Department of Computer Science and Engineering

The Chinese University of Hong Kong

# Final Year Project 2011 (1st Term)

## LYU1103

**Project Title:** i.Digi.T.able - Digital Interactive Game Interface Table Apps for iPad

**Student:** Ng Ka Hung (1009615714) khng9@cse.cuhk.edu.hk

Chan Hing Faat (1009618344) hfchan9@cse.cuhk.edu.hk

**Supervisor:** **Professor Michael R. Lyu**

< This is a blank page >

# Abstract

In this report, we are going to describe the ideas we have and document our journey of working on the Final Year Project in this semester (2011 Fall).

Our project goal is to implement an Augmented Reality (AR) game on iPad which let multi-users battle via network.

This report starts with the introduction to our project. We first introduce the technologies we based on and some basic thoughts about why we are doing this project in the Objective section.

We will then discuss about techniques we have investigate that could help and methodologies we have used in the project. For the AR engine part, we will justify the functionalities of different SDKs and analyze them in detail. The network connection method will also be discussed.

Then we will cover the design of our system. In this part, we will present the architecture and implementation plan for our project. Each module will be discussed in detail.

In the upcoming experiment part, we will focus on how we try out new techniques to assist our work in the project. We also tested the tools and see potential room of improve in that. Difficulties and challenges we are currently facing will also be mentioned.

In the conclusion part, our progress for the project is recorded. The next goal then is set in order to guide our further work in the next semester.

# Contents

# Chapter 1. Introduction

## 1.1 Motivation

The previous project **i.Digi.T.able** (supervised by Prof. R Michael Lyu) implements a **Digital Interactive Game Interface Table** using plasma display monitors and cameras to create a shared gaming platform over internet. The previous version of i.Digi.T.able enables users to play chess games and mini board games (e.g. Go chess, Chinese chess, Uno) in different place using real chess/game setup. The video image of the chess board is recorded and sent over the Internet. Hence creates a platform for users in different room to play in the same game interacting with the real object.

When we review the above project, we are inspired by how multi-user can interact “within a same platform while they are in different place” . We also found that the concept of “combining reality and computer graphics” to enhance user experience in the Augment Reality projects very interesting. Hence, we came up the raw idea of the project.

Augmented Reality (AR) has been introduced to computer industry since 1950s. However, due to limitation of hardware and low efficiency of related algorithm, its developmental progress was so slow. Until recent years, AR becomes popular especially on mobile devices and has a wide variety of application. Adding AR element to games can easily enhance the realism and impressiveness.

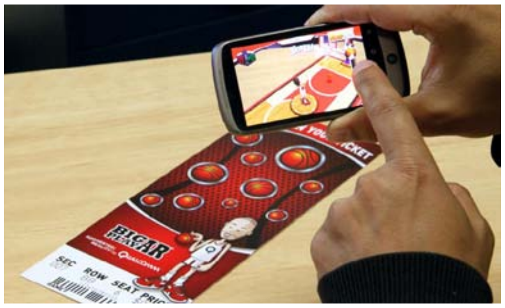


**Fig.1.1.1 Example - Hoops AR**

There are some outstanding AR games available on the App Store. Like the Hoops AR, it is a game played with a virtual basketball court superimposed on the location of the predefined marker i.e. the ticket.

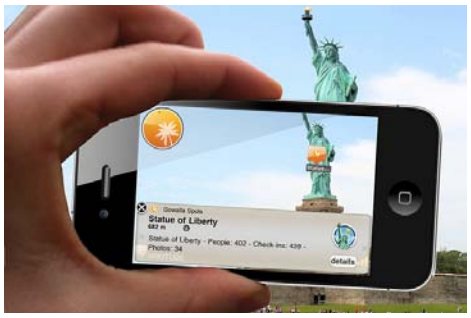


**Fig.1.1.2 Example – Wikitude World Browser**

Another example is ***Wikitude World Browser.*** It makes use of GPS, accelerometer and digital compass. By detecting surroundings, users can access information on the augmented layer instantly.

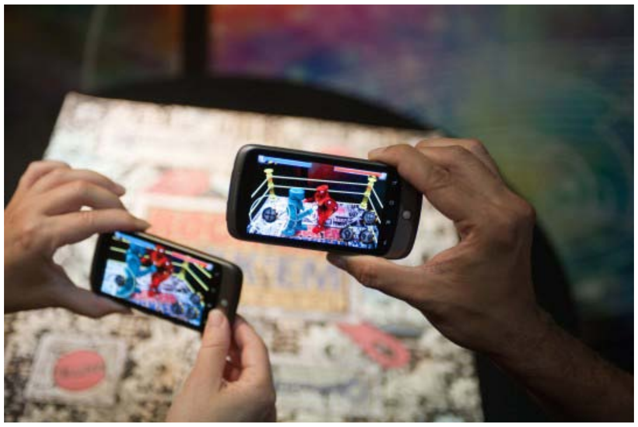


**Fig.1.1.3 Rock 'Em sock 'Em Robots is an AR game example**

We observe that multi-player AR games are so rare on the market. One of the real cases is the ***Rock 'Em Sock 'Em Robots*** made corporately by Mattel and Qualcomm. However, this game has made not much difference from a single-player game since it requires user to use the same piece of marker.

We wish to take the relative position of device to the marker into consideration. In this way, we can construct a virtual space between two devices. Each device can view object respect to its relative position in the virtual space. It is also possible to view another device's virtual position and interact with it.

## 1.2 Background

Despite the rise of AR applications, there are only a few mature software libraries which provide well developed tool kit. Two of them are the **Qualcom** and **String** AR SDKs.

The Qualcomm AR SDK utilizes computer vision technology to tightly align graphics with underlying objects and features support for image targets, frame markers, virtual buttons and simple 3D objects. It is open-source and available for both Android and iOS platforms. The String AR SDK provides less in the aspect of functionality and its extendability is very limited for free license.

Below is a brief comparison between two:

| | **Qualcomm AR** | **String AR** |
|---|---|---|
| License | Free | Free for limited version |
| Platform | iOS, Android | iOS, Android (in progress) |
| Multiple markers | Yes | No |
| 3-rd Party Integration | Yes, Unity3D | Yes, Unity3D |

Detailed comparison on  functionalities will be further discussed in later chapters.

AR applications have be deployed on different platforms, each platform has its own characteristic.

Here we compare AR application on different platforms:

| **Platform** | **Advantage** | **Disadvantage** |
|---|---|---|
| PC | 1. Large display screen<br>2. Higher computational power | 1. Lack of mobility<br>2. Need external camera |
| Mobile | 1. High market share<br>2. High mobility<br>3. Touch gesture control<br>4. Built-in camera | 1. Smaller display screen<br>2. Lower computational power |
| Tablet PC | 1. High market share<br>2. High mobility<br>3. Touch gesture control<br>4. Large display screen<br>5. Built-in camera | |

Instead of the old i.Digi.T.able system design, we are going to design another system as a "space sharing" platform. We have considered the special characteristic of our client device - iPad, hence we chose to adapt a more fashionable and interesting design.

Such set up should make good use of the functionalities iPad naturally provides. Possible impressive features we can consider to include:

1. The front cam can be used for video conferencing, players can communicate with each other directly.

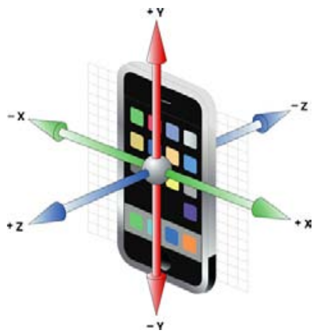
2. The front cam can be used to track the user's position towards to screen, which potentially can be used to implement VPT (Visual Perception Technology) if possible.

3. Players can interact with the game using the touch monitor other than only moving the game objects around.

4. The game can be set up on any plane platform.

5. Game virtual objects can be displayed in a 3-D manner instead of top-view only.

6. The GPS and accelerometer built in the iPad can help enhancing the system.



**1.2.1 accelerometer helps detect device orientation and rolling**

7. Wireless network system enables iPad to go anywhere, however we may need a server to host and process the data.

## 1.3 Objective

In our Final Year Project, our group is trying to use iPad as the development platform to implement an AR game.

The game should enable users using 2 different iPad clients having the experience that they are sharing the same virtual space.

Main objectives of our project:

- Track the real-object mark and determine the camera's position
- Display simple objects on virtual space depends on real space scenes
- Exchange position information between 2 iPad clients
- Implement a simple AR game on iOS platform (iPad)

## 1.4 Development Environment

| **Development** | **Platform / tool** | **Programming Language(s) involved** |
|---|---|---|
| Client program | Mac OS/ Xcode 4.0 | Objective-C |
| Server (network) | Linux / vim,pico | PHP / Shell script / C |

Most of the development processes are carried on Mac OS. It is an obligation for iOS applications to be developed on MacOS.

**Xcode**

Xcode is an integrated IDE for developing MacOS and iOS application. It includes a modified version of free software GNU Compiler Collection, supporting several common programming languages, such as C, C++, JAVA, Objective-C, Python, Ruby,etc. .Its package comes with Cocoa and Cocoa Touch frameworks. The frameworks provides User Interface, Gesture detection control and more necessary libraries for iOS devices.

The Xcode version we are currently using in our development is 4.0.2. This version is released on March 2011.

**PHP**

For the server, we have chosen PHP (PHP: Hypertext Preprocessor) to develop the server program. It is mainly because of its powerfulness as a server side programming language.

PHP is a general-purpose scripting language mainly use for web development. It is free software under PHP License. PHP can be set up on almost every operating system.
PHP programs can be deployed as web application or stand alone programs.

PHP includes free and open source library on its build. It also embeds with MySQL, SQLite as database server.

## 1.5 Runtime Environment

For the development, we have chosen iPad as the platform. iPad is a trendy tablet computing device nowadays. iOS is a mobile operation system developed by Apple Inc. Originally it is designed for iPhone but now is extended onto other Apple's product such as iPad and iPod touch. The first iOS was released in 2007.

In year 2011, there are 500,000 Apps for the iOS platform. iOS has became a popular platform on mobile. The iOS version for our runtime environment is iOS4.3 running on iPad2.

iOS4.3 runs on ARM family processor. For iPad2, it is using the Apple A5 processor and builds on top of a unix-like kernal. The abstract layering architecture of iOS gives developer choices when developing their applications in different layer.

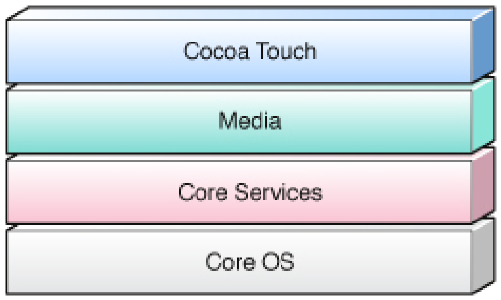


**Fig.1.5.1 iOS technology layer**

The main reason iOS is adopted as our runtime platform is due to the User Interface experience and portable feature of iPad. With a 9.7 inch multi-touch sensible monitor, virtual world can be displayed realistically. We also take advantage of the front/rear cameras built-in on iPad. The process of tracking real object is done by analysing the camera buffer.

There are also much API available since iOS is getting more popular, this would help developers in coding on their project more efficiently.

## iOS simulator

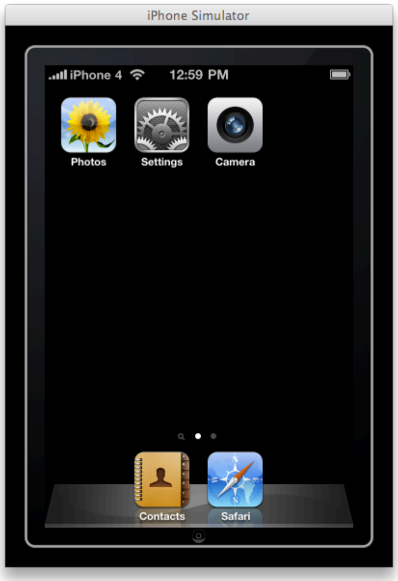


**Fig.1.5.2 iPad Screen on iOS simulator**

Apple provides a debug tool for iOS application developers that let developers need not compile the program to a designated hardware machine every time.

iOS simulator could simulate gestures on real device (such as drag and drop, pinching). Orientation, networking and memory management can also be simulated.

However for our project, camera tracking is an essential element that cannot be simulated by iOS simulator. When we test and debug our work, we shall load the program onto the machine directly.

# Chapter 2. Augmented Reality

## 2.1 History

In 1966 Professor Ivan Sutherland of Electrical Engineering at Harvard University invented the first model of one of the most important devices used in AR today - the head-mounted display. Although the model was experimental and unusable, it was the first step in the development of AR.

Until 1999, AR remained unfamiliar to the consumer. It was because of the requirement of bulky, expensive equipment and complicated software. However, AR gained recognition in some sci-fi films. One instance was RoboCop (1987). The big helmet on Murphy's head gives a stream of data and information to enhance his vision and understanding of what's around.

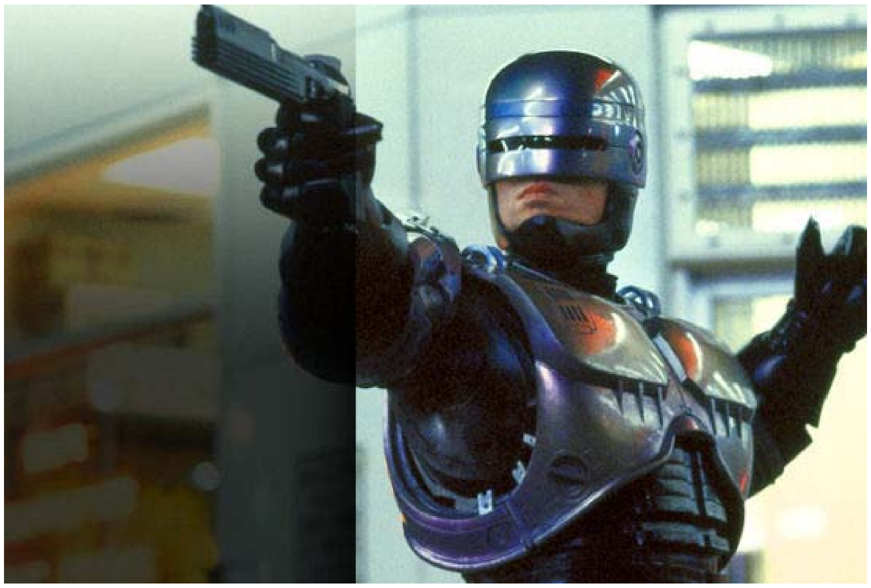


**Fig. 2.1.1 RoboCop Movie Screen Shot**

In recent years, smart phones become so common and most are equipped with camera, GPS, digital compass, etc. Therefore, the role of AR in application becomes more significant. From geo-navigation to video gaming, AR enhances the user experience and realism.

## 2.2 Types of AR

### 2.2.1 Marker-less

Marker-less AR typically uses the GPS or digital compass feature of mobile devices to locate and interact with surroundings. Sometimes, camera is also used. AR information will be displayed on the video.

**Fig. 2.2.1.1 Marker-less AR example**

### 2.2.2 Marker-based

Marker-based AR typically uses the camera feature of mobile devices to analyze markers captured in video. QR codes are probably the most seem application of this type. Besides, the pose information of the marker may be useful sometimes. Users can move the device to view the virtual model in different angle.

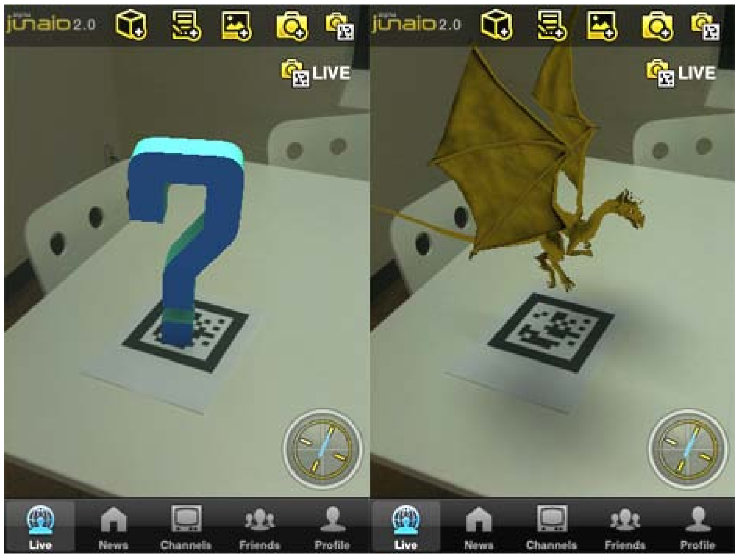


**Fig. 2.2.2.1 Marker-based AR example**

## 2.3 Applications

There are some augmented reality apps for iPhone:

### 1. Geo-navigation e.g. SkyGlass

SkyGlass is an AR compass that display geo-information augmented to the real scene on the screen. It also makes use of GPS tracker,

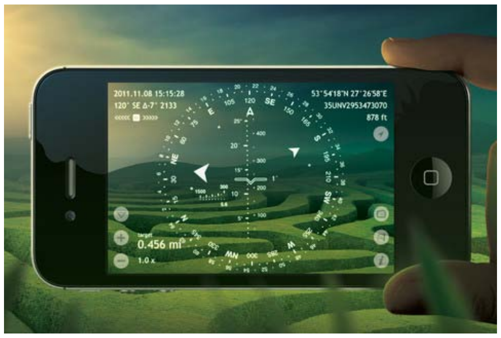


**Fig. 2.3.1 SkyGlass**

### 2. Informative e.g. Wikitude, Layer

Wikitude is an application that when user view a location with the camera, additional and interactive information would be shown on the screen.

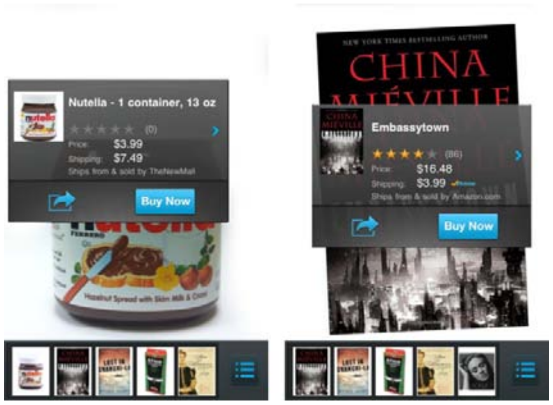


**Fig. 2.3.2 Labeling information in Augmented Reality**

### 3. Translation e.g. Word Lens

Word lens is an application used to translate words in the real world. The result will be displayed directly on the same place and same location on the screen. What the user see is the language is translated into their preferred one.

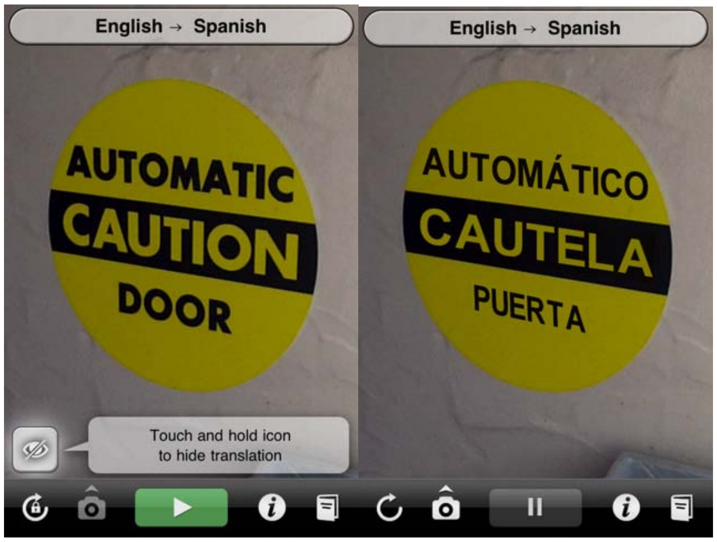


**Fig2.3.3 Word lens**

**4. Sampler e.g. The Sampler by Converse**

This kind of applications usually is developed by product resellers. Customers can view the virtual object on device. The device can be rotated and transposed by moving the real marker in front of the camera. These applications bring new experience to costumers and it is a quite attractive advertisement method.

## 2.4 AR in game industry

AR has its appearance on gaming consoles and mobile devices.

**1. Consoles e.g EyePet**

EyePet is a petting game on PS3. User can interact with the pet using marker and gesture. This is a quite new idea on existing game console platform.

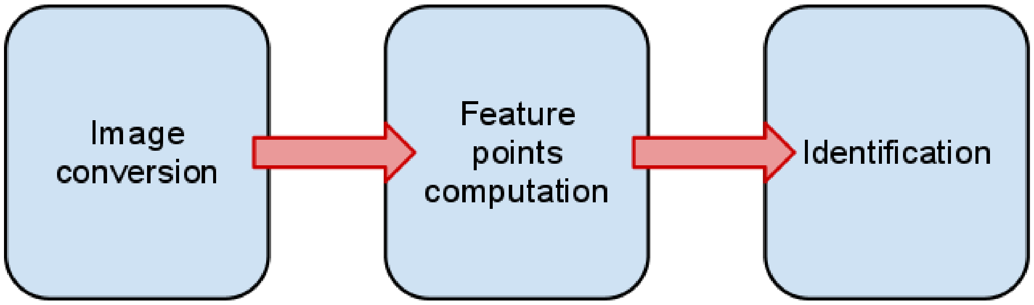
**2. AR Defender**

AR defender is a mobile AR game that involves tower and weapons. Tower defender is a quite popular game on different platforms. User experience is enhanced, user can move around with the device-in-hand as live viewer.

Fig2.4.1 AR defender game interface

## 2.5 Marker detection and recognition

Marker detection and recognition process involves three stages:



### 2.5.1 Image conversion

The first step is to convert the captured frame from colored into binary image. This process is called thresholding and it is the most basic form of image segmentation.

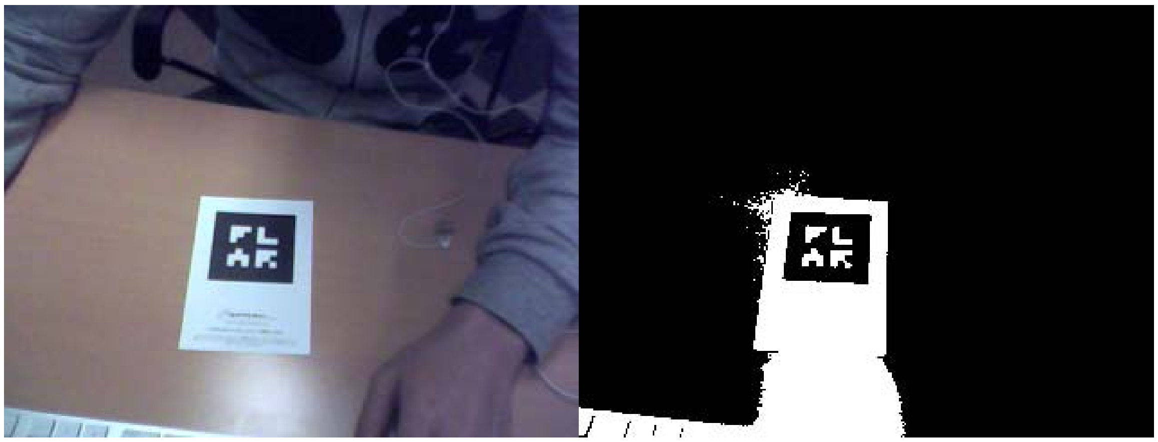


Fig2.5.1.1 Image after threshold processing

The most basic thresholding of image is to choose a fixed threshold value and then compare every pixel to that value. However, fixed thresholding is very much affected by the illumination changes in image or video stream.

In order to solve the problem caused by variations in illumination, adaptive thresholding is invented. The major difference between the two approach is that a different threshold value is computed for each pixel instead. Adaptive thresholding is more robust in this way.

Since the process is repeated over and over while camera capturing video stream, the implementation should be kept as simple and fast as possible. The pseudocode below demonstrates adaptive thresholding for input image *in*, output binary image *out*, image width *w* and image height *h*.

```
function adaptive_threshold(in, out, w, h)
     for i = 0 to w do
          sum = 0
          for j = 0 to h do
               sum = sum + in[i,  j]
               if i = 0 then
                    intImg[i, j] = sum
               else
                    intImg[i, j] = intImg[i – 1, j] + sum
               end if
          end for
          end for
          for i = 0 to w do
          for j = 0 to h do
               x1 = i – s/2
               x2 = i + s/2
               y1 = j – s/2
               y2 = j + s/2
               count = (x2 – x1) x (y2 – y1)
               sum = intImg[x2,y2] – intImg[x2,y1 – 1] –
                    intImg[x1 – 1,y2] + intImg[x1 – 1,y1 –
                    1]
               if (in[i, j] x count) ≤ (sum x (100 –t)/100)
                    then
```

```
                out[i, j] = 0
            else
                out[i, j] = 255
            end if
        end for
    end for
    end function
```

### 2.5.2 Feature points computation

The next step is to compute the feature points on binary image. The corners need to be detected accurately in order to have reliable camera pose estimation.

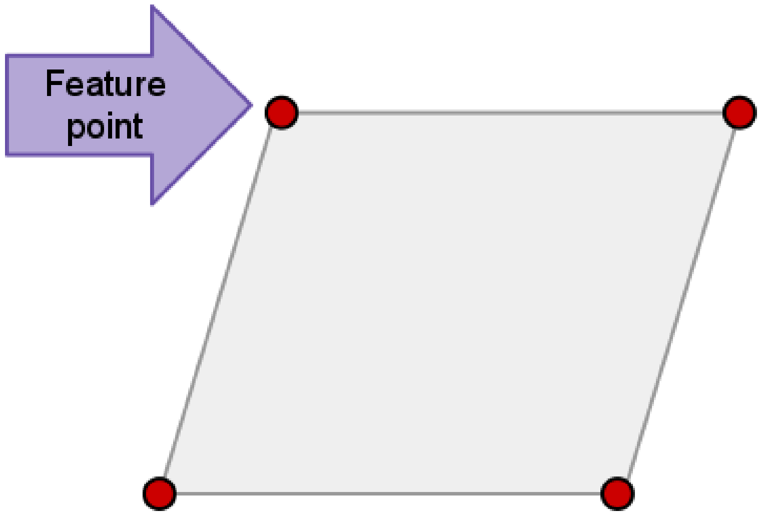


**Fig. 2.5.2.1 Feature point example**

### 2.5.3 Identification

The final step is to restore the effect of rotation, translation and perspective transformation by solving a simple linear system.

Suppose we have found the positions of four corners by feature points computation and the 3D coordinates in object space of the marker's corners are given by (xi, yi, 0) and the measured coordinates of the corners in the image are given by (Xi, Yi). The z coordinate is set to 0 because the marker is planar. All we need to compute are 9 parameters for rotation matrix R and 3 parameters for translation vector T. For simplification, we can divide the numerator and denominator by tz and rearrange to obtain:

$$X_i = \frac{a_1 x_i + a_2 y_i + a_3}{a_7 x_i + a_8 y_i + 1}$$

$$Y_i = \frac{a_4 x_i + a_5 y_i + a_6}{a_7 x_i + a_8 y_i + 1}$$

Hence we can calculate for only 8 elements. Given the 3D coordinates and 2D image coordinates, we solve the linear system below:

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \\ X_8 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -X_1 x_1 & -X_1 y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -X_2 x_2 & -X_2 y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -X_3 x_3 & -X_3 y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -X_4 x_4 & -X_4 y_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -Y_1 x_1 & -Y_1 y_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -Y_2 x_2 & -Y_2 y_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -Y_3 x_3 & -Y_3 y_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -Y_4 x_4 & -Y_4 y_4 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix}$$

The result vector of the linear system implies a normalized marker. By using the result vector, the system can provide users the pose information for drawing virtual 3D objects.

# Chapter 3 Qualcomm AR SDK

## 3.1 Introduction

Qualcomm AR SDK (QCAR SDK in short) fetches live streaming from the device camera. It then analyzes the video by marker detection and provides the 3D spatial information of the detected marker via API. Programmers can use such information to draw appropriate virtual 3D objects on the camera video. As a result, the virtual objects are blended into real footage in real-time.

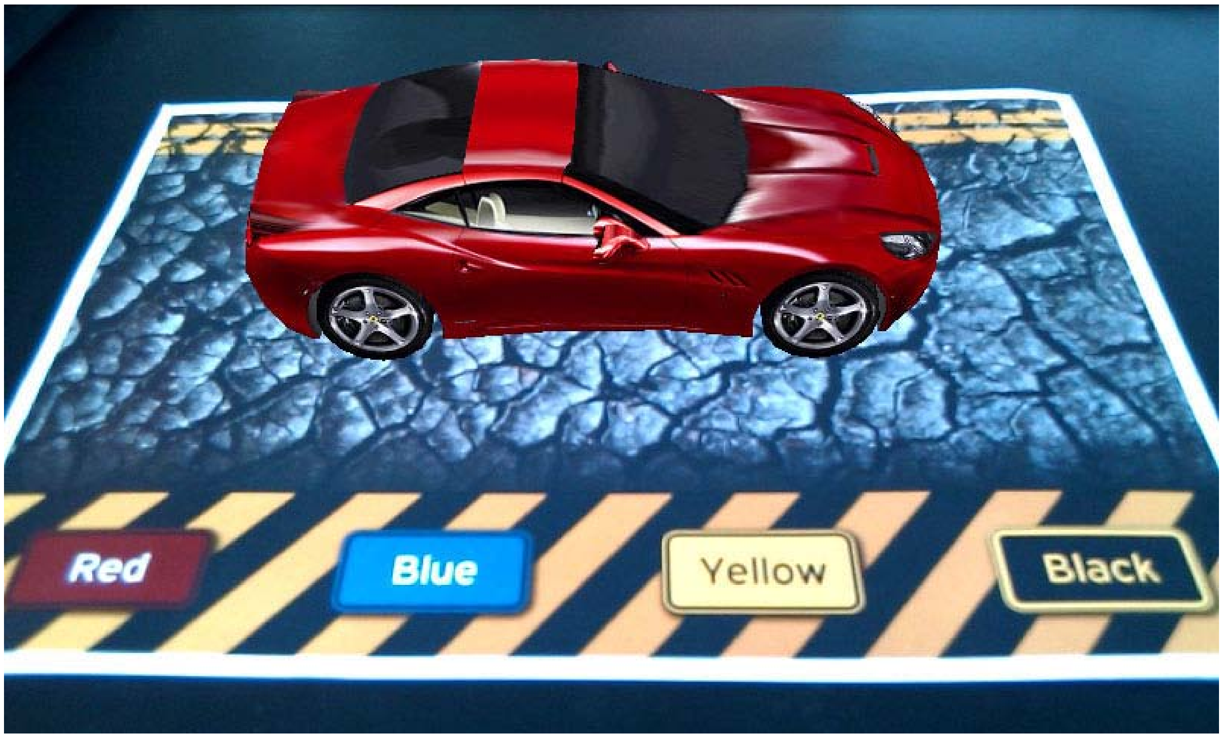


**Fig. 3.1.1 A virtual 3D car is superimposed on top of live camera preview**

Below is the overview of application development with the Qualcomm AR Platform. The platform consists of two components:

**1. Target Management System** (hosted on QDevNet http://ar.qualcomm.at)
Allows developers to upload input image for the markers to be tracked and then download the corresponding target resources.

**2. QCAR SDK**

Provides developers to link their application to the static library i.e. libQCAR.a on iOS or libQCAR.so on Android.

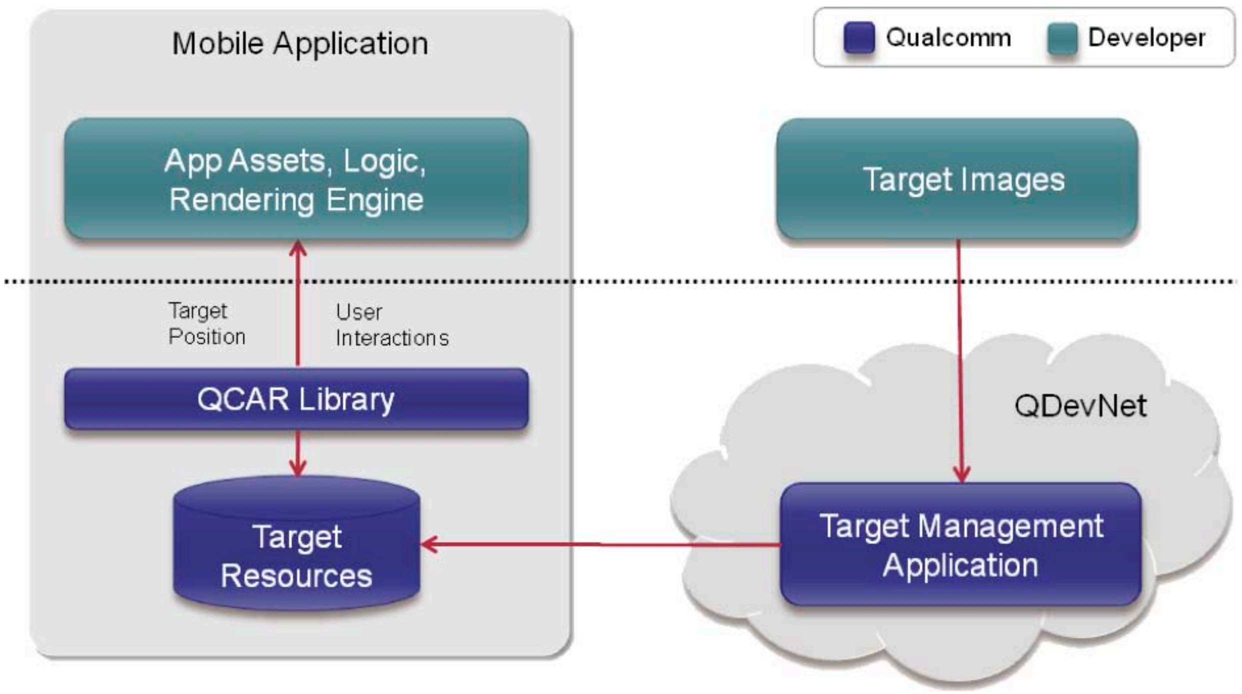


**Fig. 3.1.2 QCAR SDK library**

## 3.2 System architecture

The following are the core components of a QCAR-based application:

**Camera**

The camera class is a singleton. It gives captured camera frames to the tracker for marker detection and recognition. Developers can decide when to start and stop the camera capture.

**Image Converter**

The image converter class is a singleton. It converts captured camera frames from the camera format YUV12 to the RGB565 format for OpenGL ES rendering and the luminance format for marker tracking.

### Tracker

The tracker class is a singleton. It uses the computer vision algorithms to detect and track real world objects in captured camera frames. The target objects are evaluated and the results are stored in a state object that is accessible from application code.

### Renderer

The renderer class is a singleton. It renders captured camera frame to the video background. Its performance is optimized for specific devices.

### Application Code

The application code must involve initialization of all the above components. While the state object is updated for each processed frame, the application code should also update the virtual object location.

### Target Resources

Target resources are generated by the Target Management System. The output files from the system is a binary file storing the marker features and an XML configuration file. They are bundled in the application.

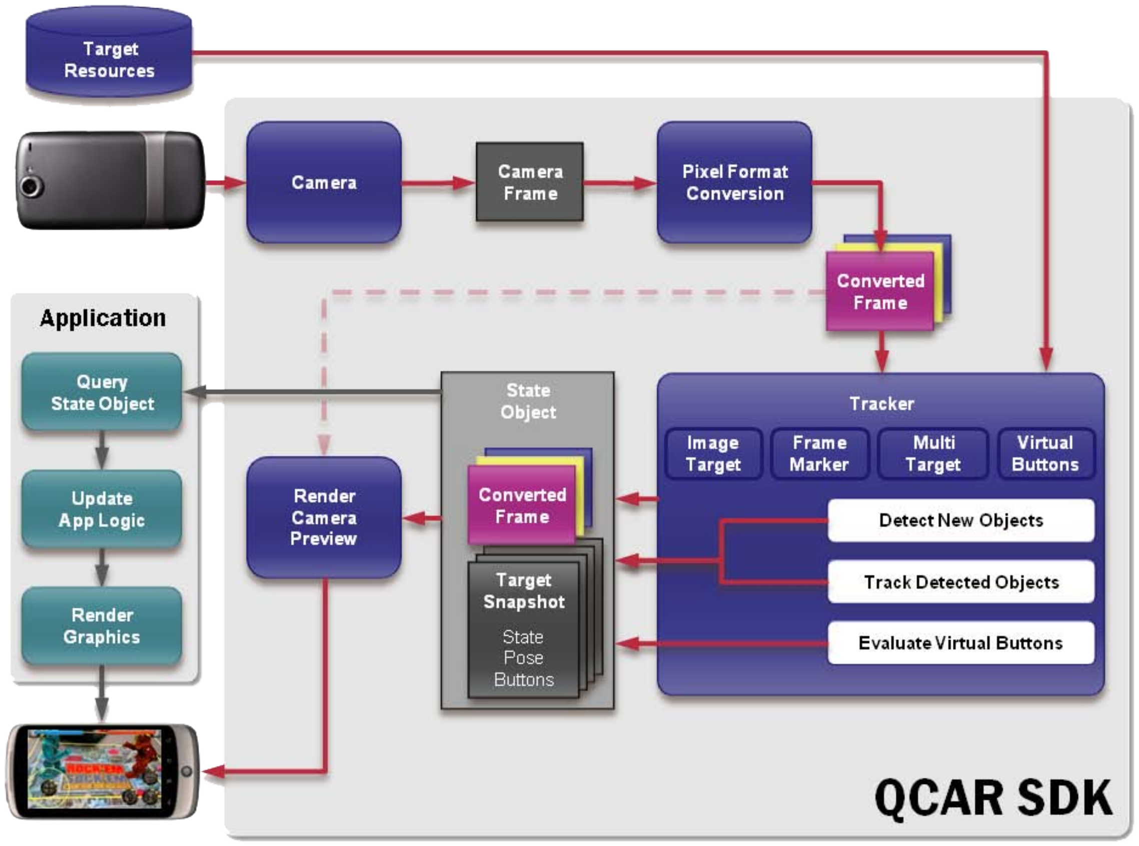


**Fig. 3.2.1 QCAR SDK System Architecture Overview**

## 3.3 Trackables

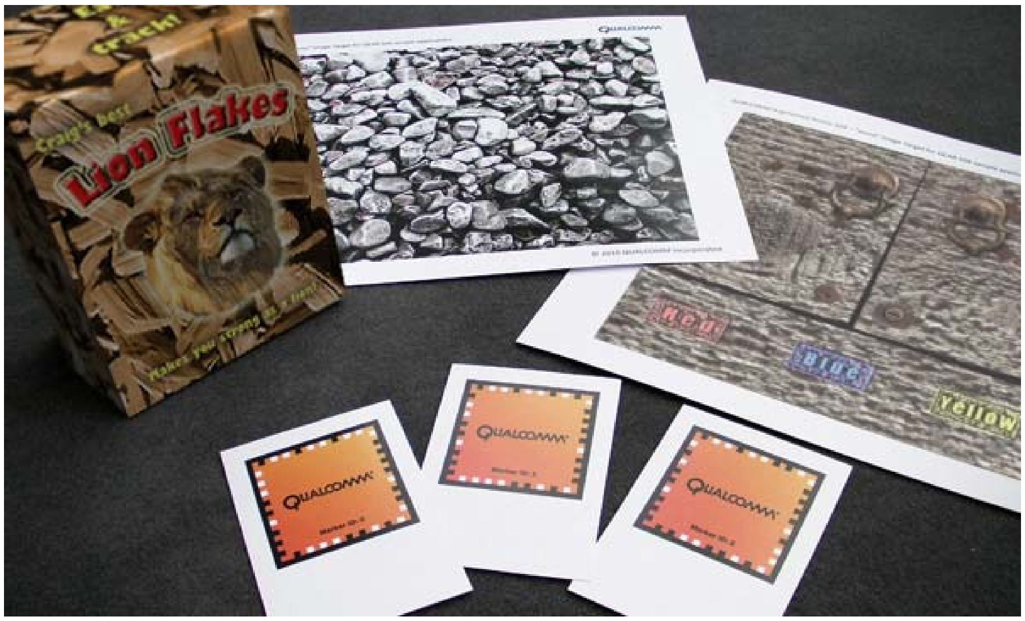


**Fig 3.3.1 QCAR Trackable Marker Samples**

**Definition**

A trackable is any real world object that the QCAR SDK can track in six degrees-of-freedom. A trackable has a name, an ID, status and pose information which are stored in the state object. Image Targets is one kind of trackables. In the following paragraph, we will only discuss Image Targets.

**Parameters**

1. Trackable type

    UNKNOWN_TYPE: an unknown trackable
    IMAGE_TARGET: an Image Target trackable
    MULTI_TARGET: a MultiTarget trackable
    MARKER: a Marker trackable

2. Trackable name

    A string which uniquely identifies the trackable from the database of targets. It has 64 characters in maximum and only contains a-z, A-Z, 0-9, [-_.]

3. Trackable status

    UNKNOWN: the state of the trackable is unknown. Usually returned before tracker initialization.
    UNDEFINED: the state of the trackable is not defined.
    NOT_FOUND: the trackable is not found in the database of targets
    DETECTED: the trackable is detected in this frame
    TRACKED: the trackable is tracked in this frame

4. Trackable pose

    A 3x4 matrix in row-major order which represents the pose information of detected or tracked trackable.

### Coordinate Systems

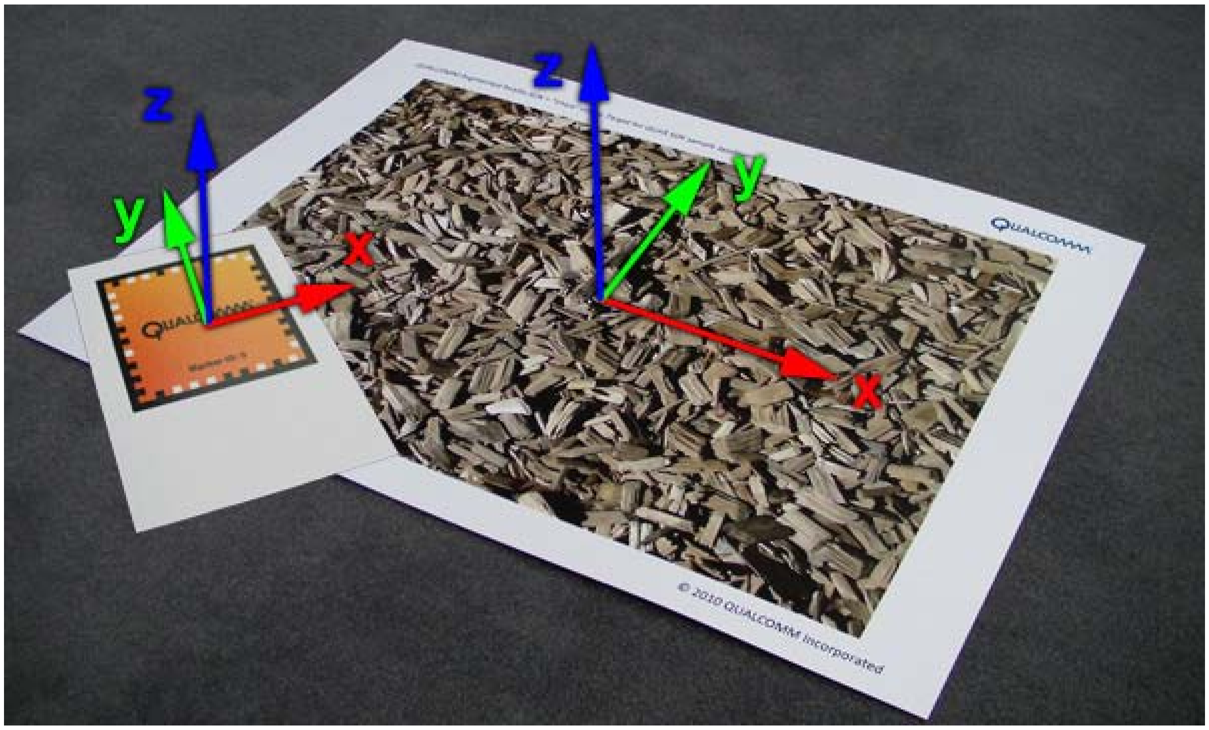


**Fig. 3.3.2 In the QCAR SDK, right-handed coordinate system is used.**

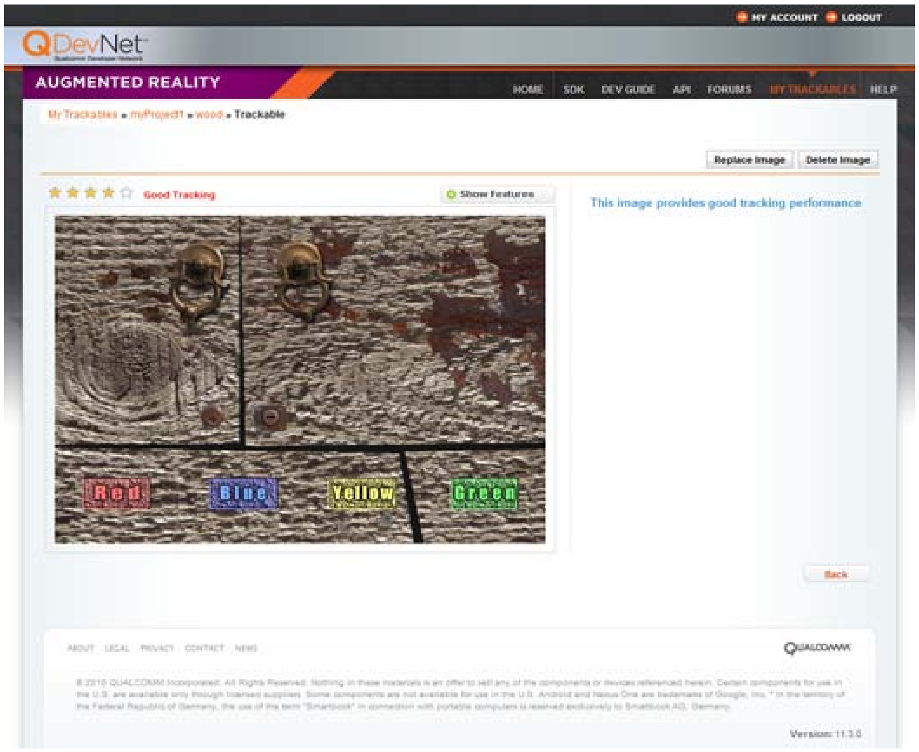
### Relevant API calls

| | |
|---|---|
| virtual **TYPE** | **getType** () const =0 |
| | Returns the type of 3D object (e.g. MARKER) |
| virtual bool | **isOfType** (**TYPE** type) const =0 |
| | Returns true if the object is of or derived of the given type. |
| virtual **STATUS** | **getStatus** () const =0 |
| | Returns the tracking status. |
| virtual int | **getId** () const =0 |
| | Returns a unique id for all 3D trackable objects. |
| virtual const char * | **getName** () const =0 |
| | Returns the Trackable's name. |
| virtual const **Matrix34F** & | **getPose** () const =0 |
| | Returns the current pose matrix in row-major order. |

| | |
|---|---|
| **Frame** | **getFrame** () const |
| | Returns the **Frame** object that is stored in the **State**. |
| int | **getNumTrackables** () const |
| | Returns the number of **Trackable** objects currently known to the SDK. |
| const **Trackable** * | **getTrackable** (int idx) const |
| | Provides access to a specific **Trackable**. |
| int | **getNumActiveTrackables** () const |
| | Returns the number of **Trackable** objects currently being tracked. |
| const **Trackable** * | **getActiveTrackable** (int idx) const |
| | Provides access to a specific **Trackable** object currently being tracked. |

## 3.4 Target management system

The Qualcomm Target Management System allows developers to upload input image and generate feature dataset as target resources. Compiled with the target resources, the application can match images in frame against the feature dataset. To access the system, developer account is required.



This image provides good tracking performance

Fig. 3.4.1 Marker management interface

**Workspace and Projects**

Once the user is logged in, the server will show the workspace. The page has all of the user's projects. Given an input image, a target is computed by processing its natural features. The feature sets used in the runtime application can have more than one target. Projects having a set of targets can be combined to create target resources for download. Only one target resource file is accepted in the runtime application and it can have multiple targets to be detected and tracked by the QCAR SDK. Typically, a new project is created for a new application. A number of images will be uploaded to compute the target data sets. Once the images to be included in target resource are decided, the user can pick those targets and download the dataset of merged natural feature.

Target Resources

An XML configuration file is included in the target resource. It allows to configure certain trackable features and a binary file which has the trackable database. The user can download the target resource file which is named by the project.

**Sample Targets**

The SDK contains input images for the ImageTargets sample applications. The official suggests developers to start by testing the target creation process with these images before uploading their own. In the paragraph below, chips.jpg and stones.jpg will be used for a more detailed explanation of this process. These images are located in:

**ANDROID:**

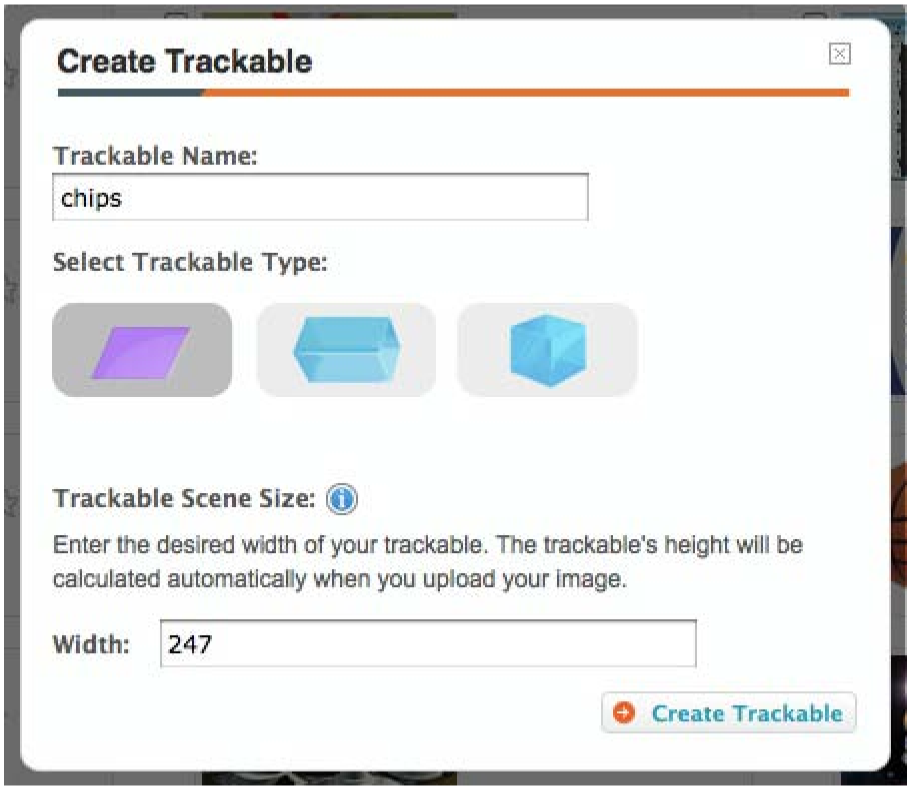
*<DEVELOPMENT_ROOT>\qcar-sdk-xx-yy-zz\samples\ImageTargets\media*

**iOS:**

*<DEVELOPMENT_ROOT>\qcar-sdk-ios-xx-yy-zz\samples\ImageTargets\media*

## How to create an Image Target

First, choose “Create a trackable” and choose “Single Image” as Trackable Type. Input a name for the result target. The name assigned will be used in the application to identify the target during marker detection and tracking. For using chips.jpg and stones.jpg, enter the trackable names “chips” and “stones” respectively.



Create Trackable

Trackable Name: chips

Select Trackable Type:

Trackable Scene Size:

Enter the desired width of your trackable. The trackable's height will be calculated automatically when you upload your image.

Width: 247

**Fig.3.4.2 The dialog for creating trackable**

The width value refers to the printed size of the trackable in millimeters. For instance, the above input will generate a trackable with width 247 millimeters.

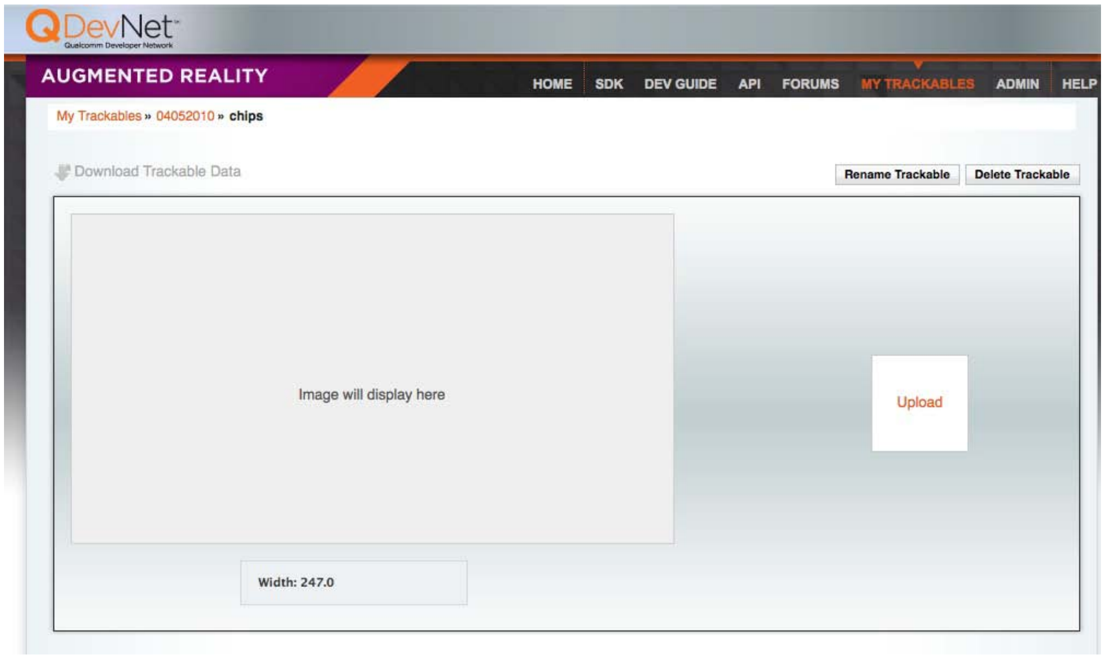


**Fig.3.4.3 Image Target trackable page**

This step is to upload image for marker. Select the “Upload” on the right and select appropriate image. The upload will start the target creation process. After completing the process, a thumbnail of the uploaded image will be shown with rating on a five-star scale. The rating implies the quality of the target for detection and tracking.

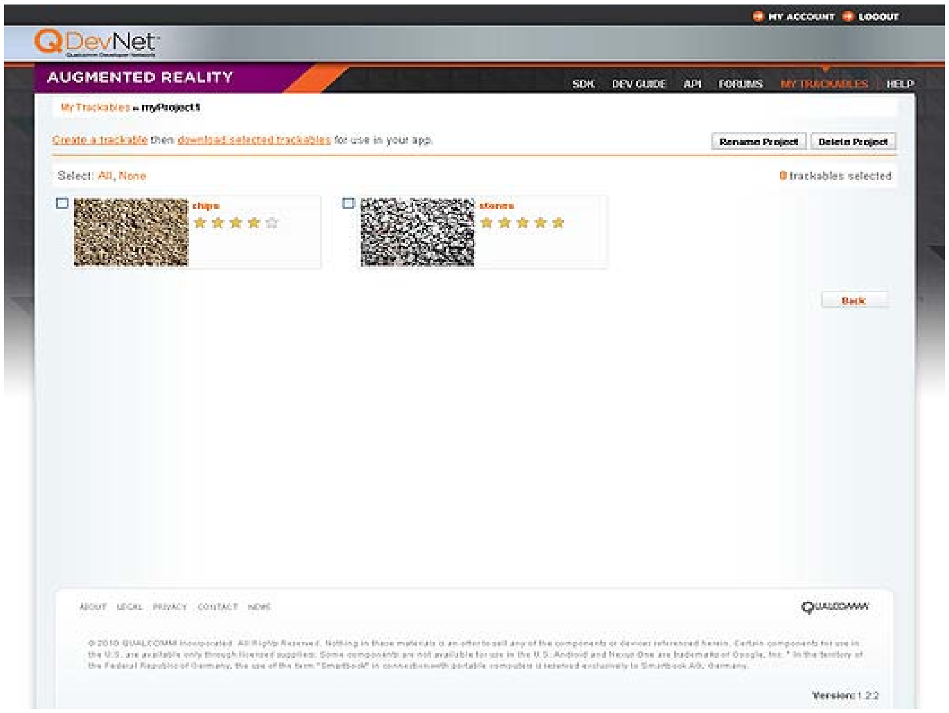


**Fig. 3.4.4 A Projects page showing two uploaded targets**

When the above steps are done, the develoepr can download the trackable by choosing “Download Trackable Data”. Alternatively, he can go back to the projects page and click “Download the Target Resources”.

**How to choose input images**
There are some notes on choosing image to be high-quality marker

**1. Rich in detail**
( for example, sport scenes, street scenes or a mixture of items)

**2. Good in contrast**
( for example, both bright and dark regions are present on the image )

**3. No repeatitive patterns**
( for example, a checkerboard should not be used )

The image needs to be printed in high resolution ( more than 200 to 300 dpi).

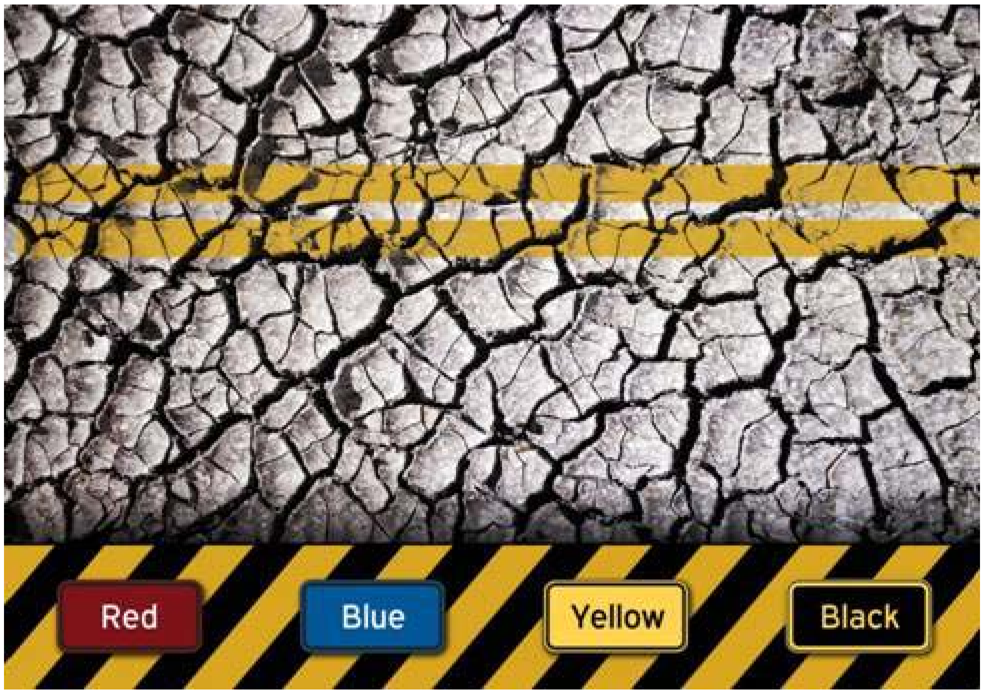


**Fig. 3.4.5 High resolution printed image for creating target resource pipeline**

To create target, the original image should be down-scaled to a resolution similar to the live preview camera resolution on the mobile device. The aspect ratio of the target image must kept the same on printed. Only 8- or 24-bit PNG and JPG formats and less than 2MB size images are acceptable.

## 3.5 Development on iOS

The Qualcomm AR SDK has been tested successfully on Mac OS X 10.6 Snow Leopard and 10.7 Lion. The downloaded installer has the following result directory:

| | |
|---|---|
| <DEVELOPMENT_ROOT>/ | |
| qcar-ios-xx-yy-zz/ | |
| build/ | QUALCOMM Augmented Reality SDK |
| include/ | Commented header files |
| lib/ | Static link libraries |
| licenses/ | License Agreements |
| samples/ | Sample applications with full source code |
| Dominoes/ | Dominoes game featuring dynamic virtual buttons, sound and touch screen interactions |
| ImageTargets/ | Sample app that tracks two Image Targets |
| FrameMarkers/ | Sample app that tracks multiple Markers |
| MultiTargets/ | Sample app that tracks a Multi Target |
| VirtualButtons/ | Sample app that shows Virtual Button interactions |
| assets/ | Additional assets for the QCAR SDK |
| readme.txt | Starting read-me document |

The SDK is separated from developer applications to ensure easier updates to the SDK while not affecting the application files.

## 3.6 Compare with String AR

We want to compare Qualcomm AR SDK with String AR in terms of development, performance and licensing.

**Portability**

Both Qualcomm AR SDK and String AR are available on iOS. They can be easily integrated to iOS projects with a few lines of code and proper configuration for linking library files. However, Qualcomm AR SDK does not provide project templates and new developers need to start from the sample code.

Regarding the Android platform, only Qualcomm AR SDK has provided complete support to developers. In fact, the SDK begins at Android earlier than iOS. On the other hand, the Android version of String AR is still in progress of development and unavailable to developers. Therefore, the Qualcomm AR SDK seems to be a better option when considering porting our application to Android platform in the future.

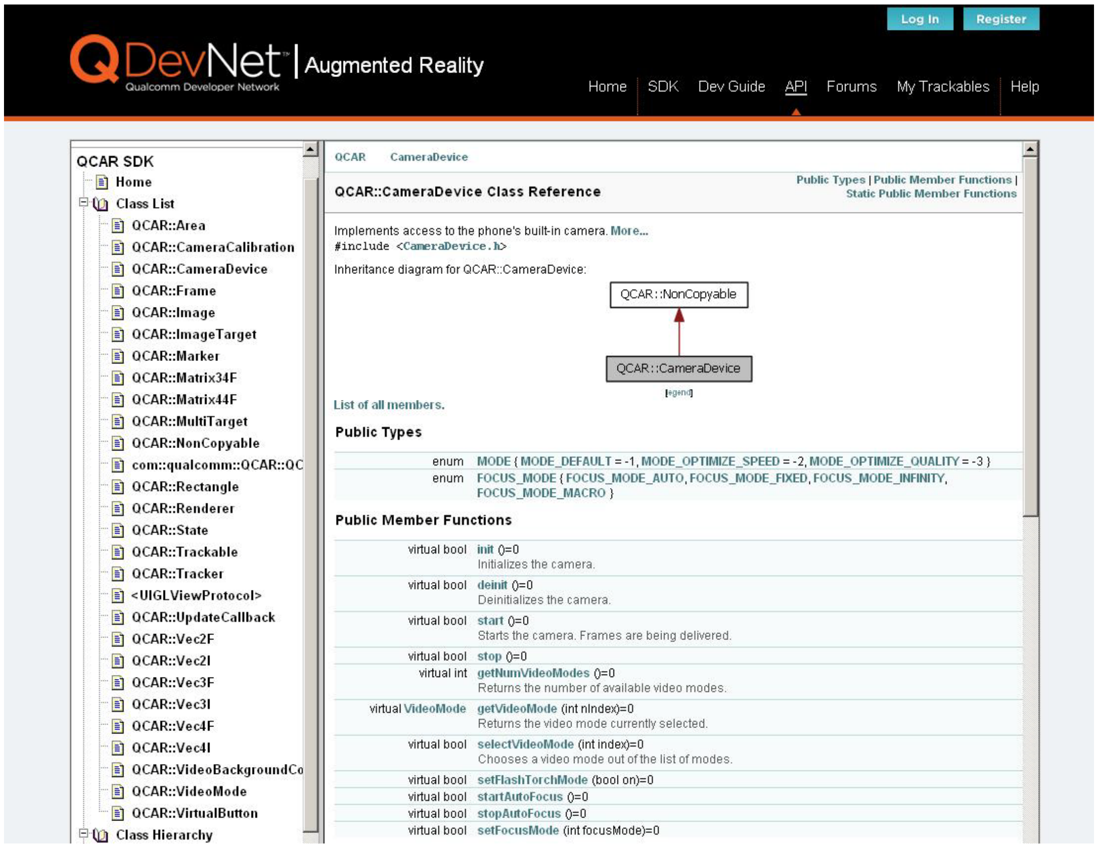
**Flexibility**

The Qualcomm AR SDK has an object-oriented structure. Most of its core classes are singletons and some creates their own threads. The classes are also interdependent. For example, the tracker class depends on the renderer class. The strong coupling between components limits the freedom of developers to do logic and integrate with other system such as rendering engines. We think that the Qualcomm AR SDK is not flexible enough.

Since we do not have any experience with the String AR SDK, we have no comment on the API's flexiblity.

**Documentation and reference**

The Qualcomm AR SDK provides online documentation at https://ar.qualcomm.at/qdevnet/api. Its search function is very easy to use. All functions are properly described in text. Besides, sample projects are offered in the download files of the SDK. Each sample project provides very good beginning of using different features of the system. We appreciate the effort made by Qualcomm officials for such complete, informative documentation and reference.



QCAR SDK
- Home
- Class List
  - QCAR::Area
  - QCAR::CameraCalibration
  - QCAR::CameraDevice
  - QCAR::Frame
  - QCAR::Image
  - QCAR::ImageTarget
  - QCAR::Marker
  - QCAR::Matrix34F
  - QCAR::Matrix44F
  - QCAR::MultiTarget
  - QCAR::NonCopyable
  - com::qualcomm::QCAR::QC
  - QCAR::Rectangle
  - QCAR::Renderer
  - QCAR::State
  - QCAR::Trackable
  - QCAR::Tracker
  - <UIGLViewProtocol>
  - QCAR::UpdateCallback
  - QCAR::Vec2F
  - QCAR::Vec2I
  - QCAR::Vec3F
  - QCAR::Vec3I
  - QCAR::Vec4F
  - QCAR::Vec4I
  - QCAR::VideoBackgroundCo
  - QCAR::VideoMode
  - QCAR::VirtualButton
- Class Hierarchy

QCAR CameraDevice

**QCAR::CameraDevice Class Reference**

Public Types | Public Member Functions | Static Public Member Functions

Implements access to the phone's built-in camera. More...
`#include <CameraDevice.h>`

Inheritance diagram for QCAR::CameraDevice:

$$\begin{tikzcd}
\text{QCAR::NonCopyable} \\
\text{QCAR::CameraDevice} \arrow[u]
\end{tikzcd}$$

[legend]

List of all members.

**Public Types**

| | |
|---|---|
| enum | MODE { MODE_DEFAULT = -1, MODE_OPTIMIZE_SPEED = -2, MODE_OPTIMIZE_QUALITY = -3 } |
| enum | FOCUS_MODE { FOCUS_MODE_AUTO, FOCUS_MODE_FIXED, FOCUS_MODE_INFINITY, FOCUS_MODE_MACRO } |

**Public Member Functions**

| | |
|---|---|
| virtual bool | init ()=0<br>Initializes the camera. |
| virtual bool | deinit ()=0<br>Deinitializes the camera. |
| virtual bool | start ()=0<br>Starts the camera. Frames are being delivered. |
| virtual bool | stop ()=0 |
| virtual int | getNumVideoModes ()=0<br>Returns the number of available video modes. |
| virtual VideoMode | getVideoMode (int nIndex)=0<br>Returns the video mode currently selected. |
| virtual bool | selectVideoMode (int index)=0<br>Chooses a video mode out of the list of modes. |
| virtual bool | setFlashTorchMode (bool on)=0 |
| virtual bool | startAutoFocus ()=0 |
| virtual bool | stopAutoFocus ()=0 |
| virtual bool | setFocusMode (int focusMode)=0 |

**Fig. 3.6.1 QCAR SDK class reference**

The String AR SDK has no online documentation but some reference PDF files in the download files. Sample projects are also provided. However, compared with the Qualcomm AR SDK, its documentation needs to improve.

## Community

The community of Qualcomm AR is open to public and growing.

The following is the statistics about its online forum (https://ar.qualcomm.at/qdevnet/forums):

| | |
|---|---|
| Threads: | 1,171 |
| Posts: | 5,267 |
| Members: | 15,017 |
| Active Members: | 1,704 |

Their moderators are quite responsible and quick in responding any questions related to the SDK.

The String AR seems to not have a public online area for user feedback or questions since we cannot find any linkage on their homepage.

**Performance**

For tracking a single marker (which is enough for our project), both Qualcomm AR and String AR show promising results in performance. The programs response instantly and smoothly to marker detection regardless how complex the rendering objects are. The frame rate is always kept at 60 fps.

**Licensing**

The Qualcomm AR SDK is free for development and distribution. Therefore, it is very suitable for doing research.

The String AR SDK, however, is only free for limited use. The limited version is for demo and only one marker is trackable. Also, release on App Store is not allowed. The minimum spending of String AR developer plans is USD $99 which we think too expensive for our project. As there is a free option provided by Qualcomm, why don’t we use String.

# Chapter 4. Networking

As the objective of our project is to create a same virtual space for both users on different iPad, clients have to exchange data through network connection. During our early phrase, we investigated a few method for implementing the network connection part. This section covers our findings and justification.

## 4.1 Network Socket

Network Socket is based on Internet-protocol to let computers communicate and exchange data via a connected network, for example the Internet.

In such implementation, we need a known IP address and a designated port number. Clients can connect to the server by setting up a socket connection.

There are different types of sockets commonly used:

**Datagram sockets**, which does not require a connection before sending data. This kind of socket uses User datagram Protocol, in real life, DNS and many online game uses UDP to transfer data.

**Stream sockets ,** the most famous protocol stream sockets implement on is TCP. Data should be sent after connection is established.

**Raw sockets**, it is used by the router to pass though raw packets. ICMP (Internet Control Message Protocol)

Implementing data exchange by socket is a very efficient way in terms of data transfer. However it would be more difficult to set up.

## 4.2 HTTP Request

HTTP Request is based on the Representational state transfer (REST) architecture, it is a commonly used software architecture for distributed hypermedia systems. A typical RESTful architecture consists of clients and a server. A request is sent from the client side, and the server receives and processes the request. Response is then replied from the server to the client.

The REST Architecture has six main constraints:

**Client-server**

This model separates client and server apart. They can be replaced independently since the server concerns with data storage and management and client side concerns interface only.

**Stateless**

For each request to the server, there is no state different. The context or state of client will not be stored in the server.

**Cacheable**

Clients can cache response therefore reducing unnecessary bandwidth.

**Layered system**

**Code on demand**(optional)

**Uniform interface**

By using HTTP Request, we transfer data through HTTP (Hypertext Transfer Protocol). Each time, client request the server for updates, client also sends server the most updated information (such as user input, location moved). Server receives client information and log them down, then response with the updated information from another client.

## 4.3 Game center

Game Center
Authentication
Friends
Voice
Auto-match
Achievements
Leaderboards

Game Center is an online multi-player social gaming network released by Apple.

The main functionalities of the Game Center are:

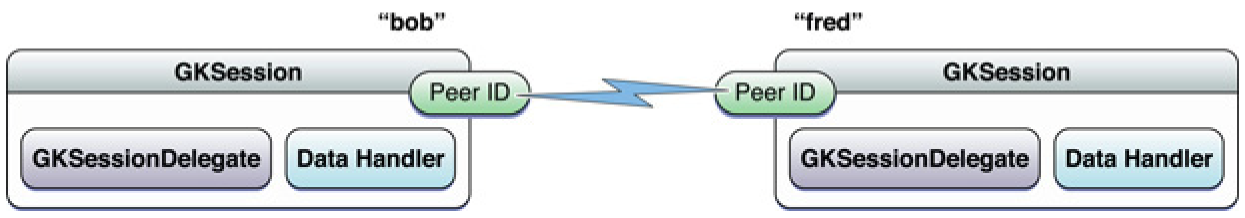
- Users can invite friends to play a game
- Match online with another user via game center
- Authentication
- Achievement tracking
- Leaderboards

Game Center supports iOS version 4.1 or above.

There are no restriction on data format as stated in the Apple developer's note.

## 4.4 Peer to Peer

iOS gamekit provides a few protocol for iOS device to connect with each other locally. For example, the ad-hoc connection between two peer device via local wireless network. Sessions are created and disclosed, then the devices are connected to the network. Data can be sent through the data channel.



However, peer picker controller only creates Bluetooth or Local wireless connections. However, the Bluetooth is only a short-distance wireless connection protocol, with the maximum range around 100m. We prefer a connection method supports a larger range.

# Chapter 5.Design and Implementation

## 5.1 Idea

Our idea is to construct a virtual space for two connected players to interact with each other. The game is played like dodge ball. To shoot a ball, the player has to tap the screen. The player needs to move left and right in order to dodge balls thrown from the opposite side.

First, both user needs a marker to start the game. User should use the front camera to track the position of the marker. The program will analyse data and generate client's relative position. So that user can view the virtual world in the iPad display screen directly in a first-person point of view.

After the relative position of a client is generated, client would communicate with the server. Position information is updated to the remoted server during the commincation. At the same time,server would notify the client if there is any update from another client.

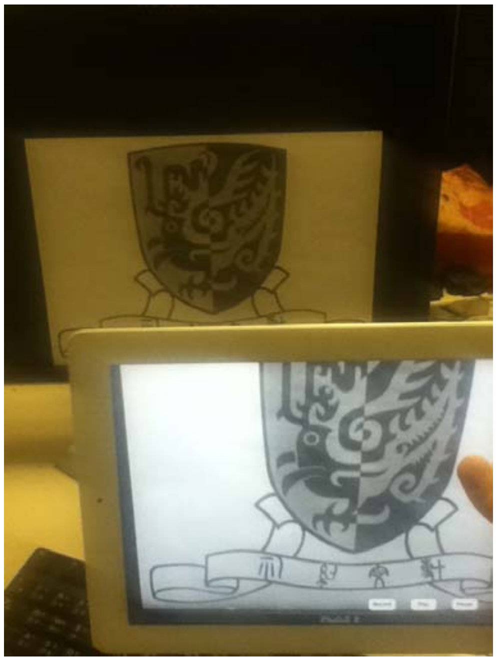


**Fig. 5.1.1 Implementation of position tracking**

## 5.2 Settings

The system contains client side and server side. In terms of implementation, we sub-divide the components of this project into there main parts:

**1. Marker tracking module**

To analysis camera input, to identify and track the camera movement base on a marker. Hence calculate the position and movement data for other modules.

**2. Network connection module**

This module provide interface for other modules to communicate with e server. Data exchange via the network is through the network connection module.

**3. Virtual world construction module**

The virtual world construction module visualize data and position information generated by other modules. It is responsible for the computation of the screen according to point of view.

**4. Game engine**

The game engine includes the AR SDK part and game rules computation module. It reads inputs from other modules and processes it. Result will be output to Virtual world construction module to display.

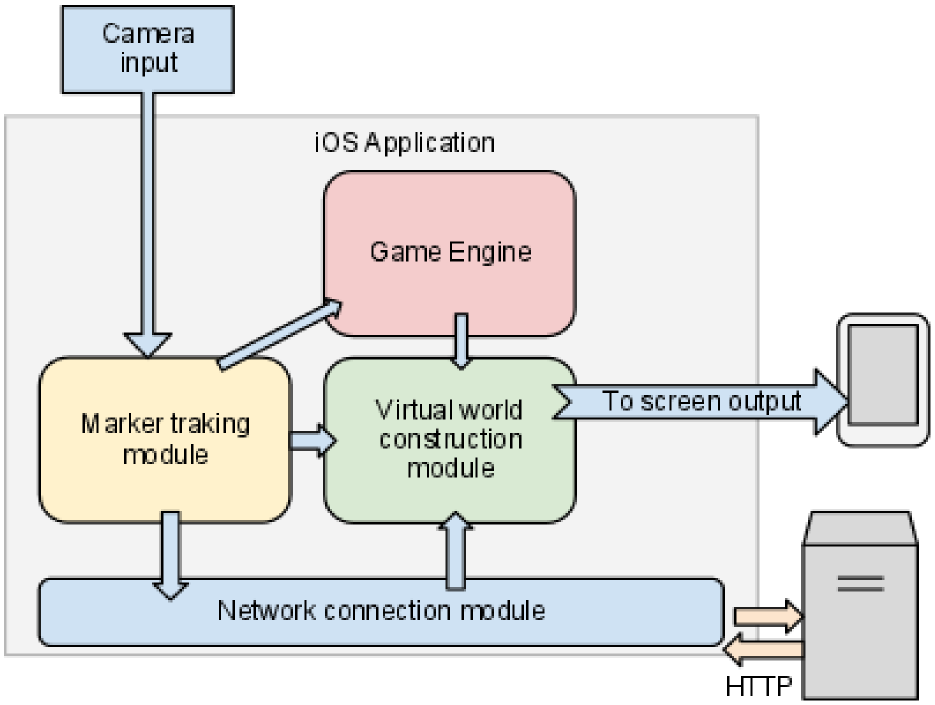


**Fig. 5.2.1 Overview of the system design**

## 5.3 Modules

### 5.3.1 Marker Tracking Module

The Marker Tracking Module is the agent that directly communicate with the Qualcomm AR SDK. It is a finite state machine with the following status:

APPSTATUS_UNINITED: the application and the QCAR component is uninitialized

APPSTATUS_INIT_APP: the application is initialized

APPSTATUS_INIT_QCAR: the QCAR component is initialized

APPSTATUS_INIT_APP_AR: the video renderer is initialized

APPSTATUS_INIT_TRACKER: the tracker and target resources are initialized

APPSTATUS_INITED: the QCAR component is started

APPSTATUS_CAMERA_STOPPED: the camera is stopped capturing

APPSTATUS_CAMERA_RUNNING: the camera is capturing

APPSTATUS_ERROR: error occurs

Every time when the state is changed, the function updateApplicationStatus:(status)newStatus is called. The parameter newStatus is checked and do suitable actions.

When all initialization works are done, the QCAR SDK will call the method renderFrameQCAR on a single background thread. The function is implemented by the user like below:

```
- (void)renderFrameQCAR
{
    [self setFramebuffer];

    // Clear colour and depth buffers
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // Render video background and retrieve tracking state
    QCAR::State state = QCAR::Renderer::getInstance().begin();

    if (QCAR::GL_11 & ARData.QCARFlags) {
        glEnable(GL_TEXTURE_2D);
        glDisable(GL_LIGHTING);
        glEnableClientState(GL_VERTEX_ARRAY);
        glEnableClientState(GL_NORMAL_ARRAY);
        glEnableClientState(GL_TEXTURE_COORD_ARRAY);
    }

    glEnable(GL_DEPTH_TEST);
    glEnable(GL_CULL_FACE);

    ImageTargetsAppDelegate* delegate = (ImageTargetsAppDelegate
*)[[UIApplication sharedApplication] delegate];

    for (int i = 0; i < state.getNumActiveTrackables(); ++i) {
        // Get the trackable
        const QCAR::Trackable* trackable =
state.getActiveTrackable(i);
        QCAR::Matrix44F modelViewMatrix =
QCAR::Tool::convertPose2GLMatrix(trackable->getPose());

        if ( RECSTATUS_RECORDING == delegate.recStatus ) {
            QCAR::Matrix44F invModelViewMatrix;
            ShaderUtils::invertMatrix(&modelViewMatrix.data[0],
&invModelViewMatrix.data[0]);
            // Get the camera's position from the inverse matrix
            NSArray* camPos = [[NSArray alloc]
initWithObjects:[NSNumber
numberWithFloat:invModelViewMatrix.data[12]],[NSNumber
numberWithFloat:invModelViewMatrix.data[13]],[NSNumber
numberWithFloat:invModelViewMatrix.data[14]], nil];
            [delegate.objStates addObject:(id)camPos];
            break;
        } else if ( RECSTATUS_PLAYING == delegate.recStatus ||
RECSTATUS_PAUSED == delegate.recStatus ) {
            NSArray* tmpPos = [delegate.objStates
objectAtIndex:delegate.currentFrame];
            float x = [[tmpPos objectAtIndex:0] floatValue];
            float y = [[tmpPos objectAtIndex:1] floatValue];
            float z = [[tmpPos objectAtIndex:2] floatValue];

            // Choose the texture based on the target name
            int textureIndex = (!strcmp(trackable->getName(),
"stones")) ? 0 : 1;
            const Texture* const thisTexture = [ARData.textures
objectAtIndex:textureIndex];

            // Render using the appropriate version of OpenGL
            if (QCAR::GL_11 & ARData.QCARFlags) {
                // Load the projection matrix
                glMatrixMode(GL_PROJECTION);
                glLoadMatrixf(projectionMatrix.data);

                // Load the model-view matrix
                glMatrixMode(GL_MODELVIEW);
                glLoadMatrixf(modelViewMatrix.data);
                glTranslatef(x, y, z - kObjectScale);
                glScalef(kObjectScale/2, kObjectScale/2,
kObjectScale/2);

                // Draw object
                glBindTexture(GL_TEXTURE_2D, [thisTexture
textureID]);
                glTexCoordPointer(2, GL_FLOAT, 0, (const
GLvoid*)&teapotTexCoords[0]);
                glVertexPointer(3, GL_FLOAT, 0, (const GLvoid*)
&teapotVertices[0]);
                glNormalPointer(GL_FLOAT, 0, (const
GLvoid*)&teapotNormals[0]);
                glDrawElements(GL_TRIANGLES,
NUM_TEAPOT_OBJECT_INDEX, GL_UNSIGNED_SHORT, (const
GLvoid*)&teapotIndices[0]);
            }
            if ( RECSTATUS_PLAYING == delegate.recStatus ) {
                if ( delegate.currentFrame <
delegate.objStates.count - 1 ) {
```

```
                    delegate.currentFrame++;
                }
            }
        }
    }

    glDisable(GL_DEPTH_TEST);
    glDisable(GL_CULL_FACE);

    if (QCAR::GL_11 & ARData.QCARFlags) {
        glDisable(GL_TEXTURE_2D);
        glDisableClientState(GL_VERTEX_ARRAY);
        glDisableClientState(GL_NORMAL_ARRAY);
        glDisableClientState(GL_TEXTURE_COORD_ARRAY);
    }

    QCAR::Renderer::getInstance().end();
    [self presentFramebuffer];
}
```

The most important part of the previous code is:

```
      for (int i = 0; i < state.getNumActiveTrackables(); ++i) {
            const QCAR::Trackable* trackable =
state.getActiveTrackable(i);
            QCAR::Matrix44F modelViewMatrix =
QCAR::Tool::convertPose2GLMatrix(trackable->getPose());
```

The function call `state.getNumActiveTrackables()` returns the total number of markers detected and tracked in the current camera frame. In our case, it always return at most 1 because we only want to track one marker. Then a reference to the trackable object is obtained from the state object. By using the helper functions provided in the SDK, we can get the pose information and draw virtual 3D objects based on them. The workflow is illustrated as below:

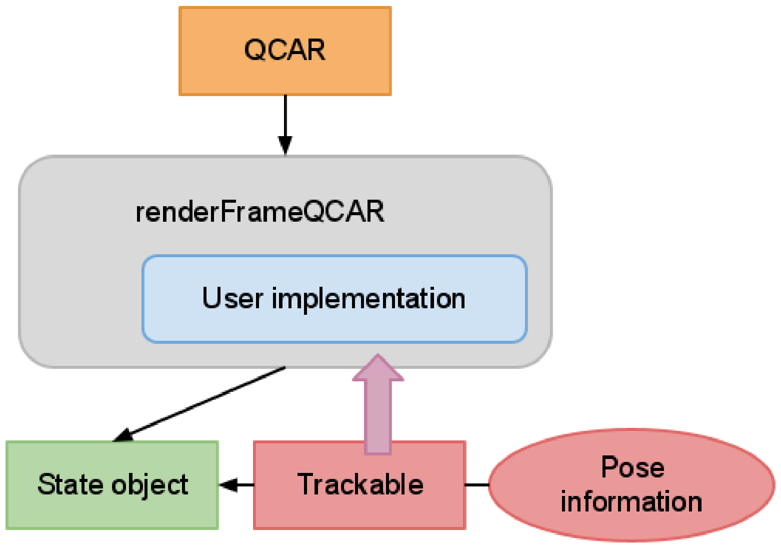


**Fig. 5.3.1 Marker tracking based on QCAR SDK**

## 5.3.2 Network Connection Module

We have discussed some prototypes about how should we exchange data via the Internet.
Here are the criteria we came up when we design the methodology for the network connection module.

**1. Data size**
The data to be sent over the Internet should be kept in a smaller packet size. Only critical and important information should be exchanged. The format of how the data is represented is also a critical factor.

**2. Network load**
As our system depends quite a lot on real time data exchange, the design of the network streaming protocol should minimize the network load in order to simulate a real time environment on different clients smoothly.

**3. Accessibility**
The server should be accessible when needed. However, after we have investigate the servers that can be set up in the CSE department. Some

ports are not opened to outside world. For stability, we found hosting the server program on web server a fairly good choice.

**Connection Protocol**

For the network connection methodology, we decided to set up a web server. The server acts as the middle-man and remembering information needed by the clients. The server is also responsible for recognizing clients and redirect correct update information for each side.

**Connection register phase**

A client send a request to register its identity the server.

Since the Server-client model is stateless, we have to make sure the server recognize different kinds (and possibly incorrect connections) correctly, a 32-digit random token string is generated when server replies the connection request initiated by client.

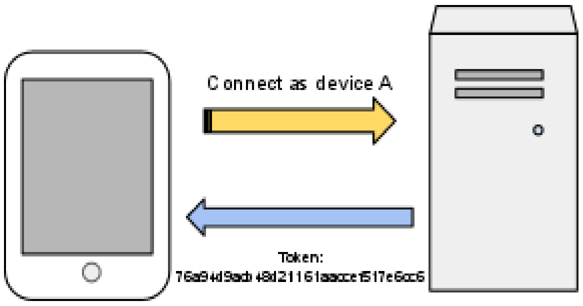


**Fig.5.3.2.1 connection established**

Server then save down the token string as a connection reference.

**Data update phase**

Client connects to server presenting the token previous assigned. Server checks if the token refers to a registered machine. If server could identify the request from client, it updates information in server and sends most updated data to client if it has been modified.

**Advantages**

The previous version we came up is the server distinguish clients depends on their IP address. However we found that it may not work all the time.

Some problems are:

1. client IP may change due to dynamic IP allocation
2. machine behind gateway may share a common IP address, so that server could not identify between the devices

Moreover, current method ensures connection session persist even after network re-connection.

## 5.3.3 Virtual world construction module - Graphics and UI

The graphics in the game is mainly created using openGL ES library.
The openGL ES (OpenGL for Embedded Systems) edition is a subset of the OPENGL library, it is specially designed for embedded systems,such as phones, tablets etc.

OPENGL is a famous 2D/3D graphics library that can create computer graphics with easy logic commands. It provides a low-level applications programming interface (API) between software applications and hardware or software graphics engines.

openGL ES is a lighter version that removes *glBegin/glEnd* for primitive objects, *GL_QUADS ,GL_POLYGONS* and other non-essential elements. The main advantage of openGL ES is its low power consumption compared to the standard OPENGL.

Version we are currently using in our project is openGL ES 1.1.

The virtual world construction module is responsible for generating the virtual scenes on augmented to the reality. It takes input arguments from the Marker Tracking module, Network connection module and Game engine. Corresponding view is then generated according to the input.

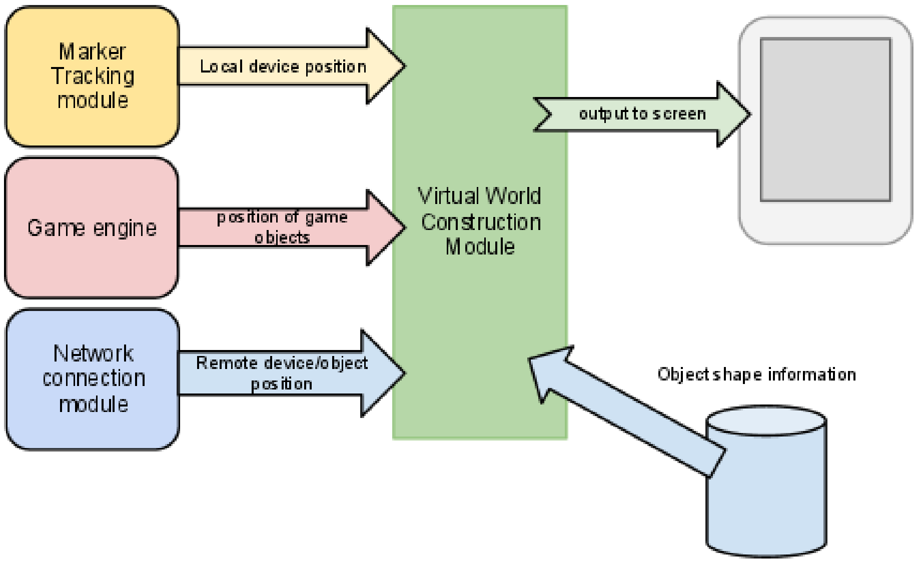


**Fig 5.3.3.1 Input and output of the Virtual World Construction Module**

### 5.3.4 Game Engine

The game engine computes game logic part. Rule and game method is defined. The game engine process inputs according to the rules.

The game engine's design is quite dependent on the game itself. For our sample game dodge ball, game engine takes consideration of user tap input and device position regarding to the marker.

This is the overview of the game engine:

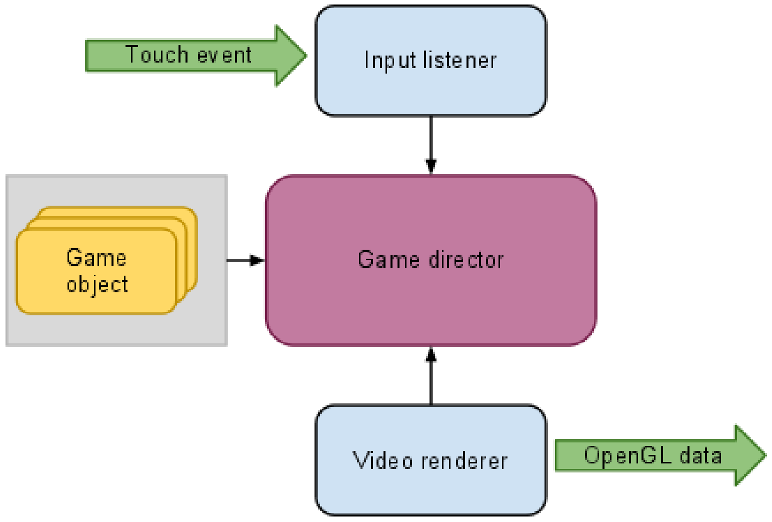


**Fig 5.3.4.1 Game engine system overview**

The input listener is an abstract layer for catching user input to the device. It will do callback functions defined by the user when respective events have been fired. The event object has information about the event such as the position where the user tap on the screen and the gesture of user motion. In the callback functions, the user can reference the game director for game objects.

The design of the input listener process matches with that of iOS. A responder chain is created for event delivery.

The below is an example of responder chain:

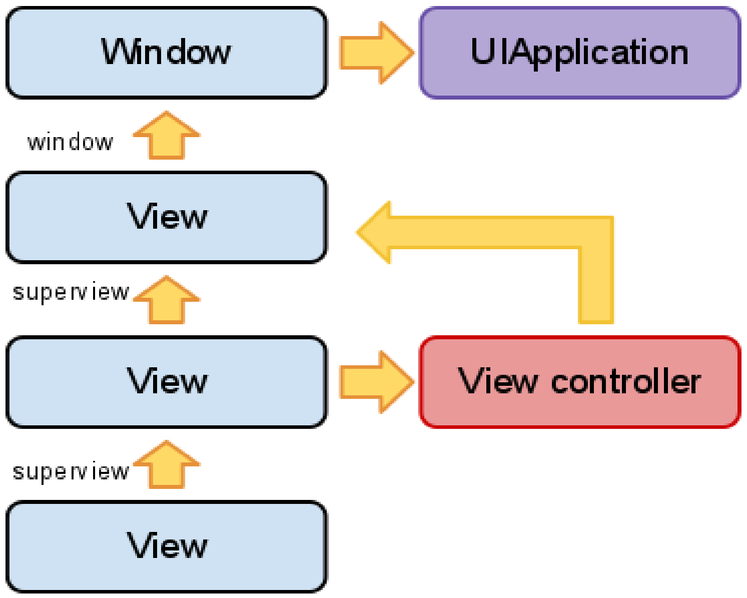


**Fig 5.3.4.2 Responder chain in iOS**

An event object is first delivery to the base view and then its inherited views.

The game director is a centralized class responsible for manage the game world. It is the agent for object creation and deletion. The game director provides a callback function to the user to write their own game logic. Before this callback, the director prepares everything such as initializing the video buffer. After this callback, the director finalize everything such as redrawing the updated video buffer. In the callback, the user can access the game object reference and get or update their information such as the x, y, z position in the virtual 3D world.

Every game object is treated as a node. Each node can have child nodes. The relation tree is like below:

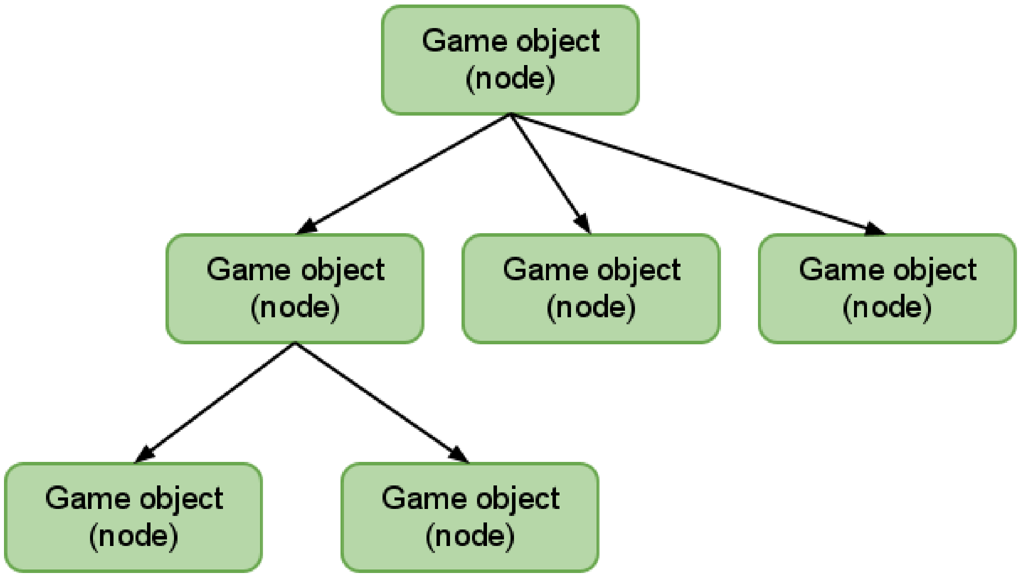


**Fig 5.3.4.3 Sample relationship of game objects**

Operations upon a node will have same effect on its child nodes. Just like a robot can be a node and its body parts are the child nodes of itself. When the robot moves, its body parts also move.

## 5.4 Game design

Here we try to illustrate the game that we implemented to demonstration AR effect.

The game is a 2-player version Dodge ball game.

**Set Up**

An A4 Marker should be set up on the wall. Each user is advised to stand at a place a few meters away the marker. The camera should be able to view the marker clearly.

Clients then should select which player they want to be. For example Player A. User then should use hold up the iPad pointing to the marker in order to set up the initial location.

If the connection is successful, each side should see another player in the virtual world on screen. User can now try to move the iPad to around, see if the perspective of view changes according to the view.

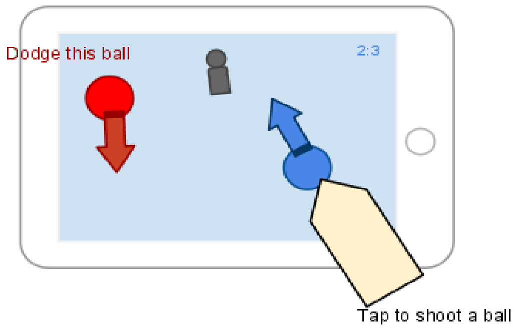


**Fig. 5.4.1 Game interface illustration**

### Start the game

After connection on both sides are established, user can then start the game. The game rule is simple, user can move around in the virtual world by moving iPad they hold. The aim for this game is to hit the opponent with ball as many time as you can; at the same time keep clear from balls thrown by remote user.

### Control

As mentioned, moving the device is equivalent to moving in the game. To throw a ball, just tap the screen, then a ball will be thrown to the other side.

### Winning criteria

<u>Player A wins a game if</u>:

*Number of times that Player B is hit by ball > Number of times that Player A is hit by ball*

<u>The game ties if </u>:

*Number of times that Player B is hit by ball = Number of times that Player A is hit by ball*

or

*No one is hit.*

# Chapter 6. Experiment

## 6.1 Camera match-moving

**Definition**

Camera match-moving is the process of analyzing a video clip or film shot to determine the location of 3D camera. It is the method that we use to find out the relative position of player with respect to their markers. Therefore, a stable camera match-moving is very important for our project.

**Objective**

Since the algorithm and computation are provided by the QCAR SDK as internal code, we cannot change or even view it. We can only investigate the effect of external criteria. In this experiement, we are going to investigate the effect of marker’s properties e.g. size, number of features on the stability of camera match-moving.

**Setup**

The experiment setup is shown as  above. An iPad is placed near the edge of table such that its front camera can capture the paper surface underneath. The paper is printed with a marker for each control. We have created an application to record the position of iPad itself by camera match-moving. When the application starts to record, the paper is pushed forward steadily. Having moved 10cm, the recording ends. The application can replay the movement of iPad by representing it as a 3D model. By observing the replay and position data, we can determine the stability of camera match-moving.

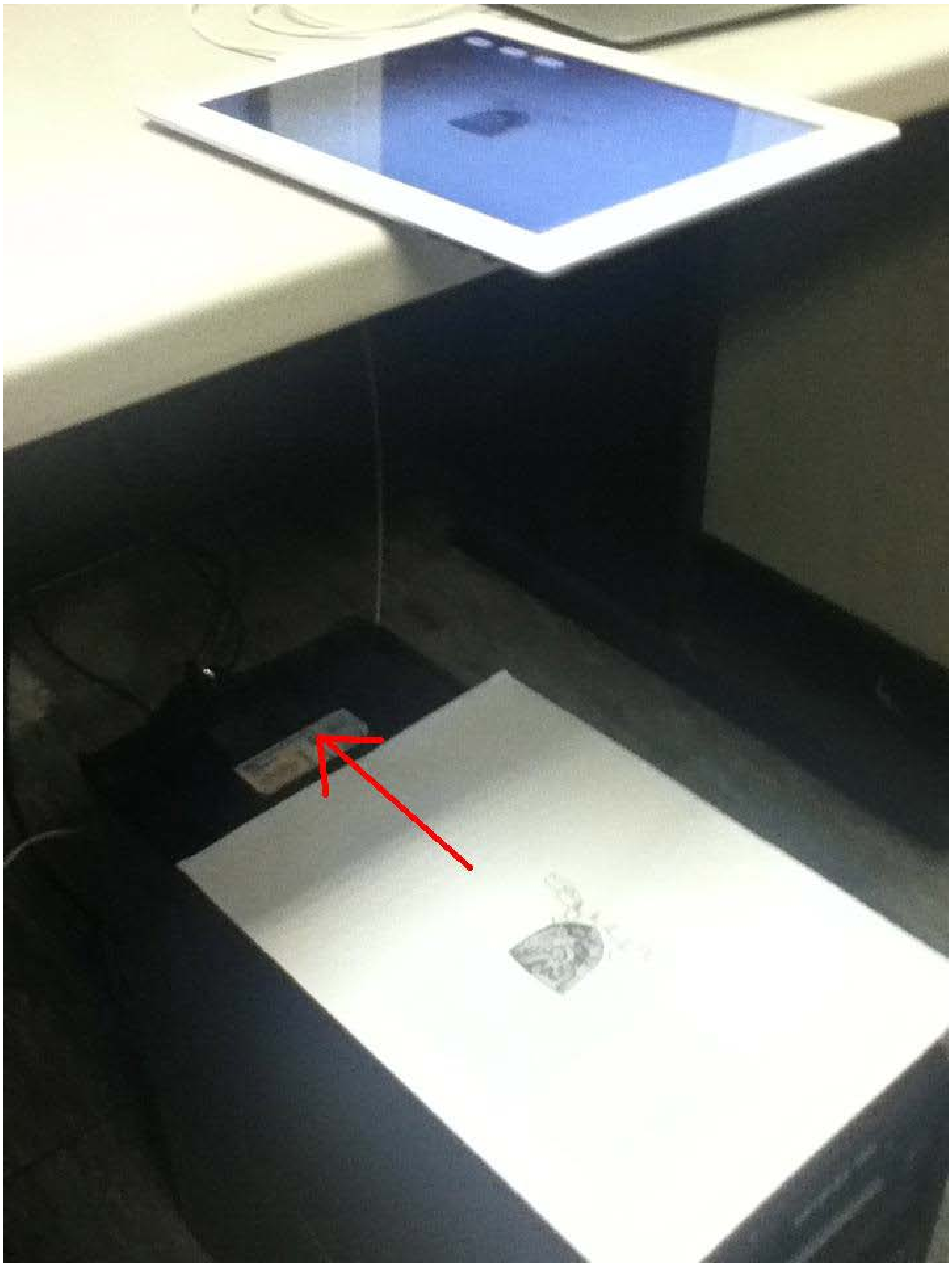


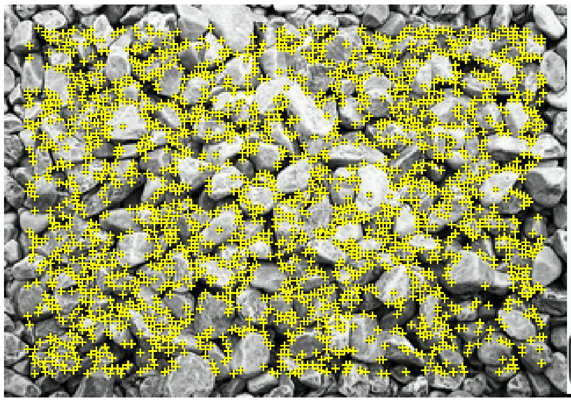
**Fig. 6.1.1 Setup of the experiment**

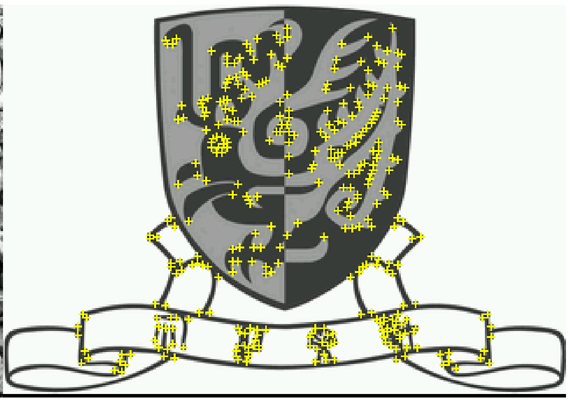
Controls

We will have 4 set of setups. Each one has different size and number of features of marker. For different number of features, we prepared two markers as below:

**Original images**

**With features shown**



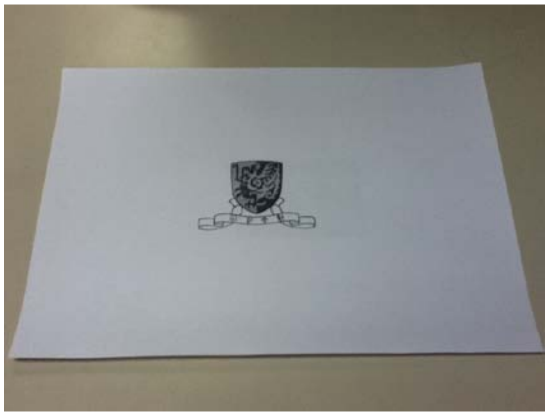


The CUHK logo has less number of features than the stones image. Below is the description from Qualcomm AR Online Management System:

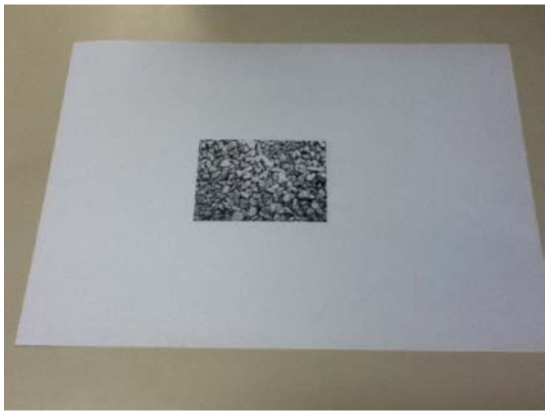
> **CUHK Logo**
>
> This image will track in most conditions but may not be robust to occlusions. If your application demands the best tracking performance then please consider improving the image based by Increasing total number of features in the image by adding more visual detail to the whole scene.
>
> **Stones**
>
> This image provides excellent tracking performance

The 4 different markers (all A4-sized) are the following:

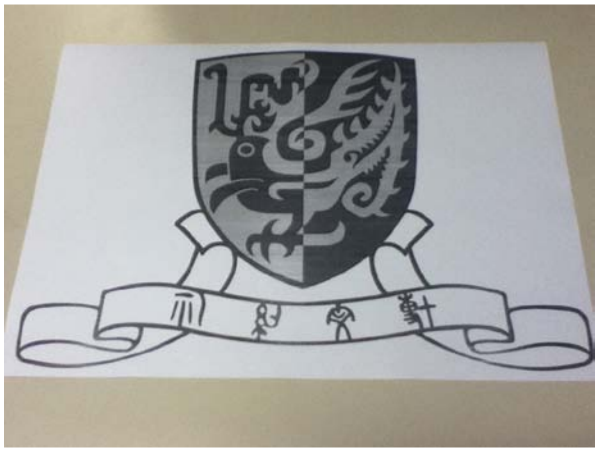
**Control A (less feature and small size)**



**Control B (more feature and small size)**



**Control C (less feature and large size)**



**Control D (more feature and large size)**

**Results**

For the detailed result, Please see appendix.

**Obersvation**

For all controls, the z value keeps constant. It is because the marker is always on the same level of horizon. Hence, we can ignore the z value from our evaluation.

For control A and control B, the x value changes regularly. However, the change differences are not constant for successive frames. Also, the y value should be unchanged since the movement of the paper is aligned to the camera capture area. However, the y value oscillates regularly.

For control C and control D, the x value changes regularly. In addition, the change differences are constant and small for successive frames. Also, the y value remains constant throughout the experiment.

**Evaluation**

The best stability of camera match-moving is given by control C and control D. In the controls, the x value changes smoothly and the y value remains unchanged. In their replays, the movement of 3D model matches the camera movement.

Both control A and control B give unstable camera match-moving results. They are too sensitive to a small change in relative movement between the camera and the marker. In their replays, the 3D model seems to teleport from one place to another and the movement is obviously not smooth. This is an implication of bad camera match-moving.

**Conclusion**

We found that the relative size of the marker is crucial to the stability of camera match-moving. A large marker gives more stable result than a small large marker. The marker size also outweighs that the number of features of the marker.

We think that a full-sized marker on A4 paper has the best result in camera match-moving.

Apart from the stability of camera match-moving, we are also interested in other factors.

**Movement of the iPad**

The movement of users' iPads is very limited because it must have the marker detected in the field of camera view. Hence, we have to allow as more freedom as possible to the device by change the property of the marker.

We found that a larger marker give the device more space to move around. Like the previous result, we think that the best size is full A4 size.

**Camera shake**

It is unavoidable that the camera shakes when user holds the device with hand. When the tracker detects minor shaking, triggers the Virtual world construction module to recalculate the objects' location in the virtual world frequently. Also information will be updated to the server. The displayed world would be shaky, which gives a worse experience to user.

We can apply algorithm to detect and compromise with minor camera shaking action. The algorithm should distinguish between “shaking” and “moving” track of the camera, so that the object display would stay more static in user’s point of view.

## 6.2 Networking

After we have set up the server, the protocol proposed in design section is tested.

**Objective**

Since we will have to send data over the network very frequently, the stability and reliability of the network medium and protocol is important. Therefore it is necessary to test out the implementation.

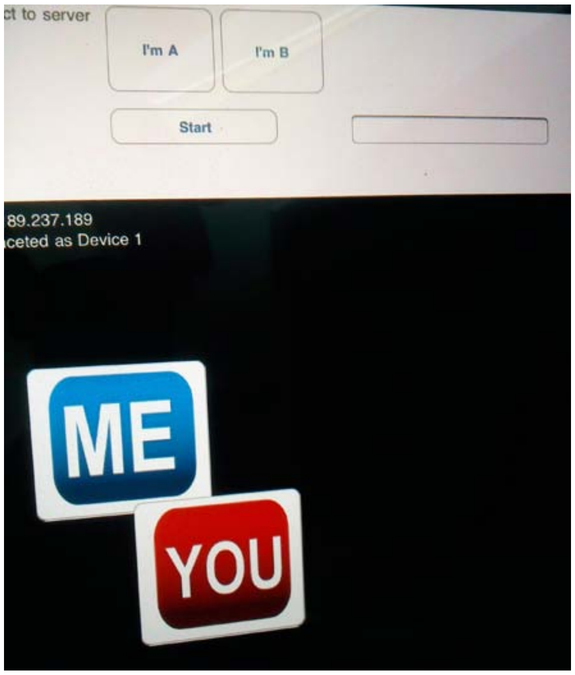
**Setup**

Server program set up on web server of the department.

2 iPad clients and 2 identical markers.

Installed experimental program which can register device on server and send marker location information to server

Instead of a 3D coordination, we send a 2D coordination to the remote client and test on a 2D plane only.



89.237.189
ceted as Device 1

Fig. 6.2.1 Program interface

On this network testing client program, user can see **ME** and **YOU** label on the screen. To connect, press “I’m A” / “I’m B” on top of the menu bar.

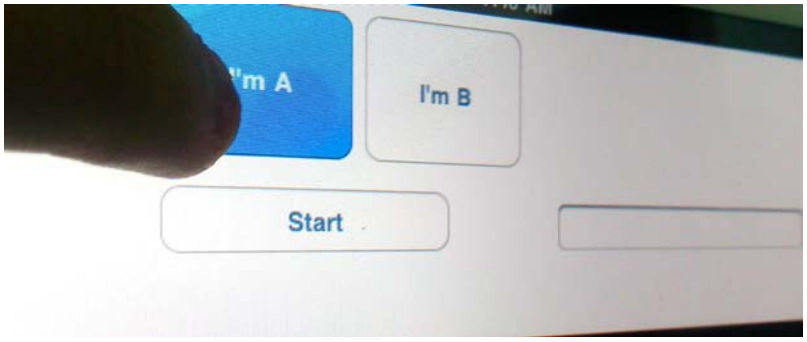
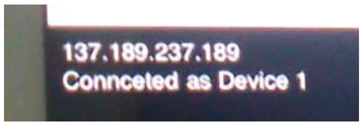


Fig. 6.2.2 Connecting to the server

If it is connected successfully, a message will be printed out.



137.189.237.189
Conncected as Device 1

***“(IP) Connected as Device (1/2)”***

Fig. 6.2.3 Message if connection succeed

**Procedure**

For the testing procedure, we try to drag and drop the "**ME**" label, it will move according to the finger gesture.

The relative coordination is recorded and send to the server immediately. During this process, the client will obtain updated coordination from another client and use this coordination to update the **"YOU"** label.

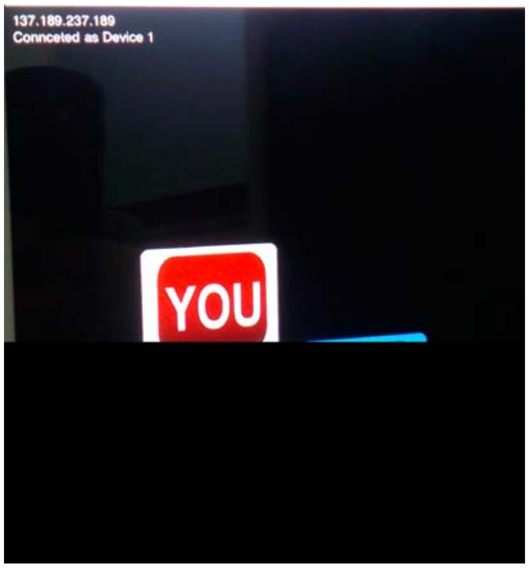


**Fig. 6.2.4 Drag and drop the "ME" label**

**Results**

The observation result : the response is instant but the tracking of "**YOU**" label is a bit lag.

**Evaluation**

The frequency of getting updated information should be raised. But that may need to a huge amount of requests from the client. One of the possible solution is to investigate a motion tweening algorithm on client side, so as to reduce the number of updates needed and hence reduce the use of network bandwidth.

# Chapter 7. Conclusion

We can see Augmented Reality an interesting and exiting field to develop. It is a new experience implementing this technology on iPad.

Our project can mainly be divided into the Marker Tracking Module, Virtual world construction module and Network connection module.

This semester, we mainly focused on tracking AR marker and analysis positional data for the device with the QCAR SDK. We also investigated how to exchange data via the network. To demonstrate the progress, we designed a simple game, which let two users play Dodge ball over the network.

We know that there is a long way to perfection. Our next step is to enhance the performance and continue to develop the rest of the part in the system.

# Chapter 8. Progress and difficulties

Here listed our work and progress during the first semester:

| Period | Progress |
|---|---|
| **Summer 2011** | Research on topic<br>Refine on topic<br>Learn basic iOS development skill<br>Hands on Objective-C language |
| **September 2011** | Search for suitable SDK<br>Test SDK and modify them<br>Test on iOS device<br>Draft game design<br>Server set up |
| **October 2011** | Hands on openGL on iOS<br>Server and network part testing<br>Build prototype |
| **November 2011** | Integrate client and server part together<br>Modify program prototype<br>Refining final design<br>Prepare for the report |

## 8.1 Difficulties and challenges

**Learning Objective-C language and iOS programming**

Objective-C is the main language for developing applications on iOS. It is a reflective, object-oriented programming language. It also adds some smalltalk style syntax which makes it confusing. It also requires much effort to learn iOS programming before we would actually start our working on project. There are much API documentation needed to be gone through for real understanding the mechanism behind.

Therefore we spent quite much time learning and digging into the details of iOS programming.

### Searching and testing SDKs

At the beginning, we have searched a few AR libraries. Since we are inexperienced with the AR field, we have no idea which one is most suitable for us. Finally, we have chosen the Qualcomm AR SDK because it has most features that we want to have in our project. However, the SDK is quite complicated at first sight and we do not understand how it works. We tried to compiled the sample files and solved a number of errors such as linking error.

### Simple trials and testings on iPad

The two iPads used in this project are the first Apple product we had ever touched. We were impressed by the user-friendliness of Apple devices. But before we can run our first app, we must install developer profile files to development. The whole process is not a simple task and we had spent some time on it.

### Server set up

We was planning to set up a server on CSE Department server, but during the process we found that it requests CSE VPN for outside world to connecting to servers with the department. However there is no support for SSL or CISCO VPN on iPad, so that we cannot connect our iPad to the department machine deriectly.

### Draft game design

The hardest part of this project is to implement a real game with the idea of remote augmented reality multiple users interaction.

# Chapter 9. Next goals

For the first term of our Final Year Project, we mainly research about Augmented Reality toolkit and focus on testing part. Much effort is also spent on getting familiar with iOS development. In the coming term, our team will concentrate to work on realizing the game part and improve existing features.

## 9.1 Stabilized camera tracking

As mentioned in experiment part, the marker tracking algorithm is yet to be stabilized. Our team will continue to work on improving the accuracy of the tracker detection and position approximation method. So that the virtual world can be constructed more accurately.

## 9.2 Network connection

The network connect is now using HTTP streaming technique, which requires reconnection each time. We will continue to investigate for a better way to communicate between clients.

## 9.3 User Interface

Apple Inc. emphasises much on UI design and user experience much. As we are now developing an application on iPad, it is nice to enhance user experience by improving the UI.

## 9.4 Assist with Ipad accessories

There are some assistant devices on iPad which we may also consider using in order to take advantage in the project, such as gyroscope, assisted GPS, ambient light sensor ,etc. We think these are great opportunity to extend our project to a better level if we try to make good use of these devices.

## 9.5 More on Game design and implementation

The game for the first semester is for demonstration of how the virtual information is exchanged. We will be adding more features and refine the game design in the next semester. In order to make the game more enjoyable, we will try to draft a game base on AR needs. User can experience how AR is helpful though playing the game.

## 9.6 Investigate possibility for more clients

The system is now designed to support two clients only. However it would seems more interesting to support more user at the same time. This many need to re-design part of the server and client program.

# Chapter 10. Acknowledgement

Heartfelt thanks must be expressed to our final year project supervisor: **Professor Michael Lyu**. for his valuable advices on our project. Professor Lyu has given us guidance though out the project process. Our project development would not been running as smooth without that.

Besides, we would also like to thank **Mr.Edward Yau** and **Mr. Un Tze Lung** in VIEW Lab for their very helpful assistance on setting up facilities for our project. We also thank for their brilliant ideas which really gave us some insights and inspirations.

# References

List of references:

[1] Daniel Chun-Ming Leung, Pak-Shing Au, Irwin King, and Edward Hon-Hei Yau. 2007. Remote augmented reality for multiple players over network. In *Proceedings of the international conference on Advances in computer entertainment technology (ACE '07)*. ACM, New York, NY, USA, 220-223. DOI=10.1145/1255047.1255094 http://doi.acm.org/10.1145/1255047.1255094

[2] R. Azuma, "A Survey of Augmented Reality," *Presence: Teleoperators and Virtual Environments.* vol. 6, no. 4, Aug. 1997, pp. 355-385.

[3] J.P. Rolland, L.D. Davis and Y. Baillot, "A Survey of Tracking Technologies for Virtual Environments," *Fundamentals of Wearable Computers and Augmented Reality,* W. Barfield and T. Caudell, eds., Lawrence Erlbaum, Mahwah, N.J.,2001, pp. 67-112.

[4] Taehee Lee, Hollerer, T., "Multithreaded Hybrid Feature Tracking for Markerless Augmented Reality", *Visualization and Computer Graphics, IEEE Transactions on,* On page(s): 355 - 368, Volume: 15 Issue: 3, May-June 2009

*[5]* FIALA, M.. ARTag, a fiducial marker system using digital techniques. In *Proc. of Computer Vision and Pattern Recognition, vol. 2*, 590–596 , 2005

[6] PINTARIC, T. 2003. An adaptive thresholding algorithm for the augmented reality toolkit. In *IEEE Int. Augmented Reality Toolkit Workshop*. ,2003

[7] Woohun Lee, Jun Park "Augmented foam: touchable and graspable augmented reality for product design simulation" *Bulletin of Japanese Society for the Design Science* (2006)

[8] *"Apple developer tool"* http://developer.apple.com/technologies/tools/

[9] *"Apple press info"* 2010 http://www.apple.com/pr/library/2010/09/01Apple-Introduces-New-iPod-touch.html

[10] Fielding, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

[11] *"openGL ES"* http://www.khronos.org/opengles/

# Appendix

## A1 Data for Camera-match moving experiment

Control A (less feature and small size) (x, y, z)

| | | | | | |
|---|---|---|---|---|---|
| 110.477478 | -1.051955 | 432.242523 | 91.276283 | 11.739739 | 432.74942 |
| 109.459114 | -0.710892 | 432.618286 | 87.844742 | 17.856461 | 433.430756 |
| 104.240768 | 1.003679 | 434.674713 | 89.05851 | 9.681972 | 433.218201 |
| 105.509697 | -0.904305 | 434.368073 | 92.722816 | 5.280146 | 431.909821 |
| 107.015648 | 0.05982 | 433.686218 | 89.110825 | 2.405899 | 433.572144 |
| 104.268211 | 1.163658 | 434.234894 | 87.57019 | 6.47157 | 432.896088 |
| 102.998291 | 9.548323 | 433.039734 | 83.488518 | 5.268199 | 433.312775 |
| 106.608894 | 5.405295 | 432.637634 | 85.274193 | 5.261956 | 432.155243 |
| 103.578857 | 8.173505 | 432.877441 | 81.566841 | -0.926027 | 433.482361 |
| 102.566032 | 6.77272 | 432.894379 | 82.417824 | -4.248106 | 433.90802 |
| 99.57106 | 5.026145 | 433.810394 | 78.450554 | -2.595711 | 435.141815 |
| 99.376488 | 2.031363 | 433.405396 | 81.374527 | -2.821459 | 433.853668 |
| 98.284508 | 11.448357 | 433.261688 | 81.41684 | 1.78176 | 433.318604 |
| 95.898254 | 7.09449 | 434.151825 | 80.089737 | 4.79507 | 433.205902 |
| 96.064774 | 3.571101 | 434.59201 | 77.269791 | 7.875554 | 433.569672 |
| 91.419708 | 2.234519 | 435.566925 | 76.022392 | 4.615079 | 434.427826 |
| 88.746315 | 1.633193 | 435.686157 | 77.212021 | 6.642534 | 432.89505 |
| 89.755013 | 2.613689 | 434.624054 | 74.825928 | 10.56168 | 433.133026 |
| 91.19622 | 9.377494 | 433.721466 | 75.729935 | 9.045829 | 432.652435 |

| | | | | | |
|---|---|---|---|---|---|
| 72.921532 | 3.773676 | 433.136536 | 70.847748 | -4.061156 | 431.253998 |
| 68.128464 | 2.730169 | 434.55899 | 64.580528 | -1.938621 | 432.384155 |
| 71.530319 | -2.293729 | 433.732666 | 74.140205 | -5.449933 | 429.901062 |
| 65.464394 | -5.413494 | 435.50943 | 71.065384 | -1.60675 | 429.947174 |
| 62.271088 | 4.276943 | 435.070129 | 75.42173 | -5.112259 | 429.188751 |
| 60.168419 | 9.839093 | 435.398438 | 66.3899 | 0.635282 | 430.952423 |
| 64.576904 | 8.174322 | 433.863892 | 58.21349 | 11.959829 | 431.324707 |
| 72.74147 | 3.747083 | 431.52243 | 63.813656 | 5.035847 | 431.214386 |
| 68.32357 | 6.916478 | 432.568756 | 55.795742 | 11.046373 | 431.331604 |
| 66.221947 | -2.988511 | 433.478882 | 61.507854 | 1.327659 | 430.389069 |
| 66.369675 | -3.235336 | 433.491211 | 47.600807 | -4.614494 | 433.959106 |
| 64.714973 | 5.50268 | 433.597046 | 60.051277 | -2.197291 | 430.428192 |
| 68.421974 | 2.84805 | 432.213287 | 54.781376 | -2.458156 | 431.508057 |
| 58.357826 | 4.812388 | 433.77063 | 66.227798 | -9.492278 | 429.853668 |
| 67.899948 | -3.946715 | 432.598297 | 61.823555 | 0.949532 | 430.837219 |
| 64.679169 | -1.340425 | 433.127045 | 52.454411 | 13.203216 | 431.370636 |
| 66.265114 | 2.234365 | 432.589661 | 59.510307 | 4.180457 | 430.342651 |
| 63.32502 | 2.492521 | 432.683533 | 44.774151 | -6.66012 | 434.051514 |
| 56.868473 | 8.018094 | 434.059601 | 51.474945 | -1.351916 | 431.815796 |
| 60.980114 | 1.224235 | 432.952454 | 54.281166 | -5.776793 | 432.23645 |

| | | | | | |
|---|---|---|---|---|---|
| 60.122196 | -1.464349 | 430.216766 | 95.746254 | -3.047214 | 419.120667 |
| 62.195911 | -1.326753 | 429.968628 | 59.178291 | -1.535937 | 430.258026 |

**Control B ( more feature and small size) (x, y, z)**

| | | |
|---|---|---|
| -198.272629 | 14.658869 | 987.771667 |
| -194.668488 | 12.451772 | 989.347656 |
| -195.429199 | 15.191667 | 989.204773 |
| -195.274033 | 17.35483 | 989.367004 |
| -196.484787 | 15.499326 | 988.889954 |
| -196.539291 | 21.028135 | 988.837219 |
| -194.740952 | 21.197088 | 989.90155 |
| -196.732056 | 17.249699 | 989.027283 |
| -196.844009 | 15.82623 | 989.038208 |
| -196.789413 | 15.713705 | 989.099365 |
| -196.862411 | 15.511185 | 988.771912 |
| -199.315933 | 16.319883 | 987.60144 |
| -199.996933 | 15.563679 | 987.338379 |
| -202.486694 | 15.194979 | 986.787109 |
| -202.486694 | 15.194979 | 986.787109 |
| -201.02063 | 18.840805 | 986.815857 |
| -197.655853 | 16.598394 | 988.055481 |

| | | |
|---|---|---|
| -203.390396 | 15.325605 | 986.602356 |
| -200.111237 | 17.915834 | 987.634644 |
| -194.405014 | 20.302979 | 989.927734 |
| -195.547424 | 16.862757 | 989.510315 |
| -194.848328 | 20.305643 | 989.845825 |
| -203.239914 | 23.420355 | 987.621338 |
| -202.850708 | 17.934452 | 989.056091 |
| -198.577621 | 36.89679 | 990.902405 |
| -189.033325 | 51.89996 | 992.683655 |
| -184.816803 | 60.586597 | 992.891174 |
| -180.784195 | 54.80616 | 993.857849 |
| -180.385605 | 48.273399 | 994.960571 |
| -192.807373 | 60.844589 | 991.232422 |
| -196.606644 | 65.971207 | 990.702881 |

| | | |
|---|---|---|
| -202.534119 | 60.696651 | 988.643127 |
| -208.073639 | 61.238132 | 987.998596 |
| -207.67804 | 67.716698 | 988.125488 |
| -209.341263 | 43.111534 | 989.828796 |
| -209.648071 | 38.240093 | 989.536865 |
| -214.787903 | 27.436312 | 990.422058 |
| -213.234879 | 32.703583 | 989.951782 |
| -208.724457 | 15.580491 | 989.749084 |
| -207.064133 | 26.828661 | 990.884827 |
| -195.166626 | 3.908515 | 994.513977 |
| -189.023361 | 7.266284 | 995.065308 |
| -189.023361 | 7.266284 | 995.065308 |
| -196.306091 | 16.774368 | 990.71936 |
| -199.517654 | -3.700207 | 986.289185 |
| -178.597183 | 2.310799 | 992.165222 |
| -173.88028 | 6.890123 | 992.277222 |
| -183.380188 | 18.108349 | 989.573792 |

| | | |
|---|---|---|
| -183.757706 | 0.056036 | 987.689697 |
| -167.7668 | -2.827701 | 992.920166 |
| -168.273315 | 4.990637 | 991.888489 |
| -171.628754 | 2.41215 | 989.505676 |
| -171.628754 | 2.41215 | 989.505676 |
| -158.923126 | -28.333313 | 992.128967 |
| -154.828979 | 6.567336 | 993.096863 |
| -143.193604 | -5.456466 | 994.656311 |
| -148.747345 | -3.452055 | 992.690247 |
| -147.821335 | 8.963762 | 992.324341 |
| -135.263168 | -16.427507 | 994.324097 |
| -140.434982 | 18.371777 | 994.817383 |
| -132.361191 | -6.941059 | 995.285217 |
| -135.552582 | 3.907674 | 994.554871 |
| -143.723862 | 7.569404 | 989.86969 |
| -137.196808 | 24.264273 | 990.51709 |
| -151.820435 | 11.15632 | 986.844604 |
| -139.688202 | 4.163665 | 989.307983 |
| -143.117722 | 6.792896 | 988.050781 |

| | | |
|---|---|---|
| -139.082031 | 12.861346 | 987.874634 |
| -130.318604 | 17.620604 | 989.345825 |
| -123.163521 | 33.25174 | 990.281067 |
| -125.346024 | 18.092115 | 989.089661 |
| -129.579926 | 13.100311 | 987.860046 |
| -123.926033 | 1.532969 | 989.439026 |
| -116.853958 | -1.640732 | 989.67572 |

| | | |
|---|---|---|
| -118.988045 | 9.50528 | 990.92157 |
| -125.740425 | 13.615618 | 987.854553 |
| -128.466461 | 16.19046 | 987.085938 |
| -126.222923 | 48.272518 | 984.318237 |
| -123.192932 | 55.322441 | 983.606567 |
| -129.172699 | 30.025593 | 983.496765 |
| -108.046127 | 11.45941 | 987.720825 |

## Control C ( less feature and large size) (x, y, z)

| | | | | | |
|---|---|---|---|---|---|
| 24.330294 | 2.275857 | 109.76297 | 20.497498 | 1.503682 | 109.80806 |
| 24.139803 | 2.175428 | 109.758232 | 20.433273 | 1.436239 | 109.769226 |
| 23.984015 | 2.156774 | 109.750145 | 20.160255 | 1.383654 | 109.773155 |
| 23.7586 | 2.099922 | 109.710426 | 20.129274 | 1.26963 | 109.713417 |
| 23.386118 | 2.076697 | 109.712173 | 19.984701 | 1.196079 | 109.708778 |
| 23.171211 | 2.16098 | 109.724792 | 19.733242 | 1.289612 | 109.711998 |
| 22.853199 | 1.994944 | 109.816246 | 19.736597 | 1.083162 | 109.710114 |
| 22.436581 | 1.776442 | 109.873016 | 19.213692 | 1.021827 | 109.798866 |
| 22.234819 | 1.794255 | 109.833244 | 18.760691 | 0.894213 | 109.837372 |
| 22.123867 | 1.898803 | 109.786125 | 18.35668 | 0.760912 | 109.821861 |
| 21.647007 | 1.624271 | 109.878113 | 18.140129 | 0.483858 | 109.846535 |
| 21.480869 | 1.779129 | 109.884056 | 17.857403 | 0.671949 | 109.823166 |
| 21.647556 | 1.579714 | 109.751419 | 17.488462 | 0.498975 | 109.881622 |
| 21.795397 | 1.625604 | 109.646019 | 17.619564 | 0.184606 | 109.894852 |
| 21.514215 | 1.598525 | 109.696198 | 17.562515 | 0.110327 | 109.822342 |
| 21.39892 | 1.572283 | 109.738731 | 17.176023 | 0.142596 | 109.8591 |
| 21.028013 | 1.464647 | 109.735329 | 17.176023 | 0.142596 | 109.8591 |
| 20.96489 | 1.555387 | 109.668564 | 16.511292 | 0.219135 | 109.802284 |
| 20.647526 | 1.46536 | 109.722214 | 16.119099 | 0.346965 | 109.882309 |

| | | |
|---|---|---|
| 15.80343 | 0.399346 | 109.955574 |
| 15.456814 | 0.360774 | 109.997253 |
| 15.056875 | 0.199396 | 109.984909 |
| 14.734602 | 0.077543 | 110.003204 |
| 14.528996 | 0.042082 | 110.013542 |
| 14.351627 | 0.062507 | 109.998245 |
| 14.153876 | -0.027513 | 110.000175 |
| 14.09222 | -0.381539 | 109.978668 |
| 13.745444 | -0.314582 | 110.017052 |
| 13.654635 | -0.34922 | 110.000099 |
| 13.392112 | -0.330366 | 109.955048 |
| 13.351233 | -0.350556 | 109.913849 |
| 13.1627 | -0.486077 | 109.876595 |
| 12.560557 | -0.220784 | 109.995857 |
| 12.076767 | -0.112317 | 110.063522 |
| 12.050927 | -0.289107 | 110.062119 |
| 11.799869 | -0.530049 | 110.068344 |
| 11.628817 | -0.631925 | 110.05793 |
| 11.359043 | -0.599416 | 110.056 |
| 11.013344 | -0.679529 | 110.115341 |

| | | |
|---|---|---|
| 10.733667 | -0.687962 | 110.133324 |
| 10.609989 | -0.809555 | 110.11557 |
| 10.735901 | -0.944562 | 110.084999 |
| 10.471407 | -0.901348 | 110.095505 |
| 10.120726 | -1.018848 | 110.134651 |
| 9.777616 | -0.875391 | 110.15361 |
| 9.629586 | -0.803828 | 110.148453 |
| 9.360086 | -1.238129 | 110.170746 |
| 9.215748 | -1.306462 | 110.176102 |
| 9.347125 | -1.28999 | 110.111107 |
| 9.061208 | -1.362825 | 110.169861 |
| 8.602625 | -1.299124 | 110.229591 |
| 8.263205 | -1.178087 | 110.275024 |
| 8.090117 | -1.256896 | 110.245247 |
| 8.116515 | -1.318458 | 110.176643 |
| 7.742511 | -1.602351 | 110.246254 |
| 7.805002 | -1.545028 | 110.235077 |
| 7.349963 | -1.42582 | 110.220779 |
| 7.145616 | -1.488717 | 110.244179 |
| 6.993147 | -1.561677 | 110.25898 |

| | | |
|---|---|---|
| 6.730315 | -1.743297 | 110.294594 |
| 6.457925 | -1.747366 | 110.224182 |
| 6.332412 | -1.897942 | 110.271935 |

**Control D ( more feature and large size ) (x, y, z)**

| | | |
|---|---|---|
| 58.799316 | 7.152175 | 276.409393 |
| 58.981915 | 6.656788 | 276.339508 |
| 58.953674 | 7.737742 | 276.255829 |
| 60.173054 | 9.432476 | 276.281891 |
| 58.831112 | 9.857171 | 276.337799 |
| 59.512722 | 9.95559 | 276.248444 |
| 60.242481 | 10.828683 | 276.265564 |
| 58.963844 | 11.315291 | 276.505249 |
| 58.5406 | 10.360284 | 276.430481 |
| 59.157227 | 10.873916 | 276.218018 |
| 57.664986 | 11.15496 | 276.439178 |
| 55.69326 | 11.173345 | 276.503815 |
| 55.067596 | 10.956958 | 276.561157 |
| 54.326786 | 11.272959 | 276.462677 |
| 52.706406 | 10.910975 | 276.622253 |
| 51.846382 | 10.81286 | 276.599335 |
| 51.545372 | 10.507464 | 276.614136 |
| 51.662586 | 11.041578 | 276.572235 |
| 51.108528 | 11.063889 | 276.488129 |

| | | |
|---|---|---|
| 50.678505 | 11.100348 | 276.340759 |
| 49.922031 | 10.840047 | 276.363159 |
| 49.461285 | 10.703835 | 276.421875 |
| 48.874603 | 10.485568 | 276.509521 |
| 48.963501 | 10.472559 | 276.412628 |
| 48.003803 | 11.003648 | 276.313538 |
| 46.892723 | 10.626365 | 276.315491 |
| 46.297382 | 10.235674 | 276.385864 |
| 44.961941 | 10.947588 | 276.514465 |
| 44.473675 | 10.661399 | 276.469604 |
| 44.632584 | 10.263469 | 276.476685 |
| 43.470875 | 10.257119 | 276.441193 |
| 43.607132 | 9.610999 | 276.536652 |
| 42.903957 | 9.132807 | 276.521332 |
| 41.66114 | 8.951489 | 276.536041 |
| 40.802425 | 8.54132 | 276.595551 |
| 40.449413 | 8.788774 | 276.664764 |
| 39.170158 | 8.618882 | 276.72937 |
| 38.471893 | 8.374916 | 276.616364 |
| 37.933109 | 8.354231 | 276.691223 |

| | | |
|---|---|---|
| 37.149055 | 8.786007 | 276.760406 |
| 36.643059 | 8.64287 | 276.841949 |
| 35.71273 | 8.302925 | 276.784241 |
| 33.957302 | 8.528069 | 276.930023 |
| 33.620388 | 8.687315 | 276.834198 |
| 32.238892 | 8.322104 | 276.880554 |
| 31.547552 | 7.522537 | 276.95462 |
| 30.750521 | 7.003766 | 276.917603 |
| 29.517263 | 7.596526 | 276.821686 |
| 28.30455 | 7.540428 | 276.855591 |
| 27.589437 | 7.127538 | 276.903168 |
| 26.718529 | 6.672648 | 277.027985 |
| 25.108902 | 7.273698 | 276.881653 |
| 24.996708 | 7.129953 | 276.773315 |
| 24.698895 | 7.630898 | 276.835999 |
| 23.182724 | 7.782124 | 276.886414 |
| 22.239967 | 7.845 | 276.908203 |
| 20.151537 | 8.378735 | 277.011658 |
| 19.042362 | 8.481663 | 276.996918 |
| 17.506901 | 7.042542 | 277.038727 |

| | | |
|---|---|---|
| 17.386066 | 7.319031 | 276.79718 |
| 15.516594 | 7.41418 | 276.974426 |
| 14.722754 | 8.217031 | 276.940765 |
| 13.658537 | 7.858841 | 276.99823 |
| 11.497663 | 8.638164 | 277.026093 |
| 12.164455 | 7.725625 | 277.097473 |
| 10.932909 | 7.581668 | 277.072632 |
| 10.140691 | 7.364721 | 276.970337 |
| 9.571096 | 7.396454 | 277.024139 |
| 8.661355 | 6.999103 | 277.065704 |
| 8.025467 | 7.302996 | 277.062439 |
| 6.464575 | 7.311372 | 277.078766 |
| 6.373775 | 6.784213 | 277.037323 |
| 4.988535 | 6.386417 | 277.122009 |
| 3.017162 | 7.099211 | 277.121948 |
| 2.760309 | 6.035836 | 277.095764 |
| 1.892072 | 5.543664 | 277.192352 |
| 0.31218 | 5.559417 | 277.192444 |
| -1.251834 | 5.906584 | 277.220978 |
| -2.051293 | 5.709883 | 277.169708 |

| | | |
|---|---|---|
| -2.708414 | 5.68773 | 277.178864 |
| -3.337235 | 5.498232 | 277.300781 |