BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem merupakan tahap untuk mempelajari interaksi sistem yang terdiri dari pelaku proses dalam sistem, prosedur, data serta informasi yang terkait. Analisis dilakukan terhadap kondisi yang sedang berjalan sebagai dasar perancangan atau perbaikan sistem lama. Analisis sistem dalam mengembangkan aplikasi adalah sebagai berikut: Analisis kondisi eksisting, analisis sistem non fungsional yang meliputi analisis *user* yang terlibat, analisis perangkat keras dan perangkat lunak yang digunakan.

3.1.1 Analisis Kondisi Eksisting

Analisis kondisi eksisting merupakan gambaran umum tentang cara penyampaian informasi yang dilakukan Museum Geologi. Media penyampaian informasi yang digunakan Museum Geologi menggunakan beberapa media diantaranya ; media visual (foto, poster dan benda koleksi museum), media audio visual (video, animasi) dan beberapa aplikasi interaktif. Media visual mempunyai kelebihan biaya murah, alat dan pemeliharaannya sederhana. Sementara kelemahannya yaitu menimbulkan rasa bosan, kurang memiliki daya tarik, keterikatan pada suatu ukuran dan memiliki rasa keterbatasan dengan audien [3]. Gambar 3.1 merupakan contoh media visual.



Gambar 3.1 Media Visual di Museum Geologi

Media audio visual yang ada di Museum Geologi memiliki kelebihan dapat menimbulkan penafsiran yang berbeda, informasi dapat diterima sesuai dengan kenyataan dan tidak membosankan. Sementara memiliki kelemahan diantaranya ; biaya relatif mahal, kejelasan suara terkadang kurang bisa dipahami dan beberapa infromasi penyajiannya terikat pada jadwal. Gambar 3.2 merupakan salah satu media audio visual.



Gambar 3.2 Media Audio Visual di Museum Geologi

Sementara beberapa aplikasi interaktif yang ada di museum memiliki kelebihan memiliki daya tarik, lebih interaktif, mengurangi penggunaan tempat untuk koleki museum karena dalam satu aplikasi terdapat beberapa informasi dan penyajian informasi tidak terikat pada jadwal. Kelemahan dari aplikasi interaktif museum diantaranya ; biaya relatif mahal, proses *troubleshooting* yang rumit, aplikasi hanya dapat digunakan oleh satu orang dan aplikasi hanya dapat menampilkan satu informasi saja seperti gambar 3.3.



Gambar 3.3 Aplikasi Interaktif di Museum Geologi

3.1.2 Analisis Masalah

Berdasarkan analisis terhadap kondisi yang sedang berjalan yang telah dijelaskan, maka yang menjadi permasalahan dalam penelitian ini adalah:

1. Bagaimana cara menampilkan koleksi batu artefak yang ada di museum geologi tanpa memerlukan tempat yang luas ?
2. Bagaimana membuat aplikasi yang dapat menampilkan lebih dari satu informasi yang berbeda secara bersamaan ?
3. Bagaimana mengembangkan aplikasi pengenalan koleksi batu artefak menggunakan *marker* dengan teknologi *multitouch* ?

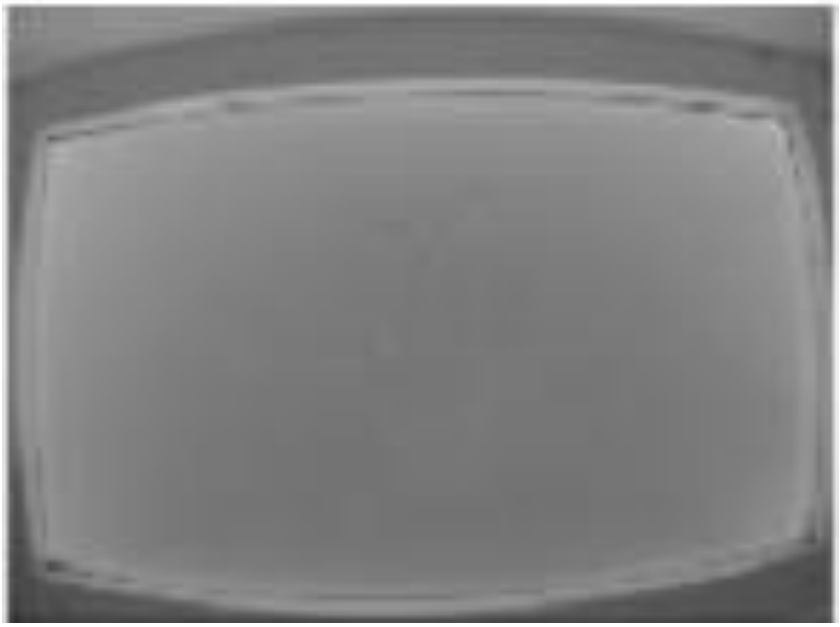
3.1.3 Gambaran Umum Sistem

Sistem yang akan dibangun memiliki empat tahapan yaitu data masukan, *Pre-Processing*, *Segmentation*, *Tracking* dan *communication Protocols*.

1. Tahap pertama adalah tahap mengambil data masukan dari kamera yang telah dilengkapi filter inframerah. Data yang diambil masih berupa data masukan mentah yang harus diolah terlebih .
2. Tahap kedua adalah *pre-processing* yang merupakan tahap dimana data masukan dibersihkan dari *noise* atau dari hal yang tidak mempunyai pengaruh dalam proses. P*re-processing* yang dilakukan dalam tahap ini adalah proses *Noise Reduction* dan *Grayscaling*.
3. Tahap ketiga adalah proses *segmentation*, yaitu tahapan memisahkan wilayah objek dan wilayah latar belakang.
4. Tahap empat adalah proses *Tracking,* dibutuhkan untuk melakukan pengecekan terhadap sentuhan secara bersamaan agar setiap sentuhan tidak mengganggu antara satu sama lain. Metoda *tracking* yang digunakan yaitu **Kalman Filter.** Filter ini bertujuan untuk mendeteksi pergerakan banyak objek serta mengikuti arah lintasan yang terbentuk.
5. Tahap kelima adalah tahap *communication Protocols* atau mengiriman data hasil pengolahan ke aplikasi menggunakan API TUIO *(Tangible User Interface Protocol).*

3.1.4 Analisis Masukan

Data masukan diambil melalui kamera *webcam* yang telah dimodifikasi dengan menambahkan filter inframerah. Sentuhan jari dan *marker* pada layar berfungsi sebagai *input* untuk sistem yang akan diterima oleh kamera, namun hasil interpretasi apa yang kamera lihat pada layar, masih membutuhkan proses transformasi sehingga dapat dimengerti oleh aplikasi. Contoh gambar asli sebelum pengolahan data dapat dilihat pada gambar 3.4



Gambar 3.4 Data masukan sebelum proses pengolahan

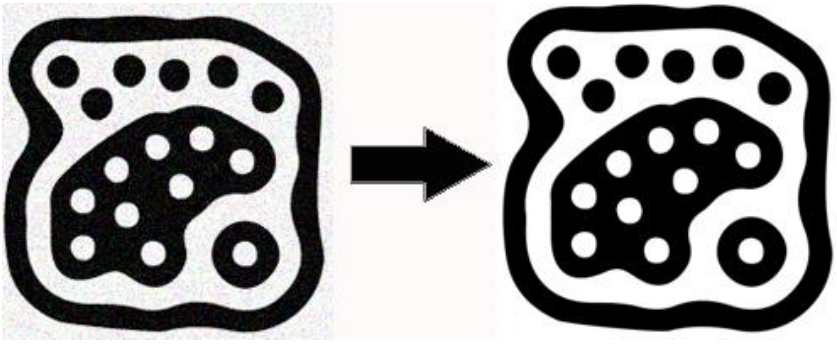
3.1.5 Pre-Processing

Preprocessing adalah proses pengolahan gambar asli sebelum gambar tersebut diolah. Gambar yang ditangkap oleh kamera selanjutnya melewati proses *noise reduction* dan *grayscaling*. Pada proses ini menggunakan perangkat lunak *Reactivison* untuk membantu proses *noise reduction* dan *grayscaling*.

3.1.5.1 Noise Reduction

Pada saat proses *capture* (pengambilan gambar), ada beberapa gangguan yang mungkin terjadi, seperti munculnya bintik-bintik yang bisa jadi disebabkan oleh proses *capture* yang tidak sempurna. Setiap gangguan pada citra dinamakan dengan *noise*. *Noise* pada citra tidak hanya terjadi karena ketidak-sempurnaan dalam proses *capture*, tetapi bisa juga disebabkan oleh kotoran-kotoran yang terjadi

pada citra (biasanya *Gaussian Noise*). *Noise gaussian* merupakan model *noise* yang mengikuti distribusi normal standar dengan rata-rata nol dan standard deviasi 1. Efek dari *gaussian noise* ini, pada gambar muncul titik-titik seperti gambar 3.5.



Gambar 3.5 Contoh *Gaussian Noise*

Pada *Reactivision,* filter yang digunakan untuk menghilangkan *noise* bintik-bintik menggunakan filter ***gaussian***. Berikut ini merupakan fungsi yang digunakan untuk menghilangkan *gaussian noise* secara vertikal pada *Reactivision* dengan menggunakan filter *gaussian*.

```
void main (void) {
    float kernel_size = 2.0;
    float rho = 20.0;

    vec2 direction = vec2(0.0,1.0);


    float dx = 1.0 / size_cx;

    float dy = 1.0 / size_cy;
    vec2  st = gl_TexCoord [0].st;

    vec4    color = vec4 (0.0, 0.0, 0.0, 0.0);
    float   weight = 0.0;
    for (float i = -1.0*kernel_size; i <= kernel_size ; i+=1.0) {
        float fac = exp (-(i * i) / (2.0 * rho * rho));
        weight += fac;
        color += texture2D (tex, st + vec2 (dx*i, dy*i)
                 * direction) * fac;
    }

    gl_FragColor =  color / weight;
}
```

Selain menghilangkan *Noise Gaussian* secara vertikal, *Reactivision* juga menghilangkan *Noise Gaussian* secara horizontal, Seperti pada fungsi berikut ini merupakan fungsi yang digunakan untuk menghilangkan *Noise Gaussian* secara horizontal menggunakan filter *gaussian*.

```
void main (void) {

    float rho = 20.0;


    vec2 dir = direction < 0.5 ? vec2(1.0,0.0) : vec2(0.0,1.0);

    float dx = 1.0 / size_cx;

    float dy = 1.0 / size_cy;
    vec2  st = gl_TexCoord [0].st;

    vec4    color = vec4 (0.0, 0.0, 0.0, 0.0);
    float   weight = 0.0;
    for (float i = -1.0*kernel_size ; i <= kernel_size ; i+=1.0) {
        float fac = exp (-(i * i) / (2.0 * rho * rho));
        weight += fac;
        color += texture2D (tex, st + vec2 (dx*i, dy*i) * dir) * fac;
    }

    gl_FragColor =  color / weight;
}
```

3.1.5.2 Grayscaling

Proses *grayscaling* merupakan proses konversi gambar dari gambar berwarna (RGB) menjadi gambar satu warna *(monochrome)*. Pada mode *grayscaling* memanfaatkan warna *gray* (abu-abu), dengan menggunakan komponen *luminance*. Pada *Reactivision* proses konversi gambar ke gambar *grayscale* menggunakan fungsi sebagai berikut.

```
void main(void)
{
    vec4 textureColor    = texture2D(tex, gl_TexCoord[0].st);
    float luminance      = dot(vec3 (0.2125, 0.7154, 0.0721),
                               vec3(textureColor));

gl_FragColor = vec4(luminance, luminance, luminance, 1.0);
}
```

3.1.6 Segmentation

Segmentasi citra bertujuan memisahkan wilayah (*region*) objek dengan wilayah latar belakang agar objek di dalam citra mudah dianalisis untuk mengenali objek.

3.1.6.1 Thresholding

Proses *Thresholding* citra adalah suatu metode yang digunakan untuk memisahkan antara obyek dan *background*. Wilayah objek diset berwarna putih sedangkan sisanya diset berwarna hitam. Hasil dari operasi pengambangan adalah citra biner yang hanya mempunyai dua derajat keabuan: hitam dan putih. Berikut ini fungsi *threshold* pada *Reactivision*.

```
#include "threshold.h"

#include <math.h>
#include <stdlib.h>

void simple_threshold( const unsigned char *source,
        int source_stride,
        unsigned char *dest,
        int width, int height,
        int threshold )
{
    const unsigned char *p = source;
    int count = width * height;
    int i;

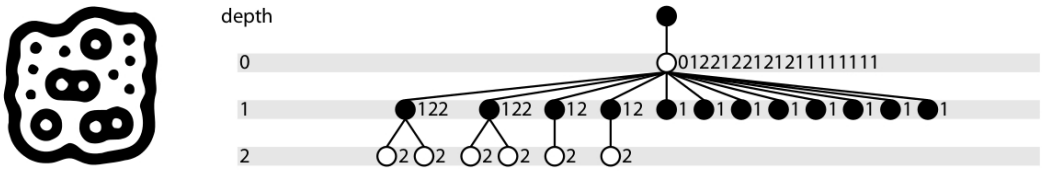
    for( i=0; i < count; ++i ){

        if( *p > threshold )
            dest[i] =  255;
        else
            dest[i] = 0;

        p += source_stride;
    }
}
```

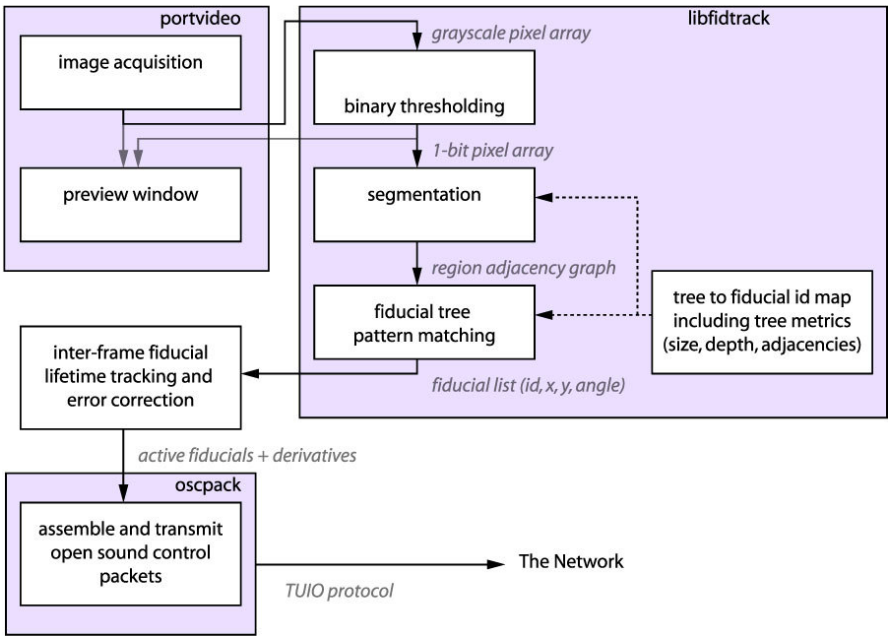
3.1.6.2 Fiducial Object

Fiducial object merupakan proses dimana, mendeteksi *fiducial marker*. Pada *Reactivision* untuk mendeteksi *fiducial marker* terdapat pencocokan pola citra (*pattern matching*) dengan menggunakan metode ***region adjacency graph*** untuk pembacaan *fiducial marker*. Sekumpulan piksel yang membentuk *region* dalam *graph* disebut *verteks*. *Region* yang saling berdekatan atau bertetangga dalam *graph* disebut sebagai *region adjacency graph* akan membentuk sebuah garis hubung antar *verteks* yang disebut *edge*. G merupakan contoh struktur topologi dari *fiducial*.



Gambar 3.6 Struktur topologi dari *fiducial*

Struktur topologi yang dihasilkan tersebut akan dibandingkan dengan struktur topologi yang terdapat pada *Reactivision*. Gambar 3.7 merupakan gambaran secara umum proses pembacaan *fiducial marker* pada *Reactivision*.



Gambar 3.7 Proses Pembacaan *fiducial*

Berikut ini fungsi yang digunakan *Reactivision* untuk medeteksi *fiducial marker* pada layar sentuhan.

```
// segmentation
step_segmenter( &segmenter, dest );
// fiducial recognition
int count = find_fiducials120( fiducials, MAX_FIDUCIAL120_COUNT,
&partialSegmentTopology, &segmenter);

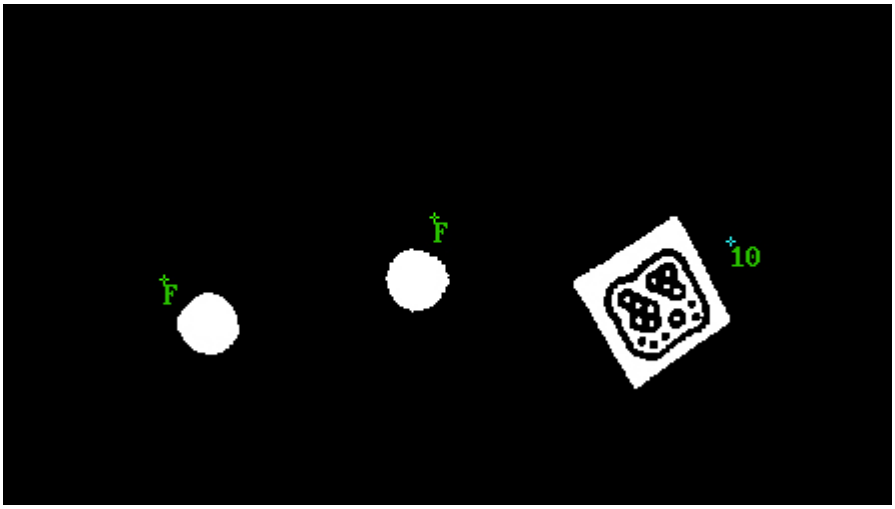
// process symbols
for(int i=0;i< count;i++) {

    //apply distortion
    int index = (int)floor(fiducials[i].y)*width+(int)floor(fiducials[i].x);
    fiducials[i].x = dmap[index].x;
    fiducials[i].y = dmap[index].y;

    FiducialObject *existing_fiducial = NULL;
```

3.1.6.3 Finger Object

Finger object merupakan proses dimana *Reactivision* mendeteksi sentuhan jari pada layar sentuhan. Seperti gambar 3.8.



Gambar 3.8 *finger and fiducial detection*

Untuk mendeteksi sentuhan pada layar sentuhan, prosesnya hampir sama dengan *fiducial object* yaitu memisahkan *background* dengan *object*. Ketika objek yang terbaca oleh *reactivison* berbentuk lingkaran, maka objek tersebut akan di bandingkan dengan *size* lingkaran sentuhan yang telah di tentukan. Apabila *size*

objek lingkaran yang terbaca sama dengan *size* lingkaran sentuhan yang sudah ditentukan maka objek lingkaran tersebut dikategorikan sebagai sentuhan dan pada *Reactivision* akan tampil simbol F yang mendefinisikan bahwa objek tersebut merupakan sentuhan jari. Berikut ini fungsi yang digunakan *Reactivision* untuk mendeteksi sentuhan.

```
int FidtrackFinder::check_finger(RegionX *finger, unsigned char *image,
                                 unsigned char *display) {

    int buffer_size = width*height;
    int mask_size = finger->width;
    if (finger->height>finger->width) mask_size = finger->height;

    float outer_radius = mask_size/2.0f;
    float inner_radius = mask_size/3.0f;

    int rounded_outer_radius = floor(outer_radius+0.5f);
    int rounded_inner_radius = ceil(inner_radius+0.5f);

    float xratio = (float)mask_size/(float)finger->width;
    float yratio = (float)mask_size/(float)finger->height;
    short outer_error = 0;
    short inner_error = 0;

    for (int x=0;x<mask_size;x++) {
        for (int y=0;y<mask_size;y++) {

            int px = finger->left+(int)floor(x/xratio+0.5f);
            int py = finger->top +(int)floor(y/yratio+0.5f);

            int finger_pixel = py*width+px;
            if ((finger_pixel>=buffer_size) || (finger_pixel<0)) return -1;

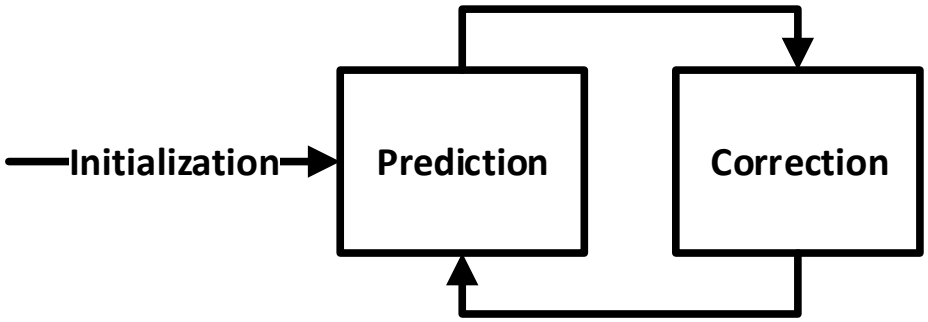
            float dx = x-outer_radius+0.5f;
            float dy = y-outer_radius+0.5f;
            int dist = floor(sqrtf(dx*dx+dy*dy)+0.5f);

            if (( dist> rounded_outer_radius) && (image[finger_pixel]==255))
            { outer_error++; } //display[finger_pixel*3+2]=255; }
            else if (( dist< rounded_inner_radius) && (image[finger_pixel]==0))
            { inner_error++; } //display[finger_pixel*3]=255; }
        }
    }
```

3.1.7 Tracking

Tracking adalah proses sangat penting pada teknologi *multitouch*. Proses *tracking* dibutuhkan untuk melakukan pengecekan terhadap sentuhan secara bersamaan agar setiap sentuhan tidak mengganggu antara satu sama lain. Metode *tracking* yang digunakan yaitu **Kalman Filter**. Filter ini bertujuan untuk

mendeteksi pergerakan banyak objek serta mengikuti arah lintasan yang terbentuk. Seperti gambar 3.9.

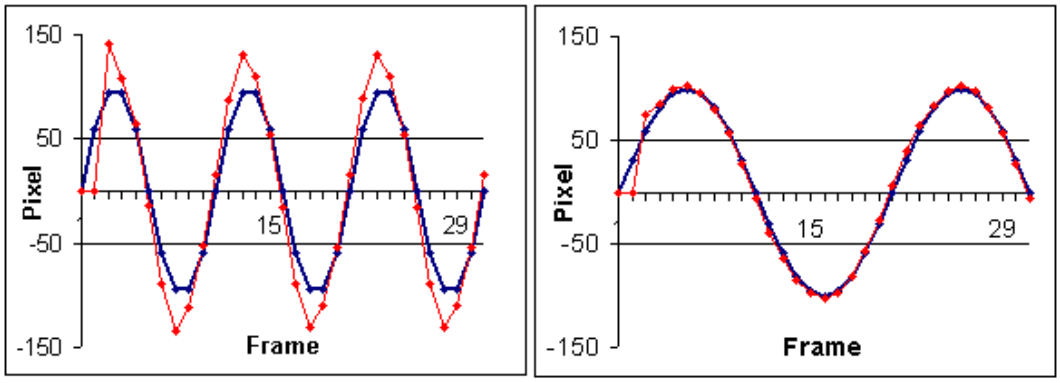


Gambar 3.9 Proses Kalman Filer

Berikut langkah secara umum kalman Filter untuk proses *tracking* :

1. ***Initialization***. Dalam langkah ini mencari posisi obyek di seluruh gambar
2. ***Prediction***. Pada proses ini memprediksi gerakan yang akan dilakukan objek dengan menggunakan titik X,Y dan *Angel* sebagai acuan prediksi.
3. ***Correction.*** Pada tahapan ini mengkoreksi prediksi gerakan yang akan dilakukan objek, jika gerakan objek tidak sesuai dengan prediksi gerakan.

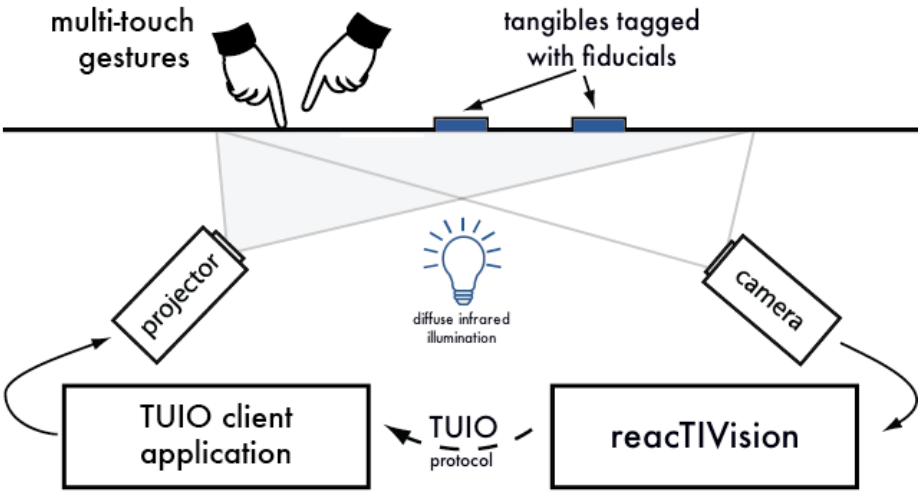
Pada gambar 3.10 merupakan contoh perbandingan gerakan antara gerakan prediksi *kalman filter* dengan gerakan *real*. Garis berwarna merah merupakan prediksi *kalman filter* dan garis bewarna biru merupakan gerakan *real* dari suatu objek.



Gambar 3.10 Contoh Prediksi Kalman Filer

3.1.8 Communication Protocols

TUIO *(Tangible User Interface Protocol)* adalah *framework open source* yang mendefinisikan secara umum protokol dan API untuk untuk proses pengiriman data sentuhan. Protokol ini mengkonversi (*encode*) data dari perangkat lunak *reactivison* ke aplikasi yang telah mempunyai kemampuan untuk membaca (*decoding*) data dari *protocol*. Alur pengiriman data secara umum seperti gambar 3.11.



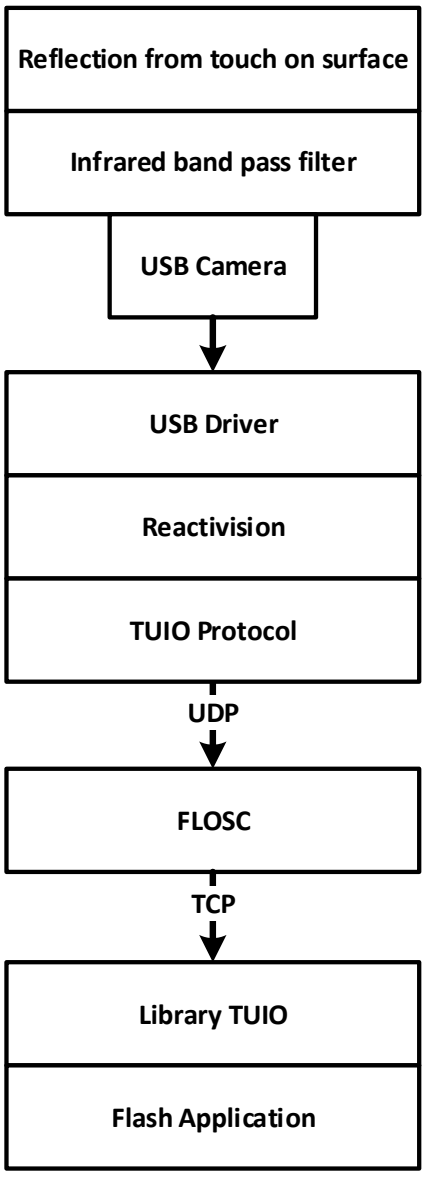
Gambar 3.11 Alur Pengiriman Data TUIO secara umum

Namun proses komunikasi protokol TUIO menggunakan UDP (*User* Datagram *Protocol*) agar waktu pengiriman data lebih cepat. Ketika menggunakan UDP kelemahannya adalah banyaknya data yang hilang. Oleh karena itu alternatif proses komunkasi data menggunakan TCP *(Transmision Control Protocol)* akan lebih aman, tetapi waktu yang dibutuhkan lebih lama. *Flash* membutuhkan untuk mengkonversi paket UDP ke TCP. Proses konversi bisa dilakukan dengan *tool* yang bernama FLOSC (Flash OSC).

3.1.9 Arsitektur Perangkat Lunak

Secara umum sistem yang akan dirancang ini memiliki alur proses seperti tertera pada gambar 3.12 dimana kamera akan menjadi peran utama dalam pembuatan sistem *multitouch*. Kamera menangkap bayangan infra merah dari objek atau sentuhan yang menyentuh layar sentuh. Gambar yang ditangkap kamera tersebut dikirim melalui driver kamera dengan kabel USB. Perangkat lunak

pendeteksi dalam hal ini *Reactivision*. Ketika perangkat lunak pendeteksi dalam hal ini *Reactivision* sudah selesai memproses input dari kamera, maka proses selanjutnya adalah mengirim data melalui protokol TUIO menggunakan UDP. Untuk aplikasi Flash data harus dikonversi terlebih dahulu menggunakan *tools* FLOSC dimana transfer data menggunakan TCP antara FLOSC dengan aplikasi Flash yang diterima oleh *library TUIO*. Data yang terima oleh *library TUIO* di konversi menjadi *event Touch* yang dapat digunakan oleh aplikasi flash. Berikut ini gambaran blok diagram arsitektur perangkat lunak.



Gambar 3.12 Blok diagram arsitektur perangkat lunak

3.1.10 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan *non*-fungsional merupakan analisis yang dibutuhkan untuk dapat menentukan spesifikasi dari kebutuhan sistem. Spesifikasi ini meliputi elemen atau perangkat-perangkat yang dibutuhkan untuk sistem yang akan dibangun sampai sistem tersebut dapat diimplementasikan. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang diperlukan sistem, *output* yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah *input* sehingga dapat menghasilkan suatu *output* yang diinginkan. Kebutuhan *non*-fungsional terbagi menjadi beberapa analisis yaitu analisis perangkat keras, analisis perangkat lunak dan analisis pengguna.

3.1.11 Analisis Perangkat Keras

Perangkat keras komputer atau *hardware* sangat mendukung dalam kinerja sistem. Semakin baik spesifikasi perangkat keras yang ada maka kinerja sistem pun akan semakin baik. Spesifikasi perangkat keras yang digunakan di Musem Geologi Bandung Ruang Sayap Barat lantai 1 seperti pada tabel 3.1.

Tabel 3.1 Spesifikasi Perangkat Keras Saat Ini

NO	Perangkat Keras	Spesifikasi
1	Processor	Intel Core i3 2.4 GHz
2	RAM	4 GB
3	Harddisk	500 GB
4	VGA	Nvidia G-force 1 GB
5	Keyboard dan Mouse	USB Wireless
6	Monitor	Layar LCD Samsung 32 Inch
7	Frame Touch Screen	*Touch Screen capacitive*
8	Tag RFID	
9	RFID *Reader*	

Sedangkan, spesifikasi minimum perangkat keras yang digunakan dalam pengembangan aplikasi pengenalan koleksi batu artefak menggunakan *marker* dengan tekonolgi *multitouch* adalah sebagai pada tabel 3.2.

Tabel 3.2 Spesifikasi Kebutuhan Minimum Perangkat Keras

NO	Perangkat Keras	Spesifikasi
1	Processor	Intel Quad Core 2.4 GHz
2	RAM	2 GB
3	Harddisk	160 GB
4	VGA	Nvidia G-force 1 GB
5	Keyboard + Mouse	USB Wireless
6	Proyektor	*Short Throw*
7	4 x Inframenrah	850nm (1 inframerah 48 LED)
8	*Webcam* dengan filter inframerah	Resolusi (640x480), filter 720nm
9	Permukaan kaca / akrilik	Ketebalan 6mm (80x80 cm)
10	Power supply DC	Minimal 12 V 2A
11	Kotak/Box	80x80x90 cm

3.1.12 Analisis Perangkat Lunak

Analisis kebutuhan perangkat lunak, dibutuhkan beberapa perangkat lunak yang mendukung untuk digunakan dalam implementasi. Pada tabel 3.3 merupakan perangkat lunak yang sudah terpasang pada komputer di Museum Geologi Ruang Sayap Barat lantai 1.

Tabel 3.3 Spesifikasi Perangkat Lunak Saat Ini

NO	Perangkat Lunak	Spesifikasi
1	Sistem Operasi	Windows 7 Professional
2	Flash Player	Flash Player 11

Berdasarkan penjelasan mengenai perangkat lunak diatas, maka disimpulkan bahwa *software* yang telah terpasang pada komputer di lantai 1 ruang sayap barat museum geologi belum memenuhi kriteria untuk implementasi sistem. Maka dari itu perlu penambahan *software* yang dibutuhkan untuk mendukung aplikasi yang akan dibangun adalah seperti pada tabel 3.4

Tabel 3.4 Spesifikasi Kebutuhan minimum Perangkat Lunak

NO	Perangkat Lunak	Spesifikasi
1	Sistem Operasi	Windows XP
2	Flash Player	Flash Player 9
3	Adobe Flash	CS 3
4	*Reactivision*	Version 1.4
5	flosc	

3.1.13 Analisis Pengguna

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

1. *User Knowledge* and *Experience*

Klasifikasi *knowledge and experience* dari pengguna aplikasi pengenalan koleksi batu artefak menggunakan *marker* dengan teknologi *multitouch* dapat dilihat pada tabel 3.5

Tabel 3.5 Klasifikasi *Knowledge and Experience*

Klasifikasi	Keterangan
Education Level	Bisa digunakan oleh berbagai kalangan, seperti ; pelajar, mahasiswa hingga masyarakat awam
Reading Level	Bisa digunakan oleh berbagai *level* pendidikan dengan *reading level* yang sedang.
Typing Skills	Tidak memerlukan *typing skills* yang tinggi.
Computer Literacy	*Moderate* (menengah)
Task Experience	Bisa digunakan oleh pengguna dengan pengalaman menggunakan aplikasi *touchscreen* yang awam.
Application Experience	Bisa digunakan pada sistem operasi *windows* XP atau lebih tinggi
System Experience	Bisa digunakan oleh pengguna dengan pengalaman menggunakan aplikasi *touchscreen* yang awam.
Native Language	Menggunakan dua bahasa yaitu bahasa Indonesia dan bahasa inggris
Use of other system	Membutuhkan aplikasi pendukung lainnya untuk menjalankan aplikasi.

2. *User Physical Characteristic*

Beberapa hal yang harus diperhatikan terhadap karakteristik fisik Pengguna untuk menggunakan aplikasi ini yaitu *color blind*, *handedness*, *gender* dan *height*. Klasifikasi *user physical characteristic* dari pengguna seperti tabel 3.6.

Tabel 3.6 Klasifikasi *User Physical Characteristic*

Klasifikasi	Keterangan
Age	14 – 50 Tahun
Gender	Pria dan Wanita
Handedness	Kanan dan Kiri
Color Blind	Pengguna yang tidak bisa membedakan warna yang satu dengan yang lainnya (buta warna) masih mampu menjalankan aplikasi, tetapi tidak akan optimal.
Height	Tinggi badan minimal 140 cm

3.1.14 Analisis Kebutuhan Fungsional

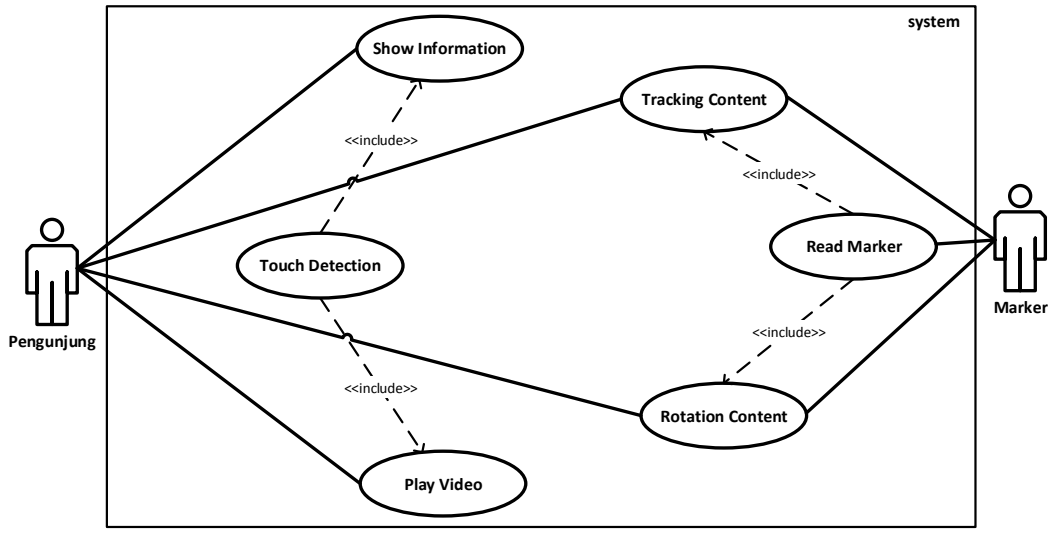
Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem dan menjelaskan kebutuhan yang diperlukan sistem yang dijelaskan pada tabel 3.7.

Tabel 3.7 Kebutuhan Fungsional Sistem

Kode Kebutuhan	Deskripsi Kebutuhan
SKPL-MAC-001	Pengunjung menampilkan menu untuk menampilkan informasi yang disediakan
SKPL-MAC-002	Pengunjung dapat menggeserkan *content* sesuai dengan posisi yang diinginkan.
SKPL-MAC-003	Pengunjung dapat merotasi *content* sesuai dengan sudut pandang yang diinginkan.
SKPL-MAC-004	Pengunjung dapat menampilkan informasi yang diinginkan dengan cara memilih menu terlebih dahulu.
SKPL-MAC-005	Pengunjung dapat memainkan video dari koleksi batuan artefak tersebut

3.1.14.1 Use Case Diagram

Use case diagram digunakan untuk menggambarkan *actor* dengan *use case* yang terdapat dalam sistem. *Use case* terdiri dari tiga bagian yaitu definisi *actor*, definisi *use case*, dan *scenario use case*.



Gambar 3.13 *Use Case* Diagram Aplikasi

3.1.14.2 Definisi Aktor

Definisi *actor* berfungsi untuk mendeskripsikan peran *actor* yang terlibat dalam *use case* ini. Untuk lebih jelasnya dapat dilihat pada tabel 3.8.

Tabel 3.8 Definisi Aktor

Nama Aktor	Deskripsi
Pengunjung	Orang yang berinteraksi dengan sistem atau orang yang mengakses semua fungsi dalam sistem
Maker	*Marker* adalah alat bantu berupa gambar yang digunakan sebagai penanda dimana objek akan ditampilkan.

3.1.14.3 Use Case Scenario dan Activity Diagram

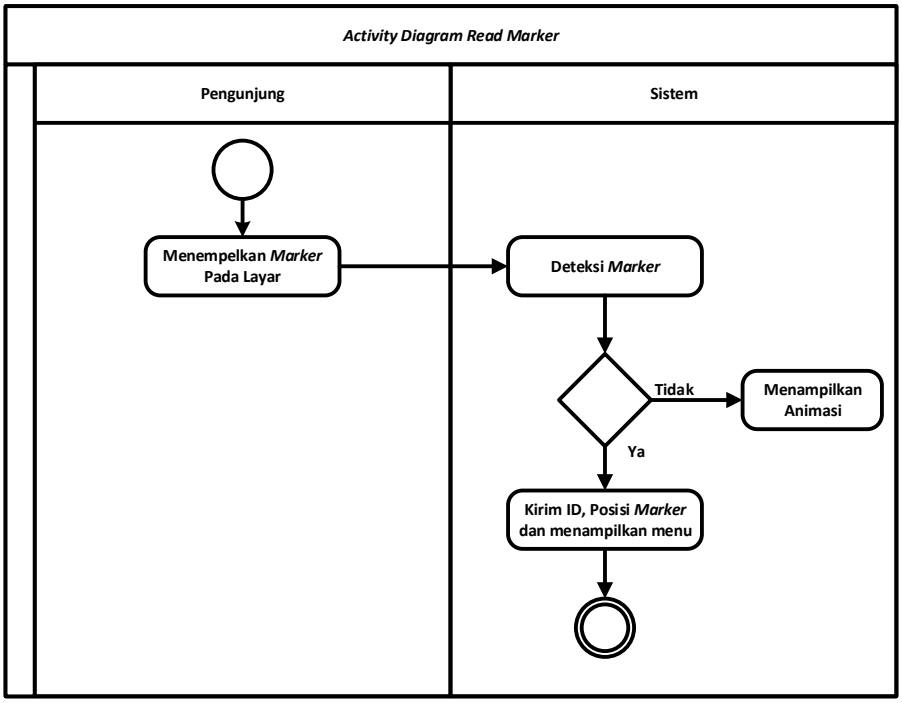
Use case scenario berisi mengenai skenario (*flow of event*) untuk beberapa *use case* utama, yang menggambarkan urutan interaksi *actor*. Berdasarkan *use case diagram* pada gambar 3.13 maka *use case scenario* untuk aplikasi yang akan dibangun akan dijelaskan pada tabel 3.9 sampai dengan tabel 3.14. Untuk memperjelas *use case scenario* juga di sertakan *activity diagram* dari *use case* yang bersangkutan.

1. *Use Case Scenario Read Marker*

Tabel 3.9 *Use Case Scenario Read Marker*

Use Case Name	*Read Marker*	
Related Requirements	SKPL-MAC-001 SKPL-MAC-002 SKPL-MAC-003	
Goal In Context	Sistem Dapat mendeteksi ID, posisi *Marker* dan menampilkan menu	
Preconditions	Sistem menampilkan animasi	
Successful End Condition	Sistem dapat menampilkan menu	
Failed End Condition	Sistem gagal mendeteksi *marker*	
Primary Actors	Pengunjung	
Secondary Actors	*marker*	
Trigger	Menyimpan atau menempelkan *marker* pada layar sentuhan	
Main Flow	**Step**	**Action**
	1	Sistem menampilkan animasi
	2	Pengunjung menyimpan / menempelkan, mengeserkan, merotasi *marker* pada layar
	3	Sistem mendeteksi ID *marker* dan posisi
	4	Sistem mengirimkan ID dan posisi *marker*
	5	Sistem menampilkan menu

Kondisi *default* pada sistem yaitu sistem akan menampilkan animasi, sistem akan bereaksi ketika sistem dapat membaca *marker*. Untuk *Use case scenario read marker* ditambahkan juga *activity diagram read menu* yang dapat dilihat pada gambar 3.14.



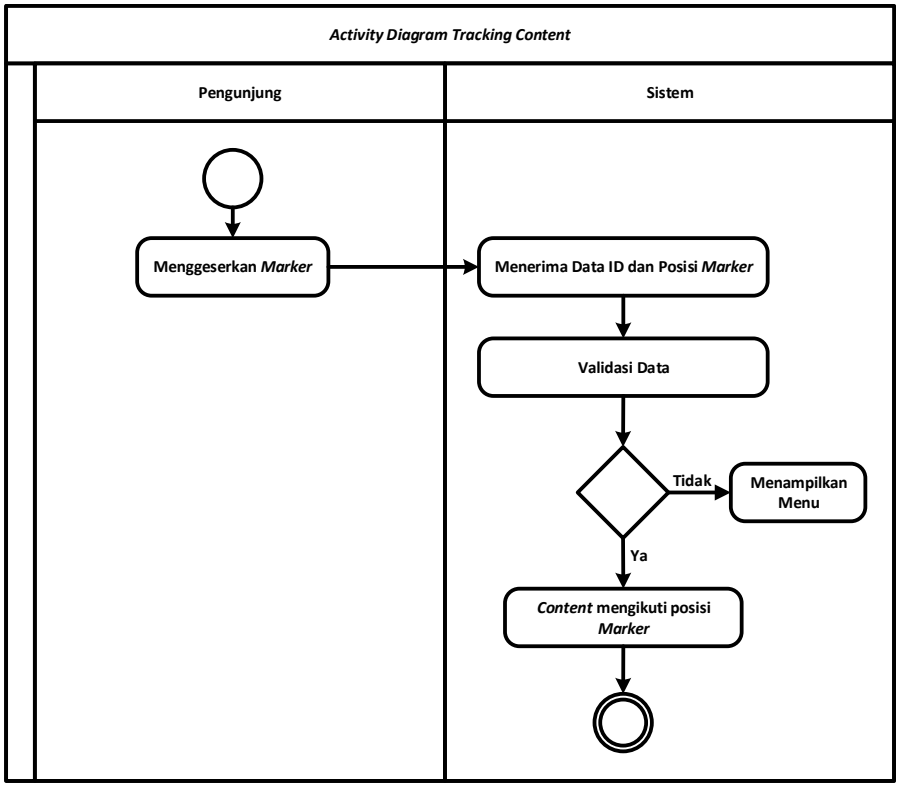
Gambar 3.14 A*ctivity Diagram Read Marker*

2. *Use Case Scenario Tracking Content*

Tabel 3. 10 *Use Case Scenario Tracking Content*

Use Case Name	*Tracking Content*	
Related Requirements	SKPL-MAC-002	
Goal In Context	Pengunjung dapat menggeserkan *content* mengikuti pergerakan *marker*	
Preconditions	Sistem menampilkan menu	
Successful End Condition	Sistem dapat memindahkan *content* mengikuti pergerakan *marker*	
Failed End Condition	Sistem gagal memindahkan *content* mengikuti pergerakan *marker*	
Primary Actors	Pengunjung	
Secondary Actors	*Marker*	
Trigger	Menggeserkan *marker* pada layar	
Main Flow	**Step**	**Action**
	1	Sistem menampilkan menu
	2	Pengunjung menggeserkan *marker* ke posisi yang diinginkan
	3	Sistem membaca pergeseran *marker*
	4	Sistem melakukan validasi posisi *marker*
	5	Sistem memindahkan *content* mengikuti pergerakan *marker*

Untuk memperjelas *use case scenario Tracking content* pada tabel 3.10 ditambahkan juga *activity diagram Tracking content* yang dapat dilihat pada gambar 3.15.



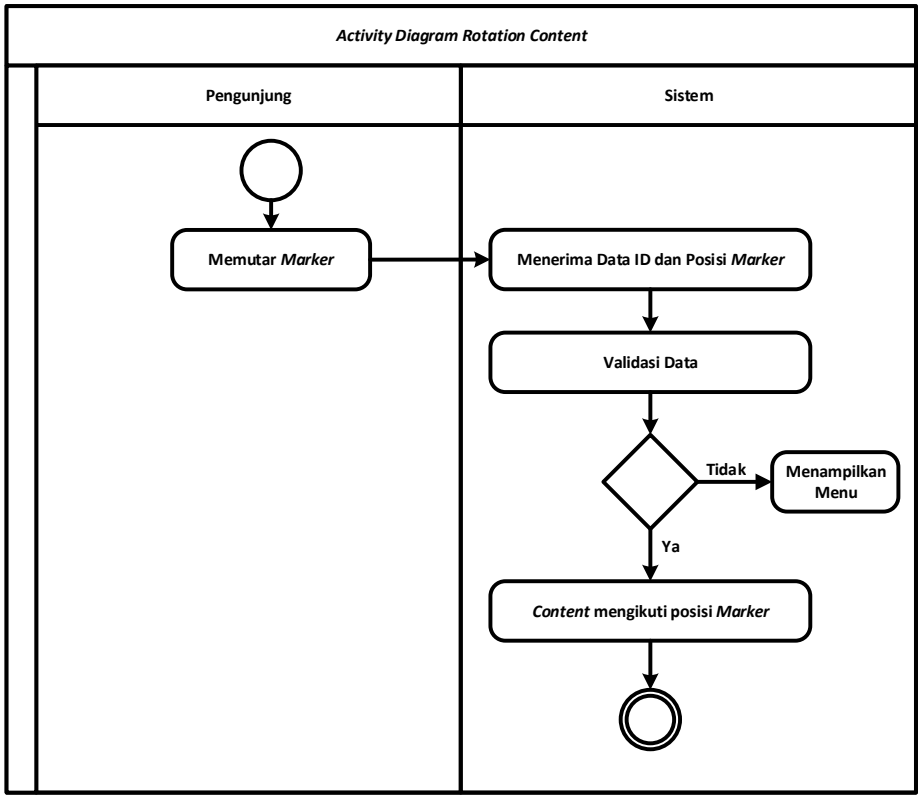
Gambar 3.15 *Activity Diagram Tracking Content*

3. *Use Case Scenario Rotation Content*

Tabel 3.11 *Use Case Scenario Rotation Content*

Use Case Name	*Rotation Content*	
Related Requirements	SKPL-MAC-003	
Goal In Context	Pengunjung dapat memutar *content* mengikuti putaran *marker*	
Preconditions	Sistem menampilkan menu	
Successful End Condition	Sistem dapat memutar *content* mengikuti putaran *marker*	
Failed End Condition	Sistem gagal memutar *content* mengikuti putaran *marker*	
Primary Actors	Pengunjung	
Secondary Actors	*Marker*	
Trigger	Memutar *marker* pada layar	
Main Flow	**Step**	**Action**
	1	Sistem menampilkan menu
	2	Pengunjung memutar *marker* ke posisi yang diinginkan
	3	Sistem membaca putaran *marker*
	4	Sistem melakukan validasi putaran *marker*
	5	Sistem memutar *content* mengikuti putaran *marker*

Untuk memperjelas *use case scenario Rotation content* ditambahkan juga *activity diagram Rotation content* yang dapat dilihat pada gambar 3.16.



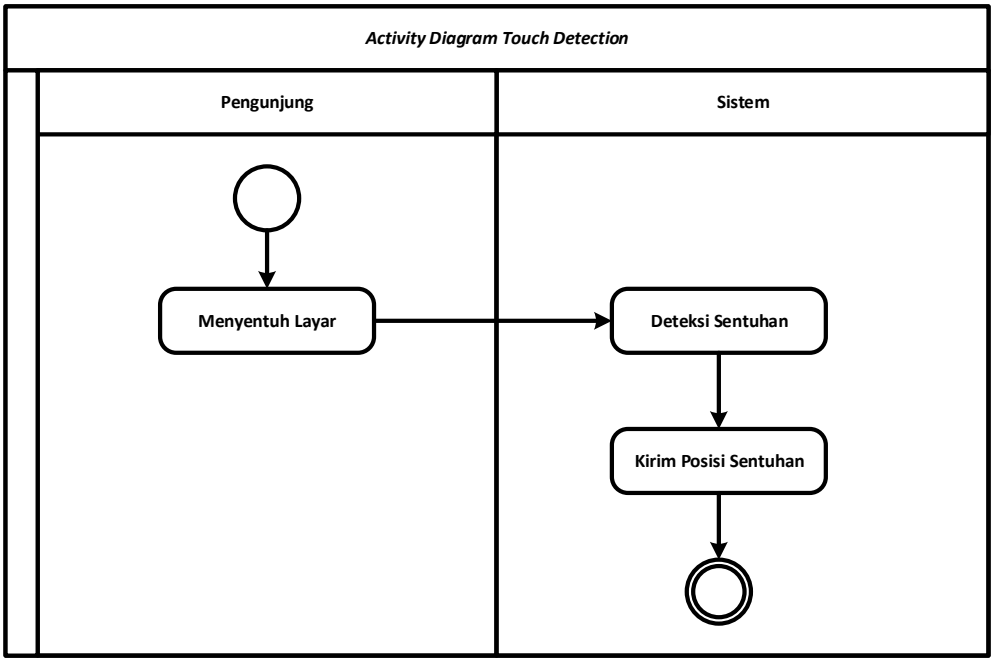
Gambar 3.16 *Activity Diagram Rotation Content*

4. *Use Case Scenario Touch Detection*

Tabel 3.12 *Use Case Scenario Rotation Content*

Use Case Name	*Touch Detection*	
Related Requirements	SKPL-MAC-001 SKPL-MAC-004 SKPL-MAC-005	
Goal In Context	Sistem dapat mendeteksi posisi sentuhan pada layar	
Preconditions	Sistem menampilkan menu	
Successful End Condition	Sistem dapat mendeteksi sentuhan dan sistem berhasil mengirimkan posisi sentuhan	
Failed End Condition	Sistem gagal mendeteksi sentuhan	
Primary Actors	Pengunjung	
Secondary Actors		
Trigger	Pengunjung menyentuh permuhaan layar	
Main Flow	**Step**	**Action**
	1	Sistem menampilkan menu
	2	Pengunjung menyentuh permukaan layar
	3	Sistem mendeteksi posisi sentuhan
	4	Sistem mengirimkan posisi sentuhan

Untuk memperjelas *use case scenario Touch Detection* ditambahkan juga *activity diagram Touch Detection* yang dapat dilihat pada gambar 3.17.



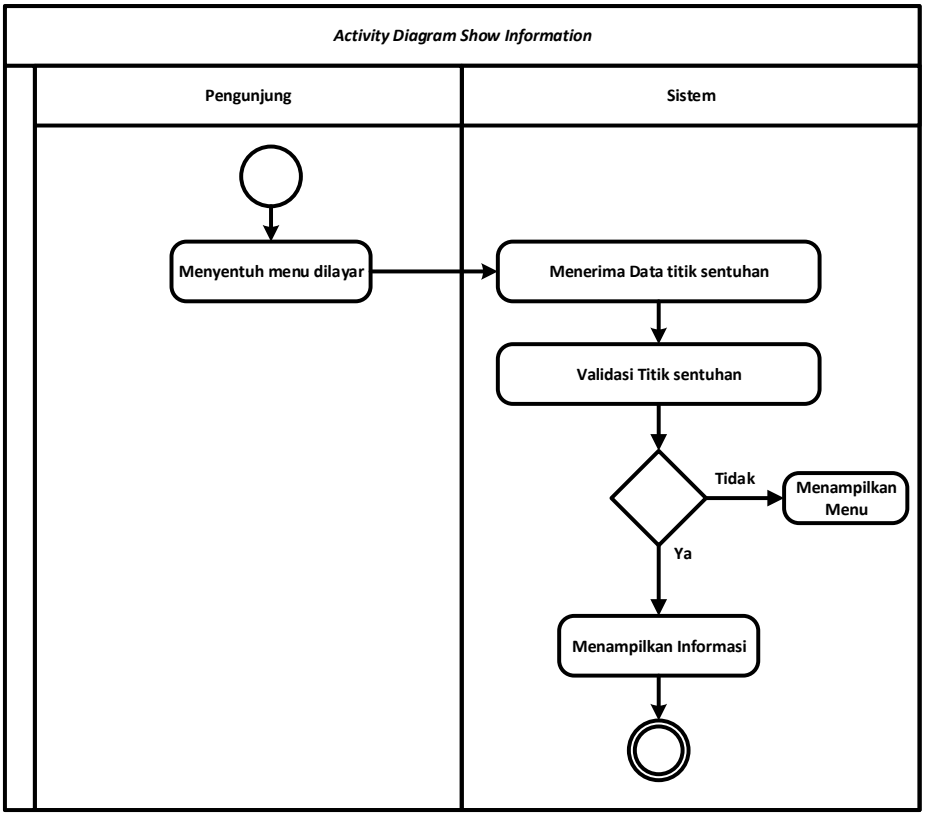
Gambar 3.17 *Activity Diagram Touch Detection*

5. *Use Case Scenario Show Information*

Tabel 3.13 *Use Case Scenario Show Information*

Use Case Name	Show Information	
Related Requirements	SKPL-MAC-004	
Goal In Context	Pengunjung dapat menampilkan informasi	
Preconditions	Sistem menampilkan menu	
Successful End Condition	Sistem dapat menampilkan informasi	
Failed End Condition	Sistem gagal menampilkan informasi	
Primary Actors	Pengunjung	
Secondary Actors		
Trigger	Menyentuh menu yang telah disediakan	
Main Flow	**Step**	**Action**
	1	Sistem menampilkan menu
	2	Pengunjung menyentuh menu yang telah disediakan
	3	Sistem membaca titik sentuh pengunjung
	4	Sistem melakukan validasi
	5	Sistem menampilkan informasi

Untuk memperjelas *use case scenario* menampilkan informasi ditambahkan juga *activity diagram* menampilkan informasi seperti pada gambar 3.18.



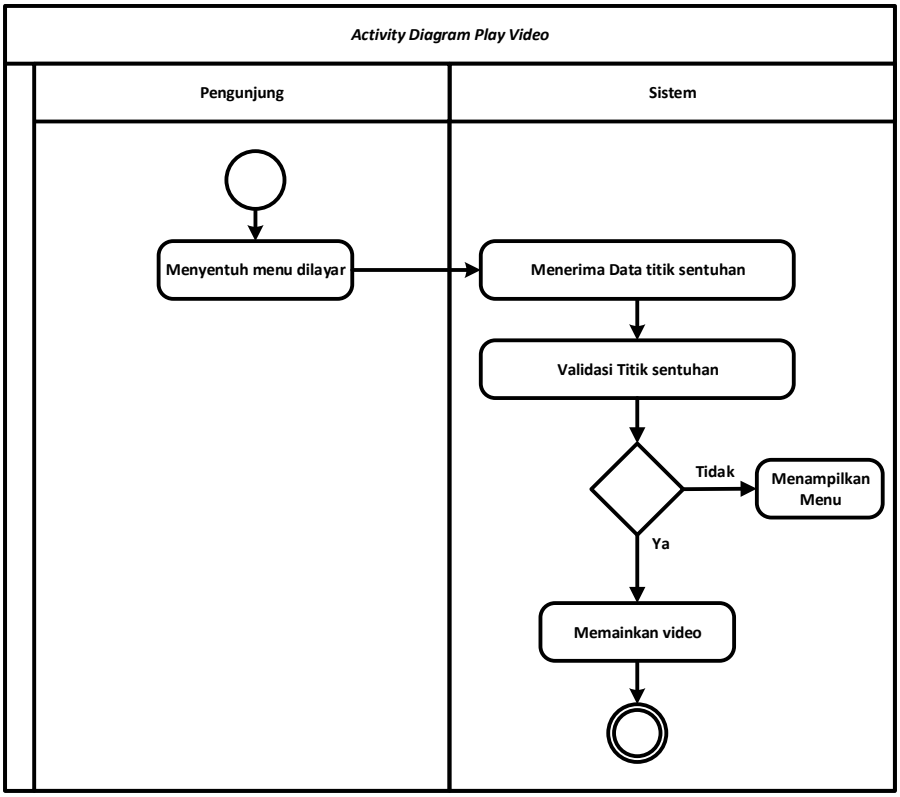
Gambar 3.18 *Activity Diagram Show Information*

6. *Use Case Scenario Play Video*

Tabel 3.14Tabel 3. 14 *Use Case Scenario Play Video*

Use Case Name	Memainkan Video	
Related Requirements	SKPL-MAC-005	
Goal In Context	Pengunjung dapat memainkan video	
Preconditions	Sistem memainkan video	
Successful End Condition	Sistem dapat memainkan video	
Failed End Condition	Sistem gagal memainkan video	
Primary Actors	Pengunjung	
Secondary Actors		
Trigger	Menyentuh menu yang telah disediakan	
Main Flow	**Step**	**Action**
	1	Sistem menampilkan menu
	2	Pengunjung menyentuh menu yang telah disediakan
	3	Sistem membaca titik sentuh pengunjung
	4	Sistem melakukan validasi
	5	Sistem memainkan video

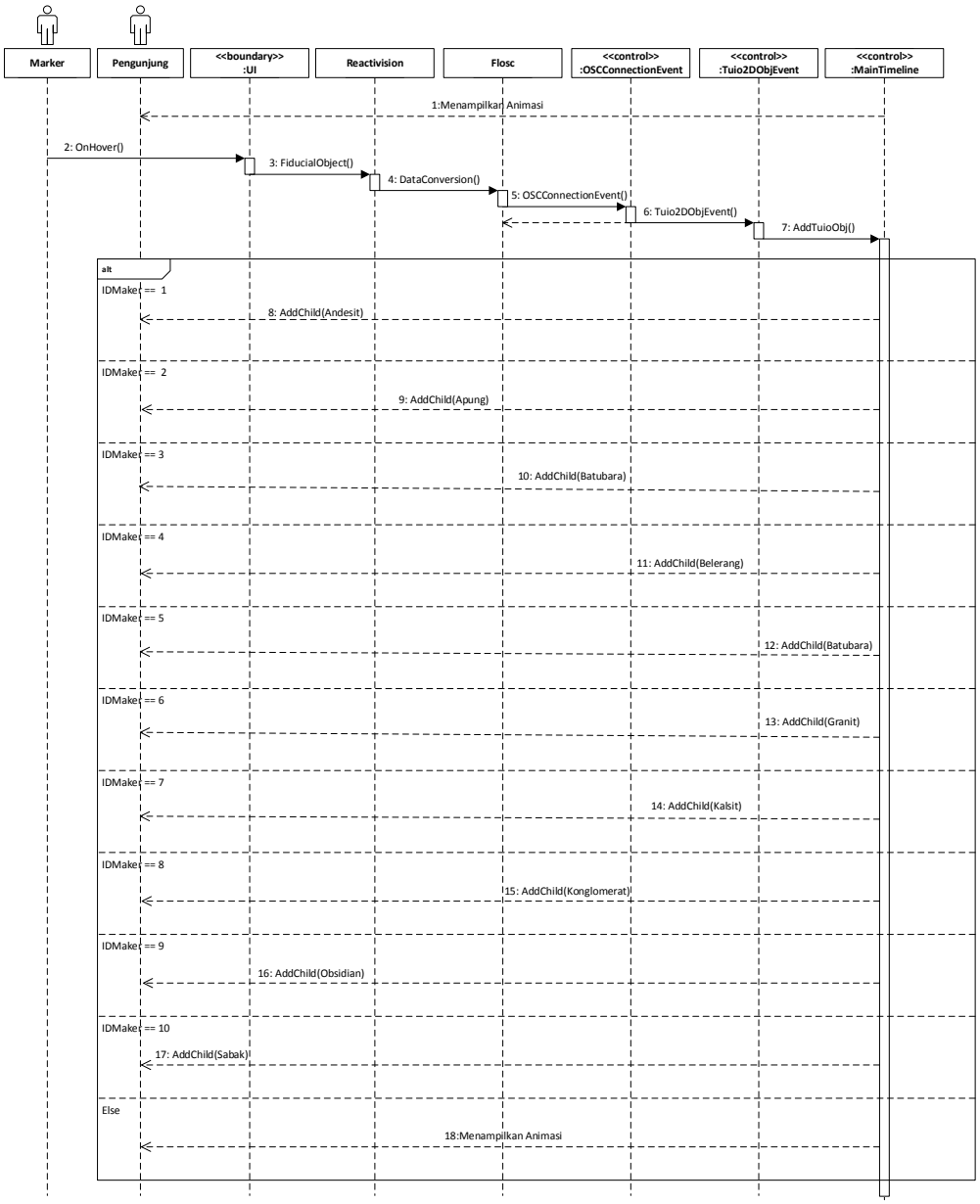
Untuk memperjelas *use case scenario* memainkan video ditambahkan juga *activity diagram* memainkan video seperti pada gambar 3.19.



Gambar 3.19 *Activity Diagram Play Video*

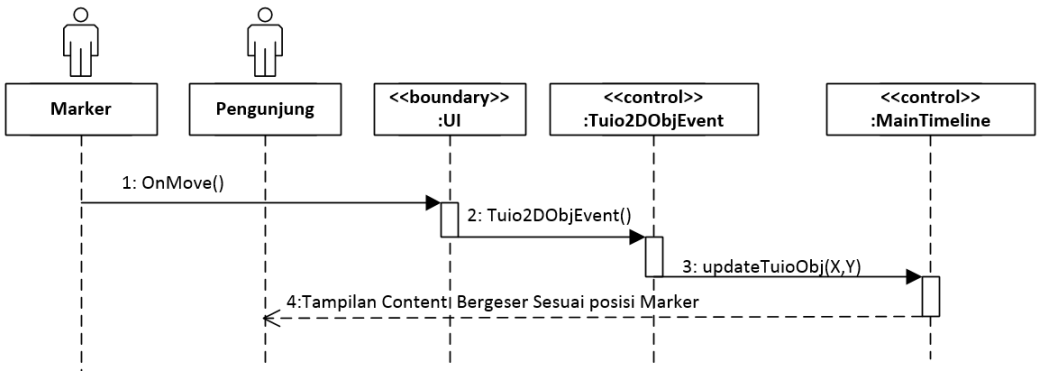
3.1.14.4 Sequence Diagram

Sequence diagram menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini di dalam *use case*. Komponen utama *sequence diagram* terdiri atas obyek yang dituliskan dengan kotak segiempat. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan progres vertikal. Berikut ini adalah *sequence diagram* pada aplikasi *artefact stones*. Pada gambar 2.20 merupakan *sequence diagram* dari aplikasi *artefact stones.*



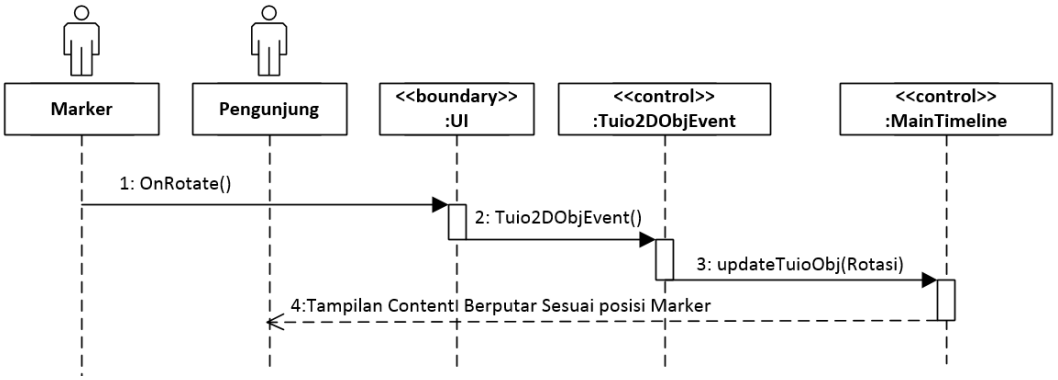
Gambar 3.20 *Sequence Diagram Read Marker*

Sequence diagram diatas menunjukan rangkaian interaksi antar obyek pada proses pembacaan *marker*. Pada gambar 3.21 menunjukan rangkaian interaksi antar obyek pada proses *Tracking Content.*



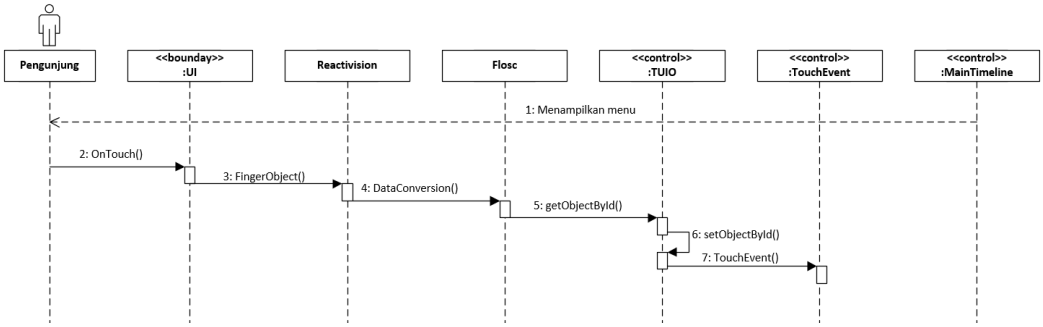
Gambar 3.21 *Sequence Diagram Tracking Content*

Pada gambar 3.22 menunjukan rangkaian interaksi antar obyek pada proses *Rotation Content.*



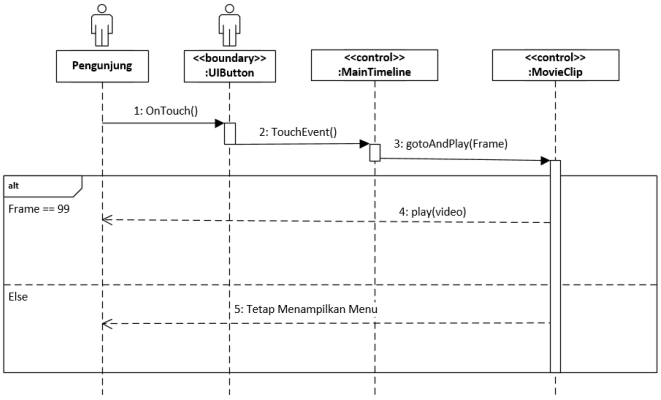
Gambar 3.22 *Sequence Diagram Rotation Content*

Pada gambar 3.23 menunjukan rangkaian interaksi antar obyek pada proses *Touch Detection.*



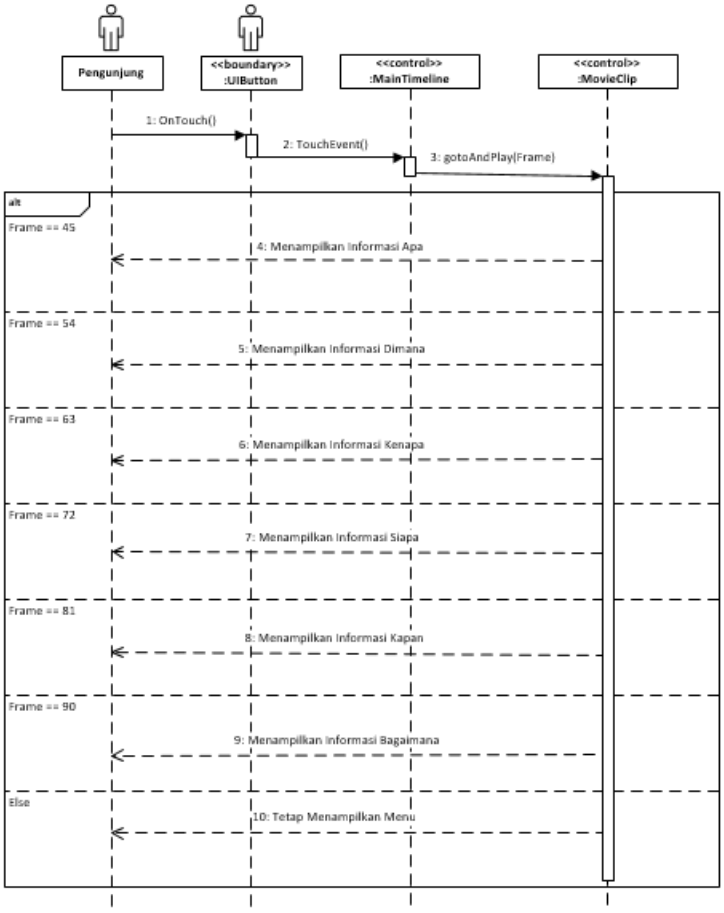
Gambar 3.23 *Sequence Diagram Touch Detection*

Pada gambar 3.24 menggambarkan rangkaian interaksi obyek pada proses *play Video.*



Gambar 3.24 *Sequence Diagram Play Video*

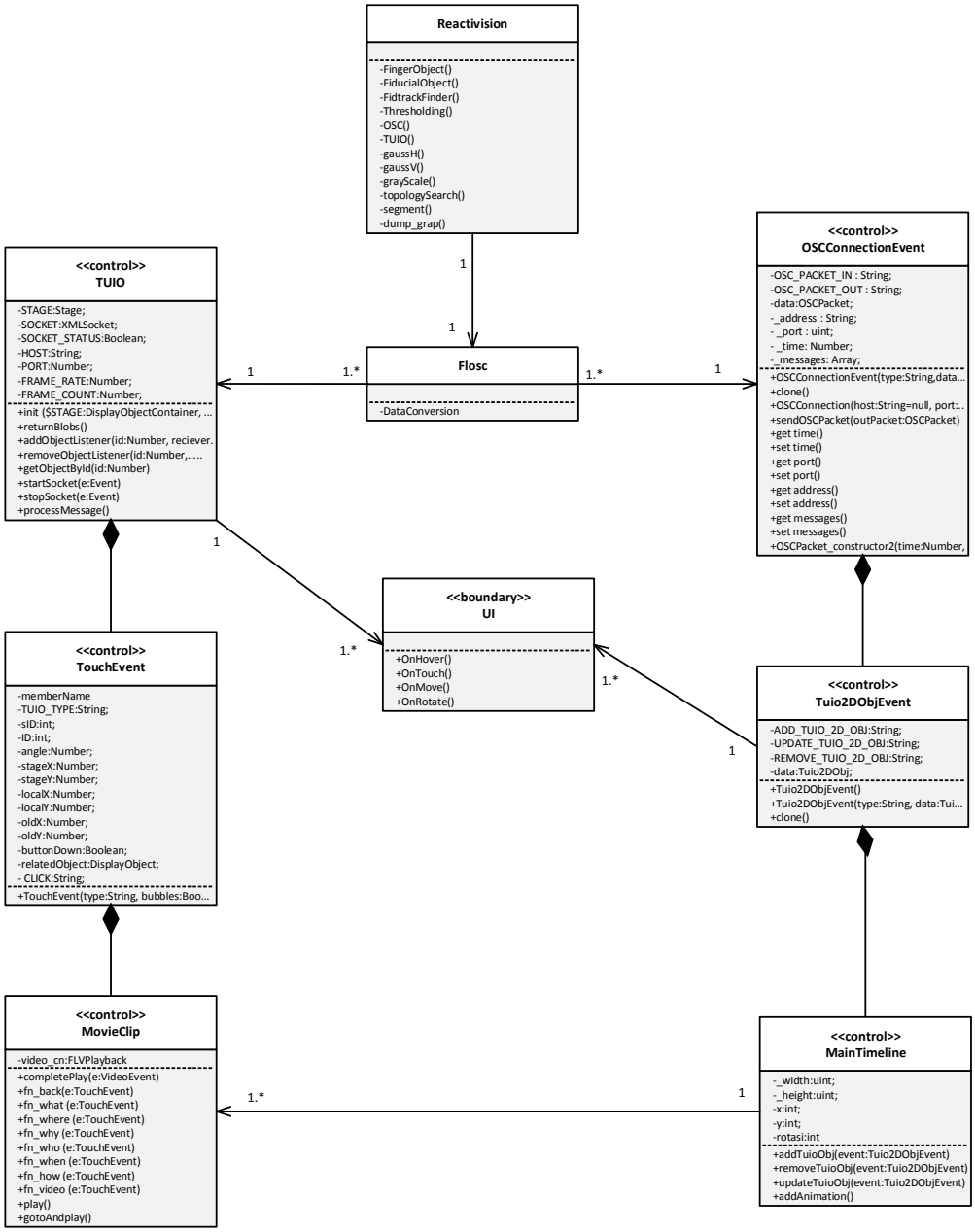
Pada gambar 3.25 menunjukan rangkaian interaksi antar obyek pada proses *Show Information.*



Gambar 3.25 *Sequence Diagram Show Infromation*

3.1.14.5 Class Diagram

Class diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (*attribut* atau *property*) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda atau fungsi). *Class diagram* yang terdapat pada aplikasi *arteface stones* terdapat pada gambar 3.26



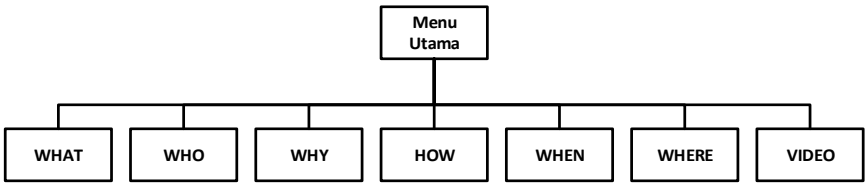
Gambar 3.26 Diagram Kelas Aplikasi

3.2 Perancangan Sistem

Perancangan merupakan penggambaran, perencanaan, dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam suatu kesatuan yang utuh dan berfungsi.

3.2.1 Perancangan Struktur Menu

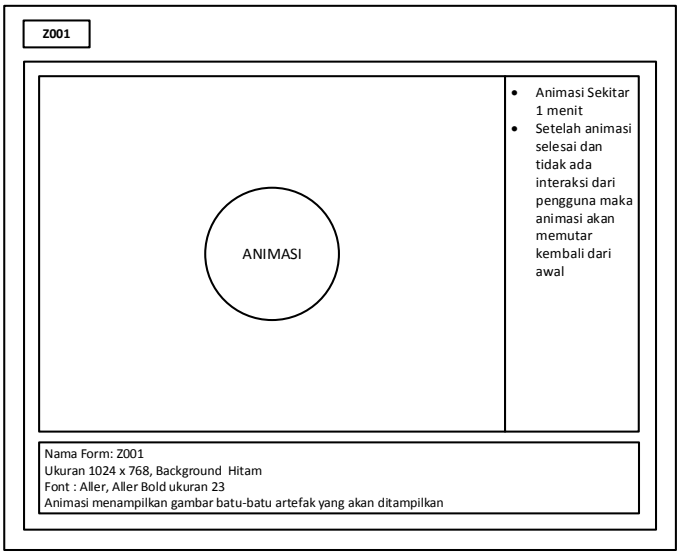
Dalam pembangunan aplikasi diperlukan sebuah struktur menu untuk menunjukan keterkaitan dan tingkatan antara menu utama dengan sub menunya, sehingga pengguna tidak mengalami kesulitan dalam memilih menu-menu yang diinginkan. Perancangan struktur menu aplikasi ini dapat dilihat pada gambar 3.27



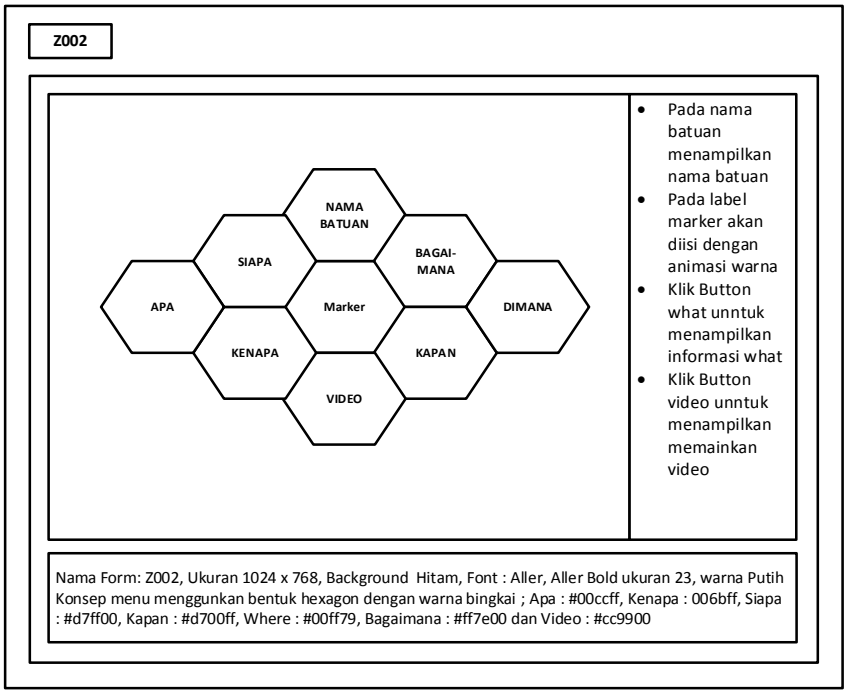
Gambar 3.27 Struktur Menu

3.2.2 Perancangan Antarmuka

Perancangan antarmuka merupakan salah satu bagian penting dalam perancangan sistem karena nantinya antarmuka tersebut akan menjadi fasilitas yang menjembatani interaksi manusia dengan sistem. Gambar 3.28 merupakan tampilan aplikasi pertama kali.



Gambar 3.28 Rencana Animasi

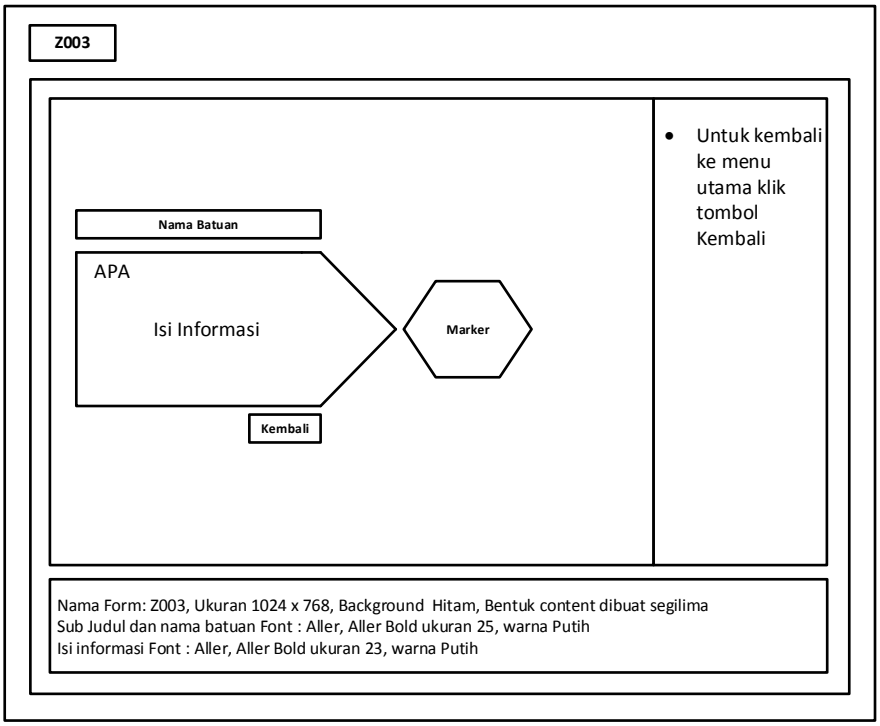


Gambar 3.29 Rencana tampilan menu

Pada gambar 3.29 menggambarkan rencana tampilan menu pada aplikasi, ketika *marker* telah disimpan diatas layar pengguna. Berikut ini penjelasan fungsi tombol yang ada pada menu.

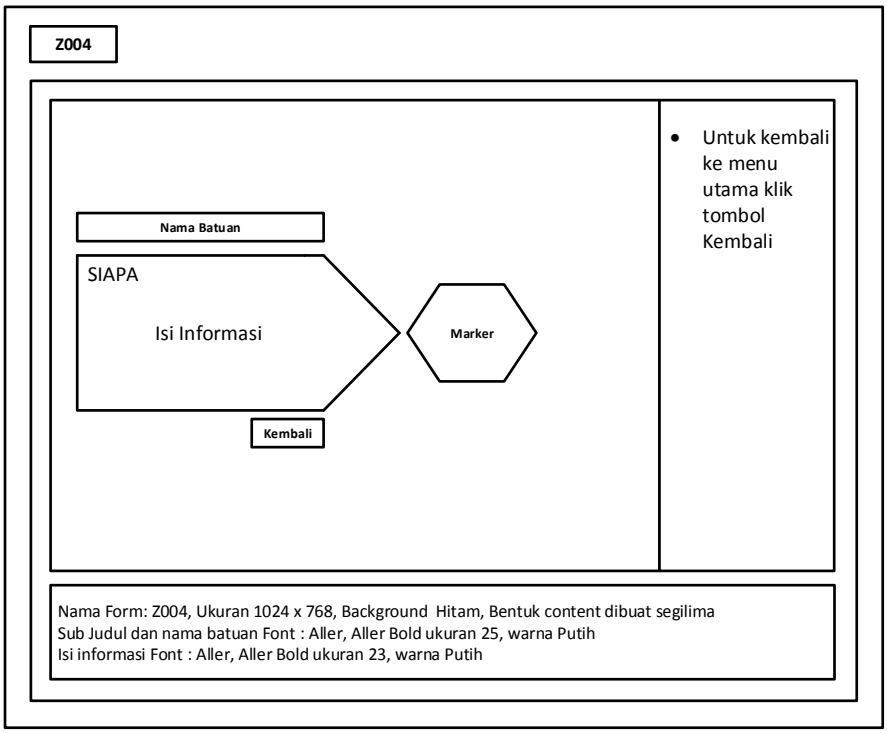
Penjelasan :

a. Tombol Apa : menampilkan informasi menganai apa batuan tersebut
b. Tombol Siapa : menampilkan informasi menganai siapa atau unsur apa saja yang membentuk batuan tersebut
c. Tombol Kenapa : menampilkan informasi menganai mengapa batuan tersebut bisa terbentuk
d. Tombol Kapan: menampilkan informasi menganai kapan batuan tersebut terbentuk
e. Tombol Dimana : menampilkan informasi letak terbentuknya batuan
f. Tombol Bagaimana : menampilkan informasi bagaimana proses terbentuknya batuan tersebut
g. Tombol Video : menampilkan video informasi mengenai batuan tersebut

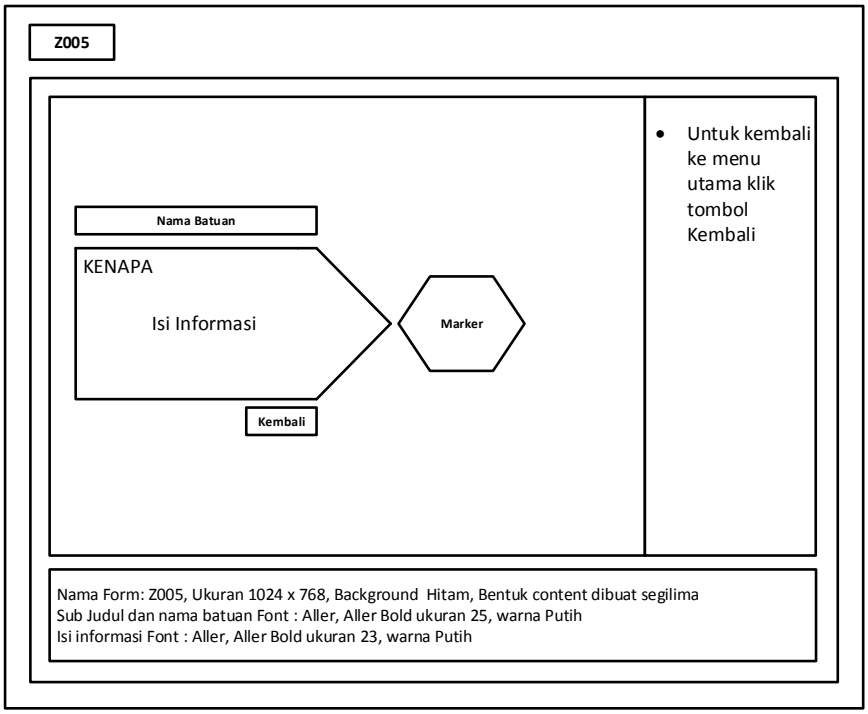


Gambar 3.30 Rencana tampilan menu Apa

Pada gambar 3.30 menggambarkan rencana tampilan menu Apa dan gambar 3.31 menggambarkan rencana tampilan menu Siapa pada aplikasi.

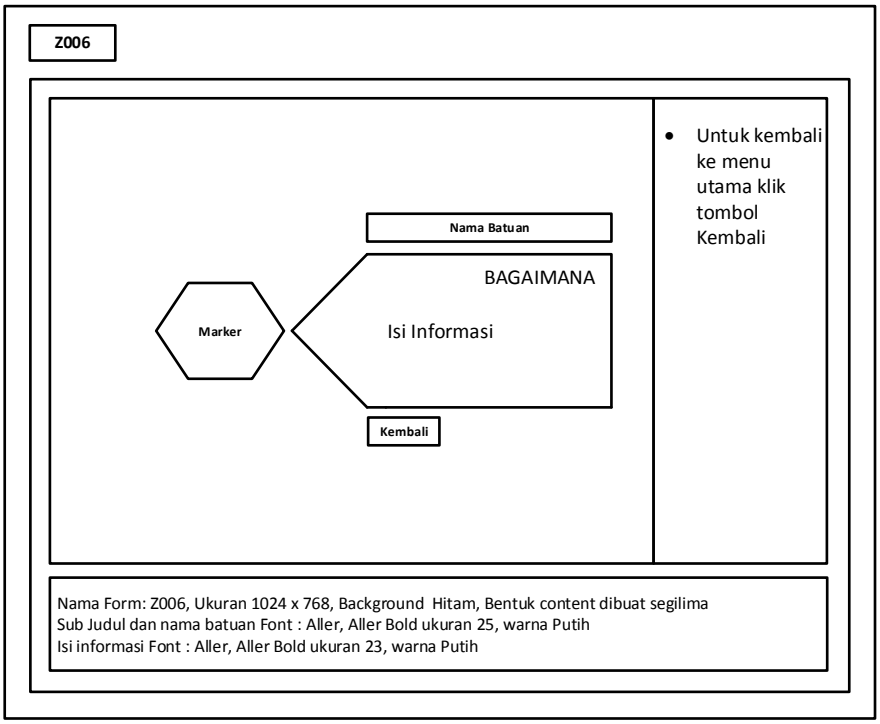


Gambar 3.31 Rencana tampilan menu Siapa

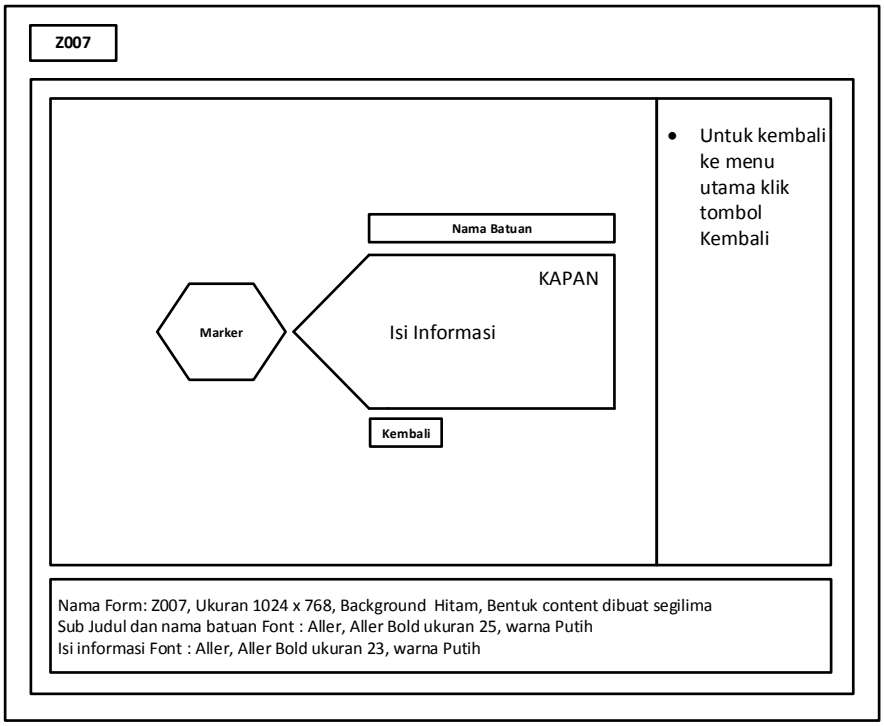


Gambar 3.32 Rencana tampilan menu Kenapa

Pada gambar 3.32 menggambarkan rencana tampilan menu Kenapa dan gambar 3.33 menggambarkan rencana tampilan menu Bagaimana pada aplikasi.

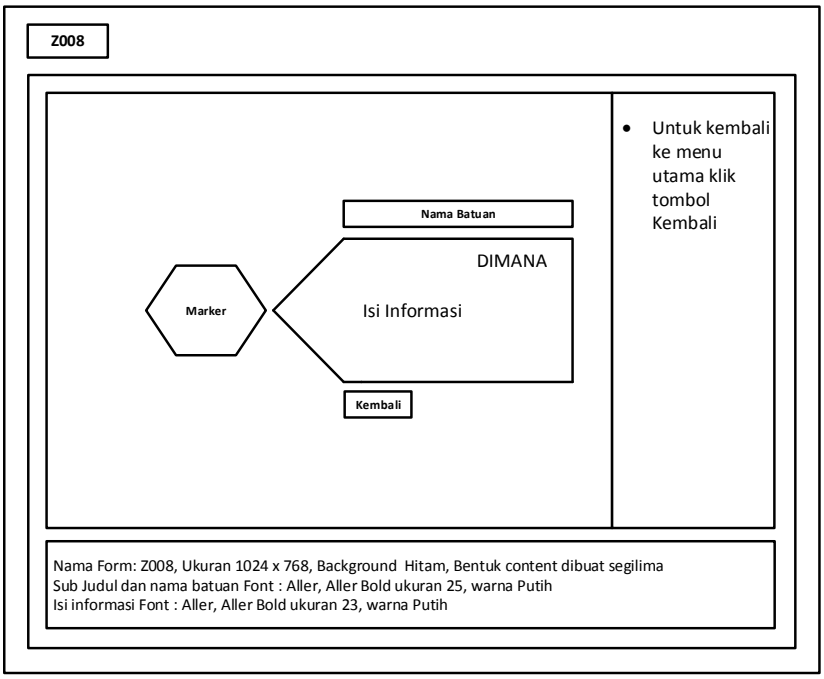


Gambar 3.33 Rencana tampilan menu Bagaimana

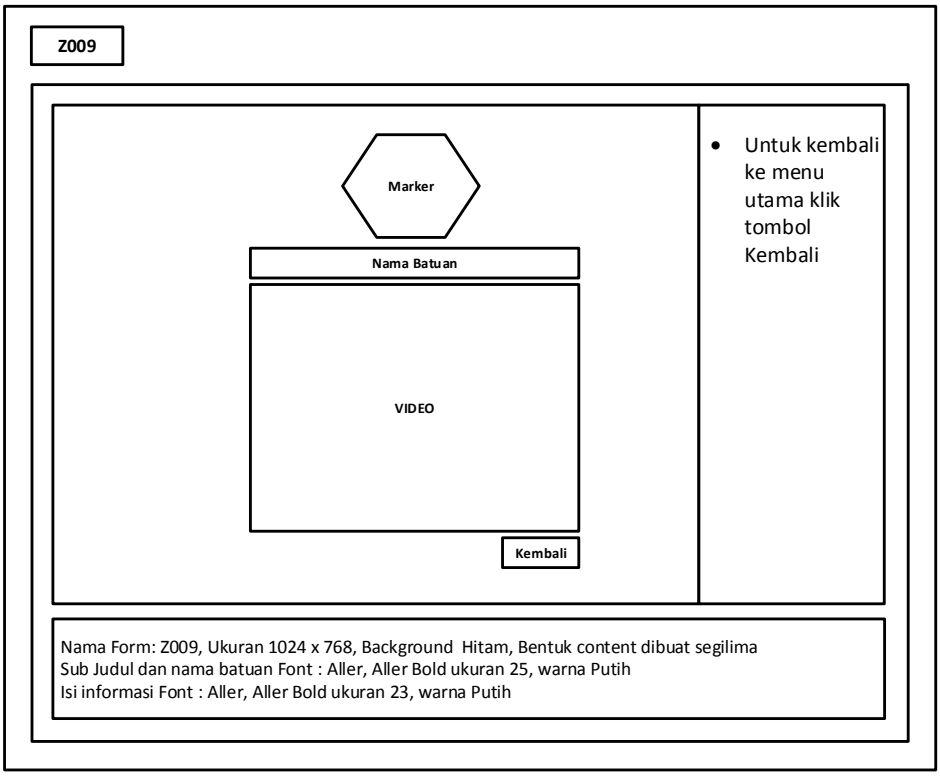


Gambar 3.34 Rencana tampilan menu Kapan

Pada gambar 3.34 menggambarkan rencana tampilan menu Kapan dan gambar 3.35 menggambarkan rencana tampilan menu Dimana pada aplikasi.



Gambar 3.35 Rencana tampilan menu Dimana



Gambar 3.36 Rencana tampilan menu video

Pada gambar 3.36 perencanaan video, ukuran video tidak ditampilkan secara *fullscreen*. Ukuran video yang ditampilkan diperkecil agar dapat menampilkan video lain di koleksi batuan artefak lainya. Selain itu video juga di *setting* pada *mode mute*, agar suara yang dikeluarkan video tidak bercampur dengan video yang lain, sehingga pengguna dapat memutar video dari setiap koleksi batuan artefak secara bersamaan.

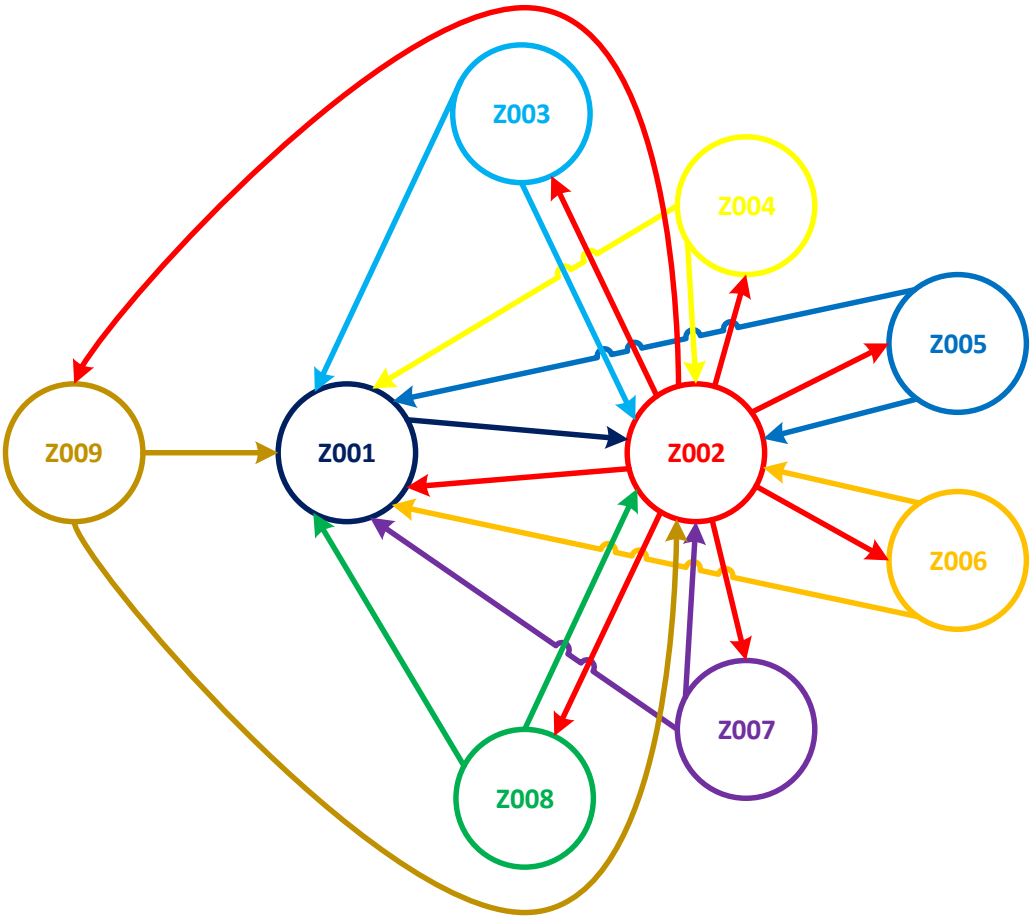
3.2.3 Jaringan Semantik

Jaringan semantik merupakan gambaran grafis yang menunjukan antar obyek (lingkaran) dan garis yang menggambarkan formasi antar obyek tersebut. Adapun jaringan semantik dari aplikasi dapat dilihat pada gambar 3.37.

Adapun keterangannya adalah sebagai berikut:

a. Z001 : merupakan antarmuka untuk tampilan animasi *opening* dan *idle*
b. Z002 : merupakan antarmuka untuk tampilan menu utama.
c. Z003 : merupakan antarmuka untuk tampilan informasi Apa
d. Z004 : merupakan antarmuka untuk tampilan informasi Siapa

e. Z005 : merupakan antarmuka untuk tampilan informasi Kenapa
f. Z006 : merupakan antarmuka untuk tampilan informasi Bagaimana
g. Z007 : merupakan antarmuka untuk tampilan informasi Kapan
h. Z008 : merupakan antarmuka untuk tampilan informasi Dimana
i. Z009 : merupakan antarmuka untuk tampilan video batuan



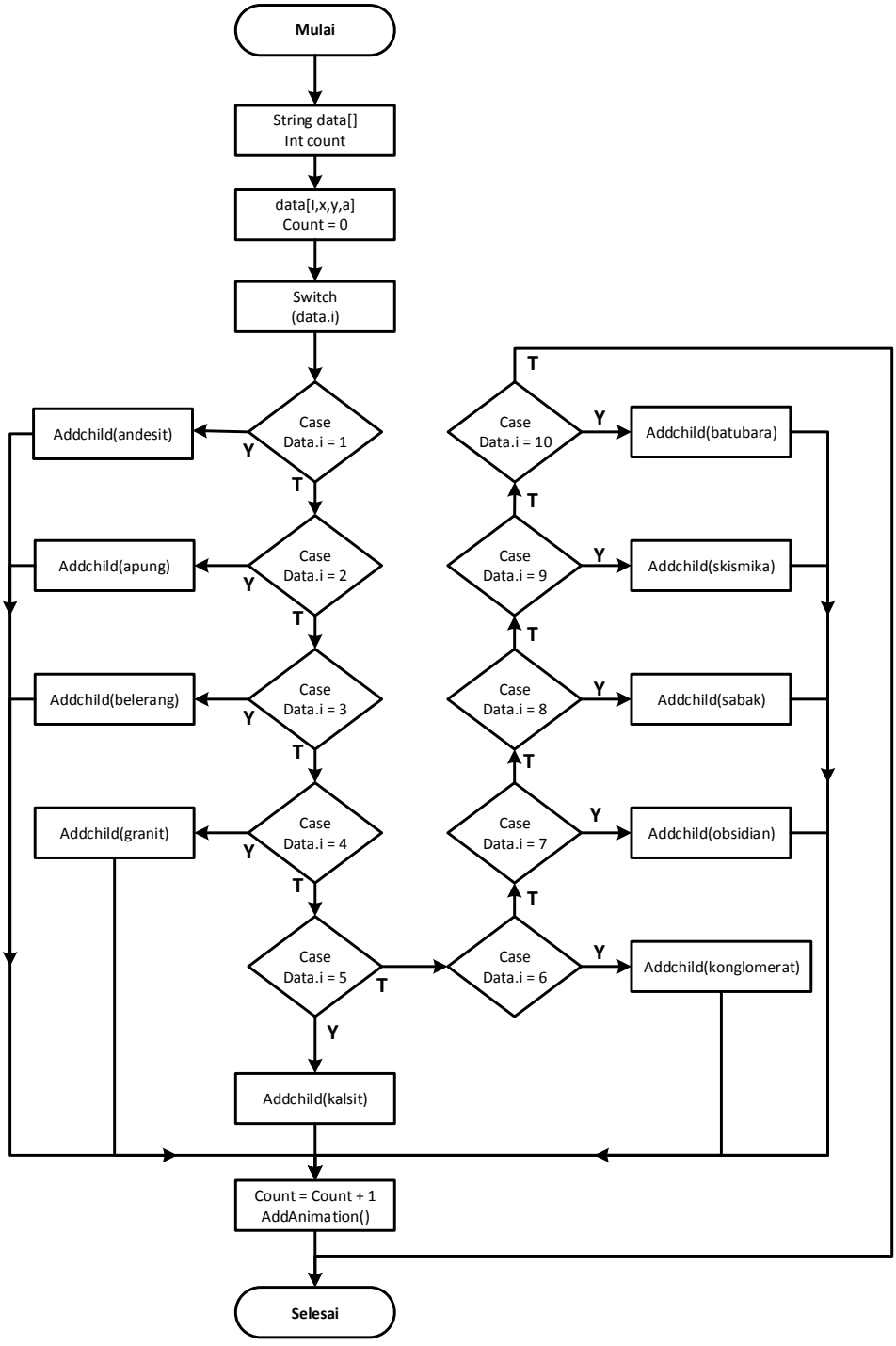
Gambar 3.37 Jaringan Semantik

3.2.4 Perancangan Method

Perancangan *method* merupakan perancangan yang berfungsi untuk mendeskripsikan *method* yang berada di dalam aplikasi *artefact stones*. Adapun perancangan *method* yang digunakan dalam aplikasi *artefact stones*.

1. Perancangan *method* Maintimeline.addTuioObj()

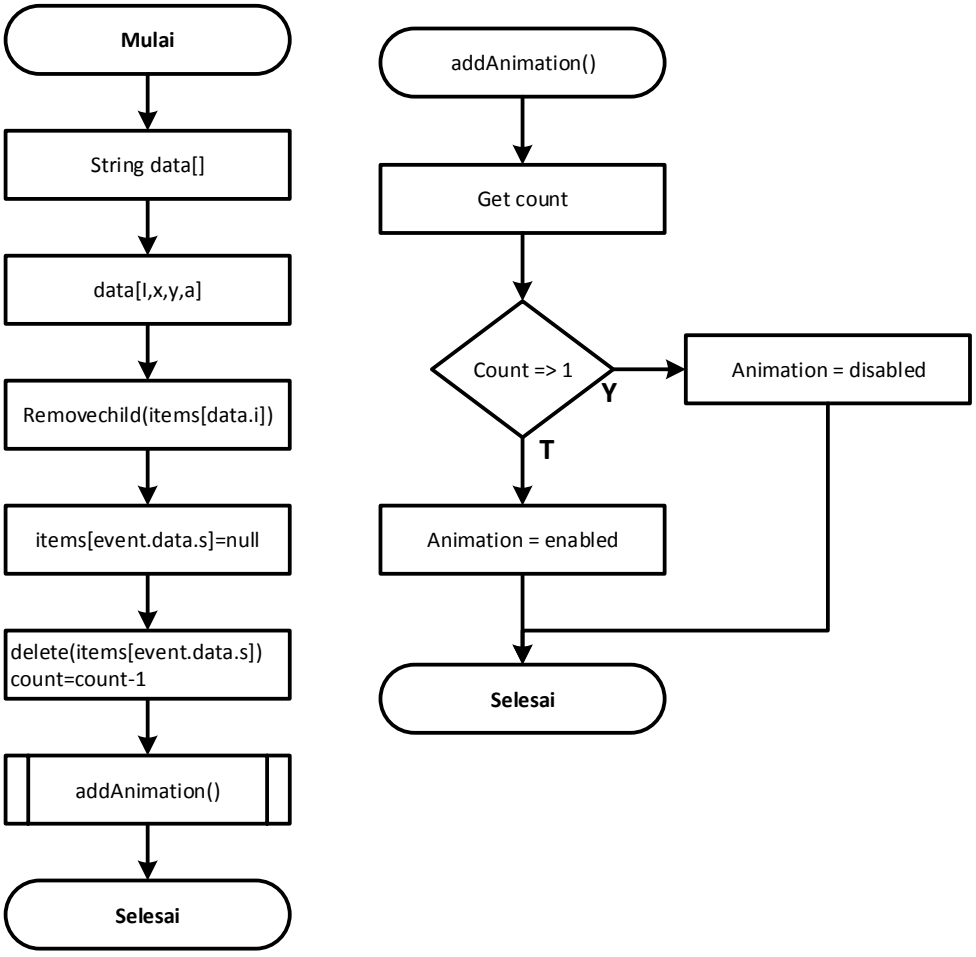
   Gambar 3.38 merupakan *method* yang dijalankan untuk menampilkan *content* batuan atau menu berdasarkan ID pada *marker* yang terbaca.



Gambar 3.38 Flowchart AddTuioObj

2. Perancangan *method* Maintimeline.RemoveTuioObj()

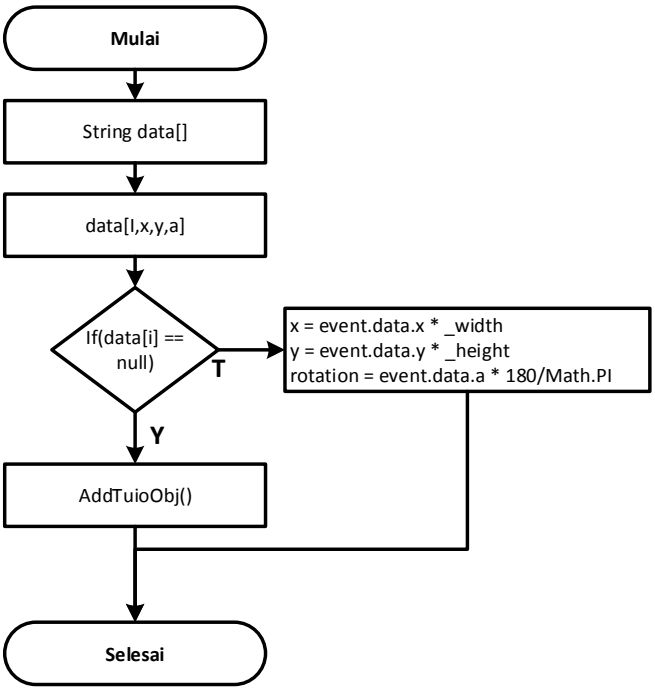
   RemoveTuioObj merupakan *method* yang dijalankan untuk menghapus *content* atau menu yang ditampilkan dilayar sentuhan. Ketika *marker* tidak terbaca maka sistem menghapus content berdasarkan ID *marker*, jika tidak ada *marker* yang terbaca maka sistem akan menampilkan animasi seperti pada gambar 4.39.



Gambar 3.39 Flowchart RemoveTuioObj

3. Perancangan *method* Maintimeline.UpdateTuioObj()

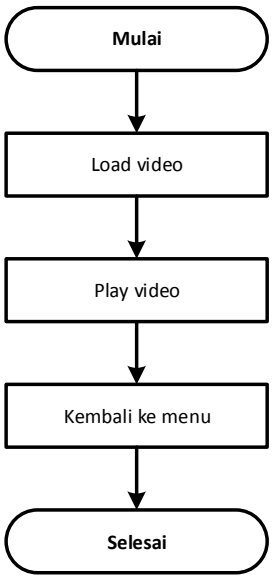
   UpdateTuioObj merupakan *method* yang dijalankan untuk meng-*update* posisi *content* atau menu yang ditampilkan dilayar sentuhan. *Flowchart* dari *method* UpdateTuioObj seperti pada gambar 4.40.



Gambar 3.40 Flowchart UpdateTuioObj

4. Perancangan *method* MovieClip.CompletePlay()

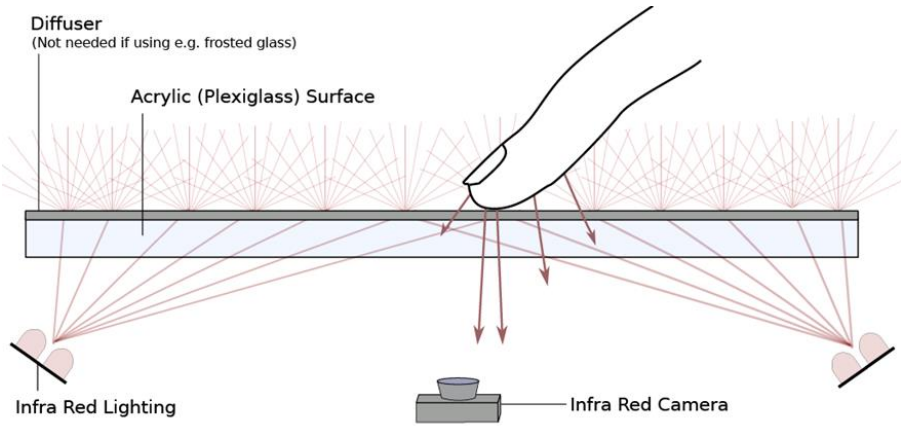
   CompletePlay merupakan *method* yang dijalankan untuk menjalankan video dari *content* batu artefak yang ditampilkan dilayar sentuhan. Setelah video selesai dijalankan maka sistem akan kembali ke menu utama. *Flowchart* dari *method* CompletePlay seperti pada gambar 4.41.



Gambar 3.41 Flowchart CompletePlay

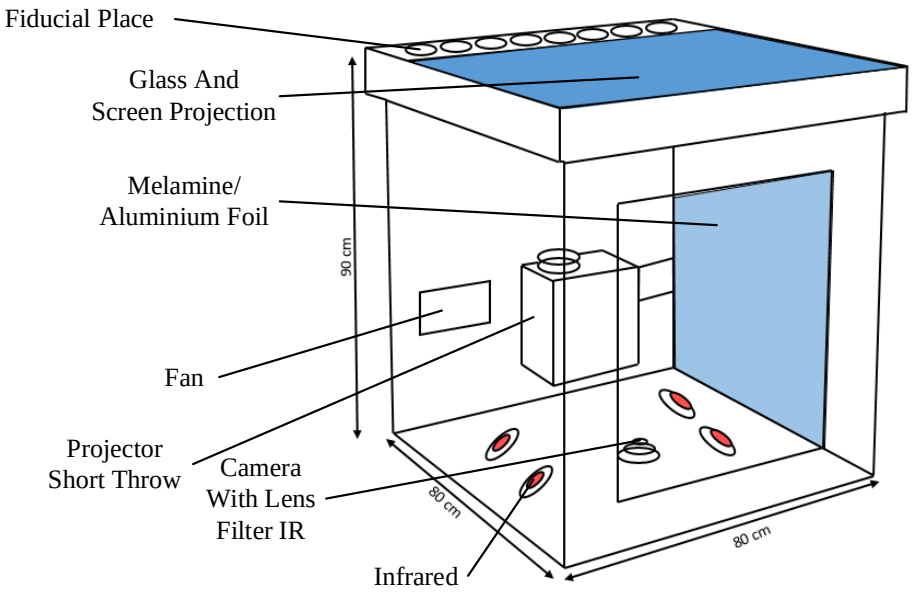
3.2.5 Rancangan Multitouch Menggunakan Teknik Rear-DI

Metode yang digunakan dalam pembuatan perangkat keras pada tugas akhir ini menggunakan metode *Rear Diffused Illumination* (DI) dimana Infra merah diletakan dibelakang layar sentuhan dan diarahkan ke layar permukaan sentuhan. Seperti pada gambar 3.42.



Gambar 3.42 Konsep Metode Rear DI

Laser infra merah digunakan untuk menguatkan pendeteksian cahaya sentuh yang akan ditangkap oleh kamera infra merah. Pada gambar berikut ini merupakan komponen yang digunakan dan rencana bentuk perangkat keras yang akan dibuat.

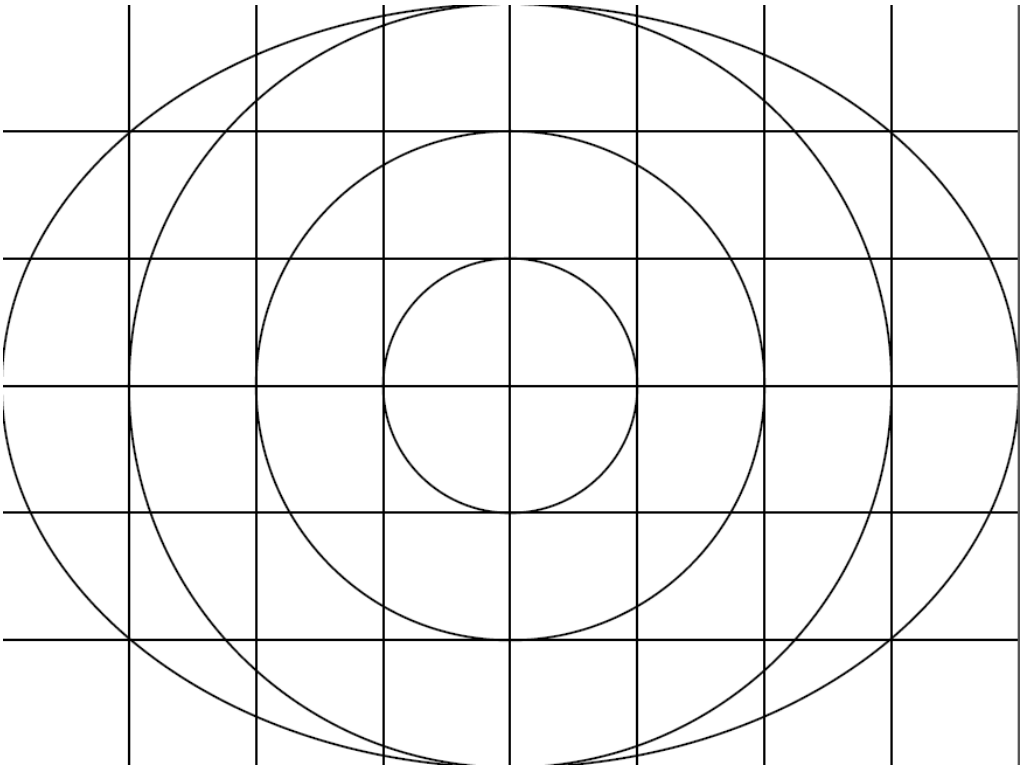


Gambar 3.43 Konsep Perangkat Keras

Proyektor digunakan untuk menampilkan aplikasi ke layar sentuhan (*surface*). Proyektor ditempatkan tepat dibelakang layar sentuhan atau dengan *mode rear table. Fiducial Place* merupakan tempat penyimpan *marker fiducial* yang tidak digunakan. Gambar 3.43 merupakan konsep perangkat keras yang akan dibangun.

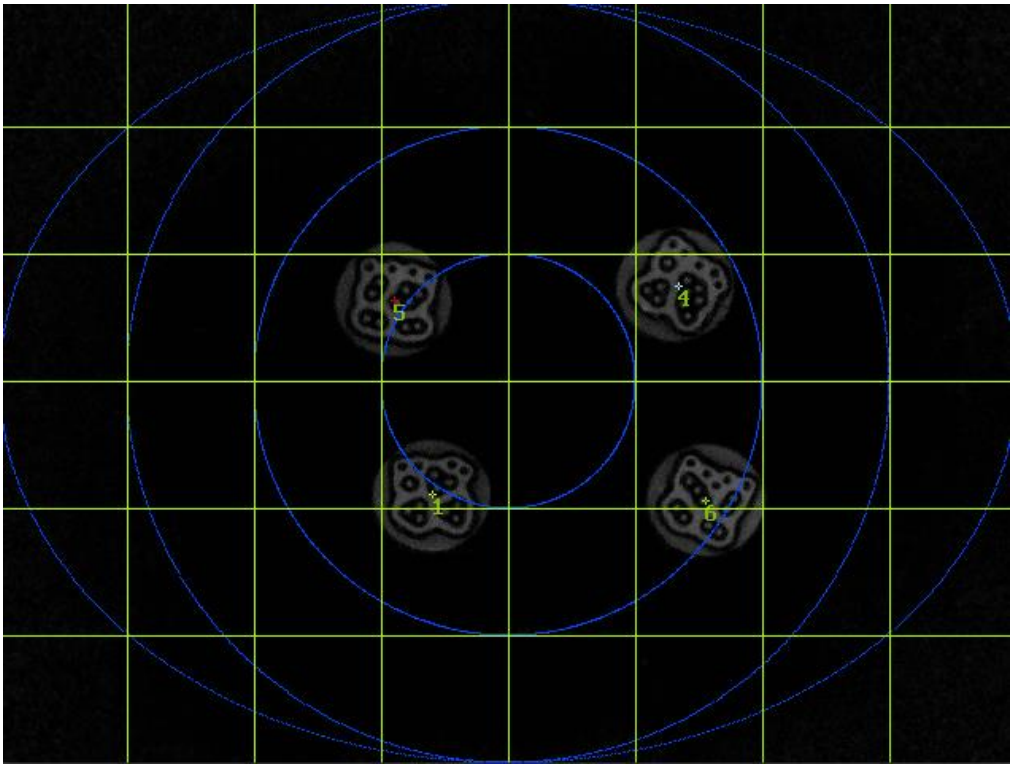
3.2.6 Kalibrasi Reactivision

Kalibrasi pada perangkat lunak Reactivision cukup mudah, langkah pertama cetak dalam kertas gambar seperti pada gambar 3.44.



Gambar 3.44 *Rectangle Calibration*

Pada saat mencetak kertas kalibrasi, cetak dengan ukuran layar sentuhan atau disesuaikan dengan ukuran layar sentuhan. *Reactivision* menyediakan 3 jenis bentuk layar kalibrasi. Jika layar sentuhan berbentuk lingkaran gunakan *square calibration*, jika layar sentuhan berbentuk *wide*, gunakan yang *wide calibration*. Lalu tekan "C" pada *keyboard* untuk memulai kalibrasi, maka akan tampil gambar seperti pada gambar 3.45.



Gambar 3.45 *Grid Calibration*

Setelah itu letakan kertas kalibrasi yang telah dicetak diatas layar permukaan sentuhan, lalu geserkan *grid calibration* pada *reactivion* sesuaikan dengan kertas kalibrasi yang telah dicetak. Gunakan tombol pada *keyboard*, tekan A untuk menggeser ke kiri, D untuk menggeser ke kanan,W untuk menggeser ke atas dan X untuk menggeser ke bawah. Setelah posisi *grid calibration* pada *Reactivision* sama dengan kertas kalibrasi tekan K pada *keyboard* untuk menyimpan hasil kalibrasi.