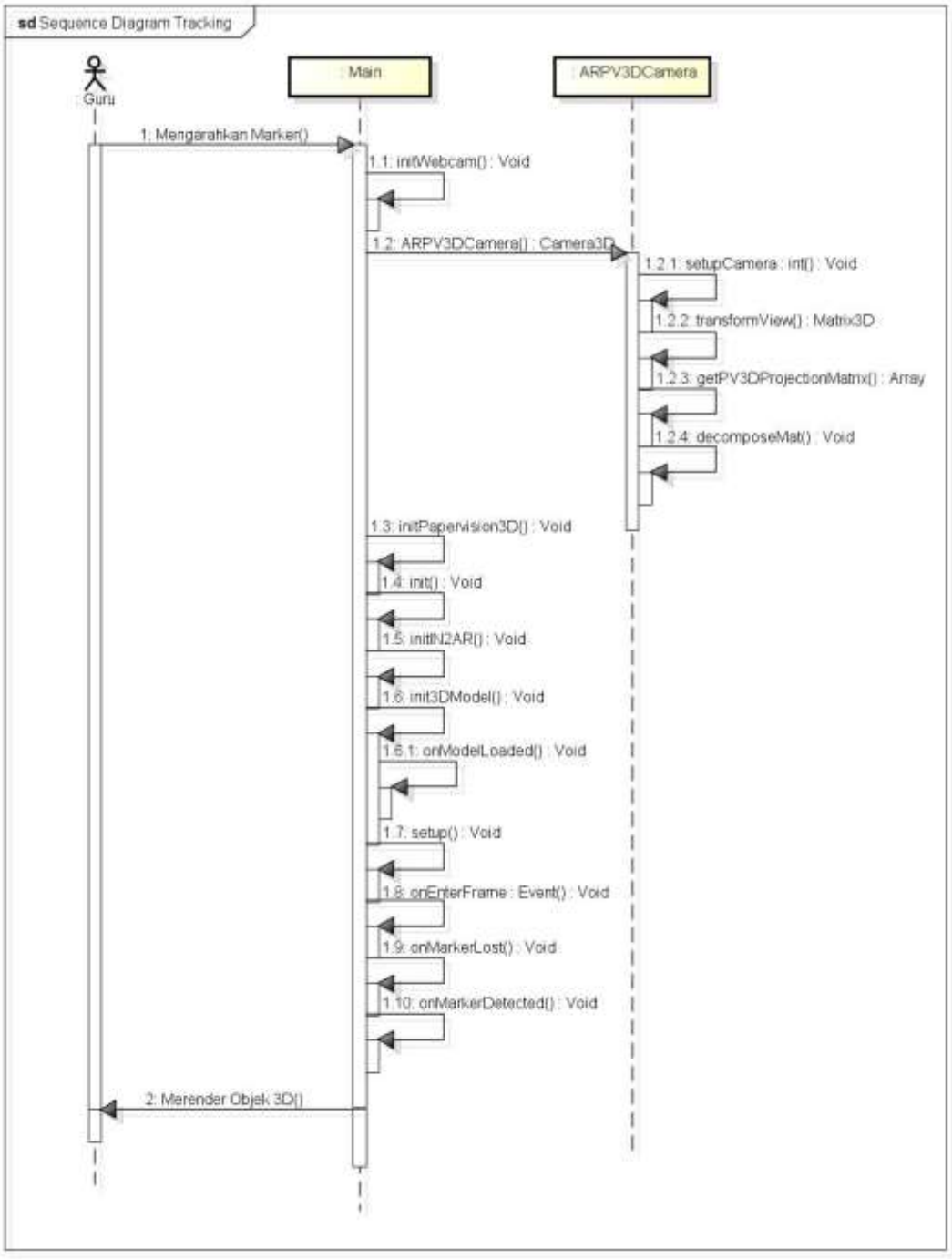
Penjelasan dari masing-masing *class diagram* dapat dilihat pada tabel 3.10.

Tabel 3.10 Deskripsi *Class Diagram* Sistem

User Application		
Class	**Jenis *Class***	**Deskripsi**
Main	*Boundary*	*Class* yang menampilkan *interface* layar interaksi dengan pengguna berupa kamera *realtime* dan hasil *rendering* objek 3D.
Box	*Control*	*Class* yang mengatur *pointer* yang dijadikan masukan interaksi dari gerakan tangan.
ARPV3Dcamera	*Control*	*Class* yang mengaktifkan fungsionalitas kamera pada *webcam 3D* untuk ditampilkan.
CmotionTacker	*Control*	*Class* yang mengatur model interaksi gerakan menggunakan *motion detection*.
CamButtonManager	*Control*	*Class* yang mengatur pendeteksian gerakan pada *button* dan mengatur seluruh aktifitas *button*.
CamButton	*Control*	*Class* yang mengatur fungsionalitas untuk pengaturan *button* rotasi objek 3D.
CamButtonEvent	*Control*	*Class* yang menginisialisasikan fungsionalitas untuk menginisialisasi *event* pada *button*.
Asset	*Entity*	*Class* yang menginisialisasikan fungsionalitas untuk menampung *file button* yang digunakan untuk berinteraksi dengan objek 3D.

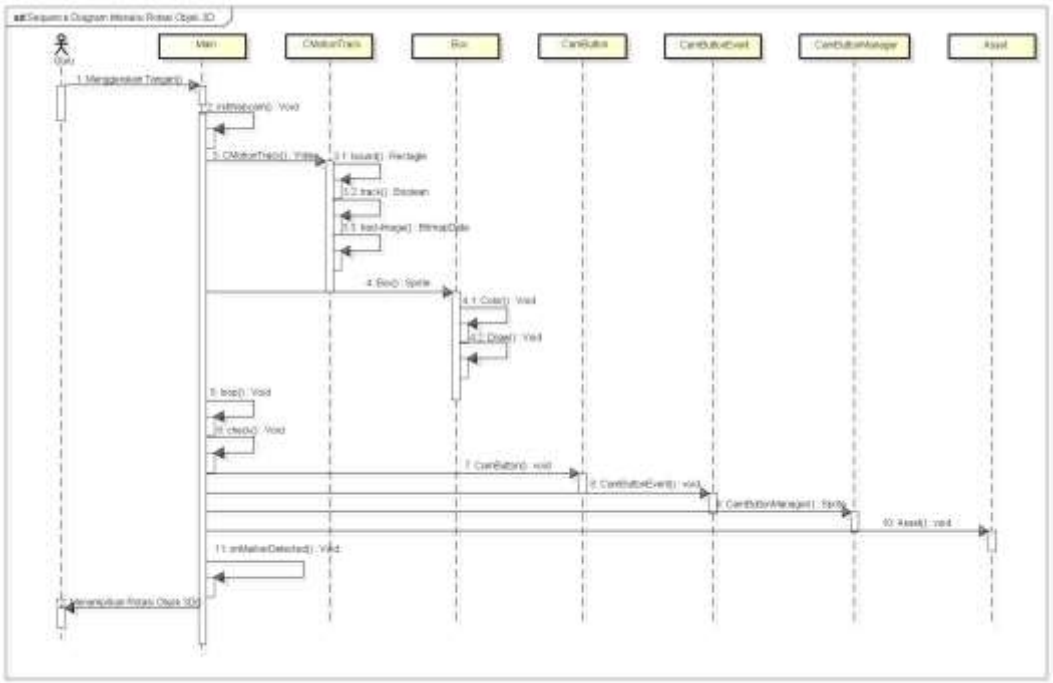
3.3.4 *Sequence Diagram*

Sequence Diagram menggambarkan kelakuan objek pada *use case* dengan mendiskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Gambaran yang ada pada *sequence diagram tracking* dapat dilihat pada gambar 3.26.



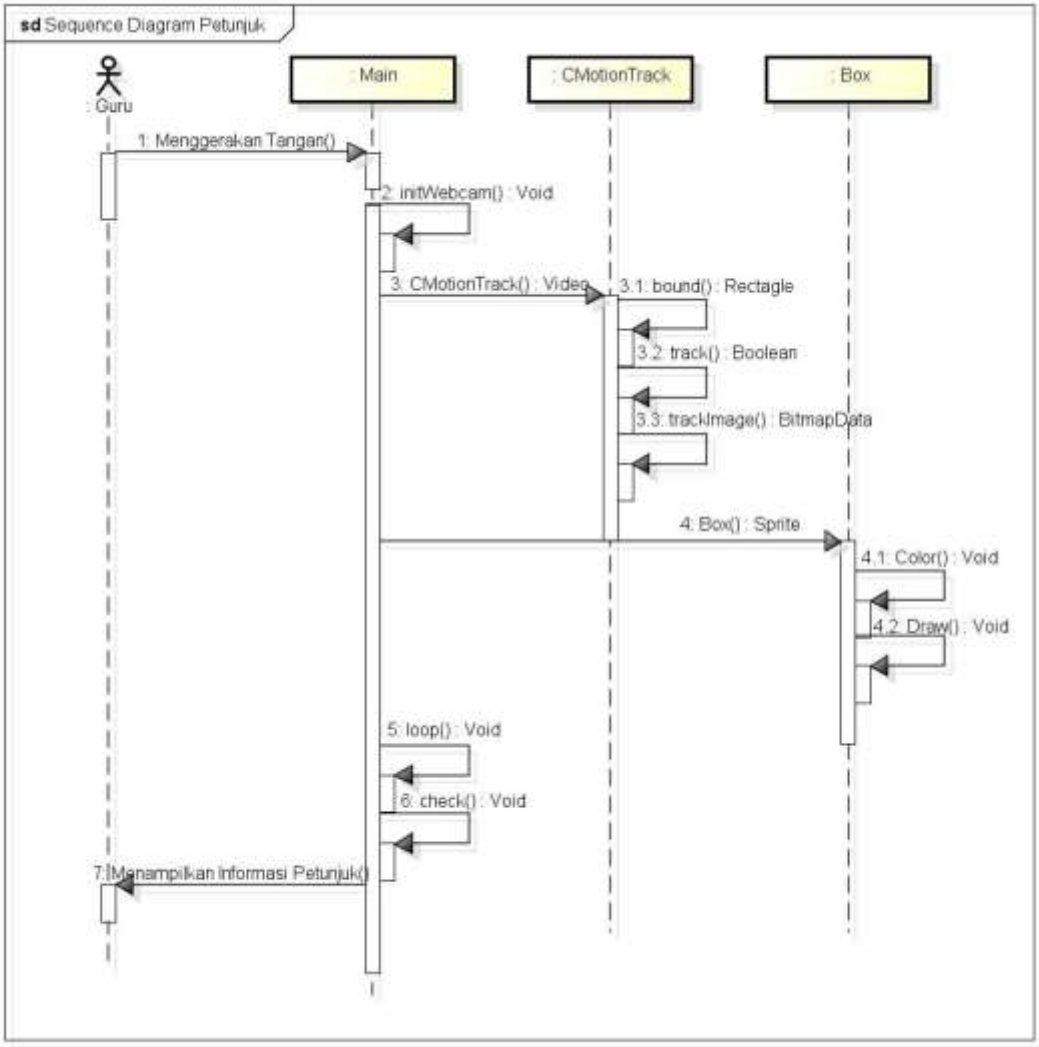
Gambar 3.26 *Sequence Diagram Tracking*

Gambaran yang ada pada *sequence diagram* interaksi rotasi objek 3D dapat dilihat pada gambar 3.27.



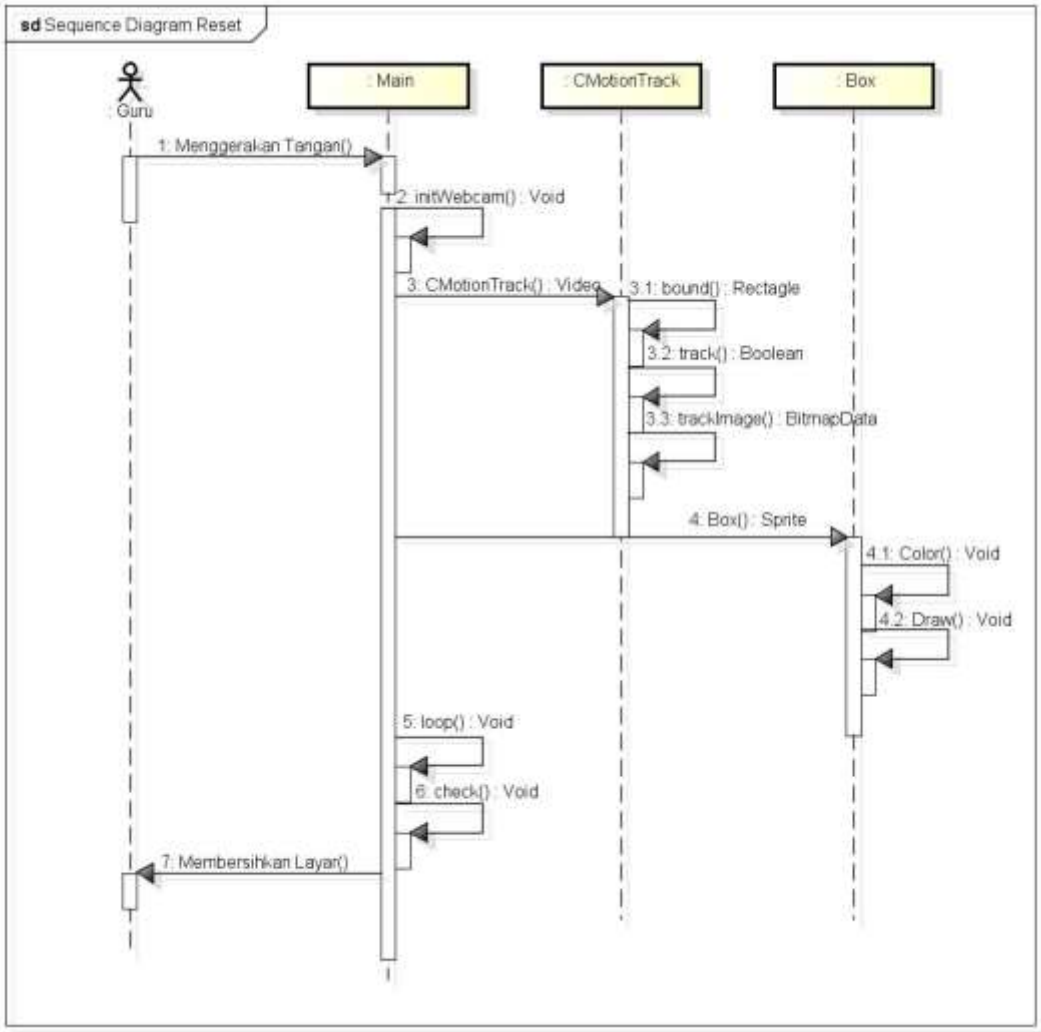
Gambar 3.27 *Sequence Diagram* Interaksi Rotasi Objek 3D

Gambaran yang ada pada *sequence diagram* petunjuk dapat dilihat pada gambar 3.28.



Gambar 3.28 *Sequence Diagram* Petunjuk

Gambaran yang ada pada *sequence diagram reset* dapat dilihat pada gambar 3.29.



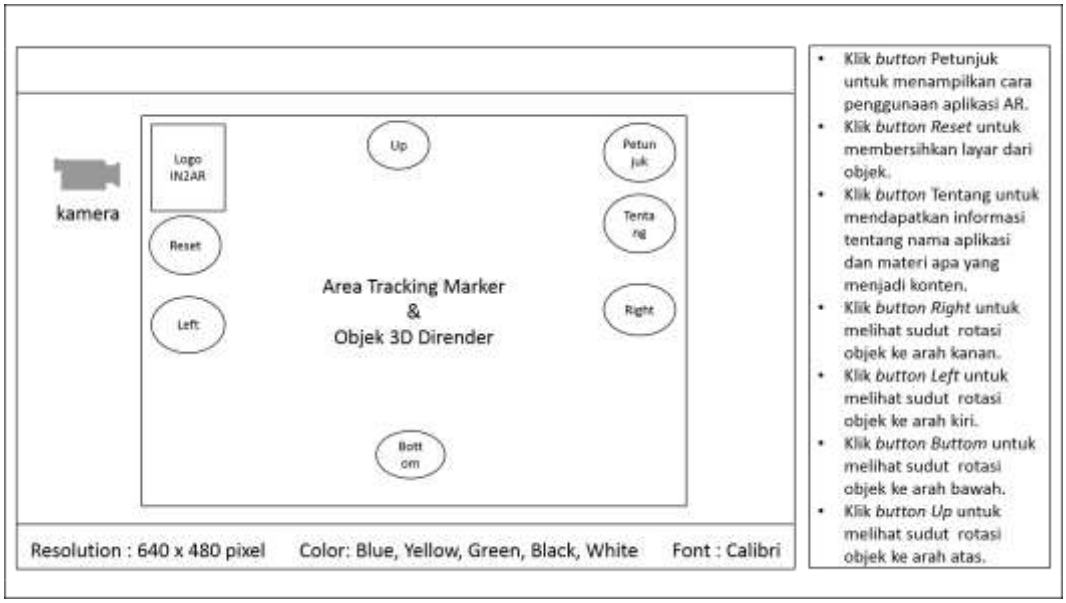
Gambar 3.29 *Sequence Diagram Reset*

3.3.5 Perancangan Sistem

Perancangan akan dimulai setelah tahap analisis terhadap sistem selesai dilakukan. Perancangan dapat diartikan sebagai penggambaran, perencanaan, dan pembuatan sketsa dari beberapa elemen yang terpisah kedalam satu kesatuan yang utuh dan berfungsi.

3.3.5.1 Perencangan Antarmuka

Perancangan antarmuka bertujuan untuk mencari bentuk yang optimal dari aplikasi yang akan dibangun dengan pertimbangan faktor-faktor permasalahan dan kebutuhan yang ada pada sistem seperti yang telah ditetapkan pada tahap analisis. Adapun perancangan antarmuka aplikasi yang akan dibangun dapat dilihat pada gambar 3.30.



Gambar 3.30 Perancangan Antarmuka