5/22/2010

TI
UNJANI

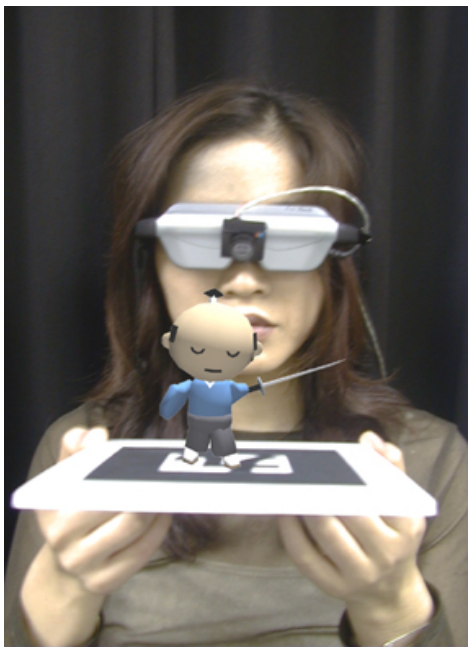
# PENGENALAN AUGMENTED REALITY

Aceng Sobana
2010

# Daftar Isi

## 1 Pengenalan AR

AR merupakan suatu konsep perpaduan antara *virtual reality* dengan *world reality*. Sehingga obyek-obyek *virtual* 2 Dimensi (2D) atau 3 Dimensi (3D) seolah-olah terlihat nyata dan menyatu dengan dunia nyata. Menurut Azuma, Augmented Reality adalah variasi dari Virtual Reality. Pada teknologi *Virtual Reality*, pengguna berinteraksi dengan lingkungan yang diciptakan secara virtual yang merupakan simulasi dunia nyata, akan tetapi pengguna tidak bisa melihat dunia nyata yang ada di sekelilingnya. Pada teknologi AR, pengguna dapat melihat dunia nyata yang ada di sekelilingnya dengan penambahan obyek virtual yang dihasilkan oleh komputer. Supaya obyek AR 3D terlihat langsung pada medianya, maka diperlukan alat khusus yang disebut dengan *Head Mounted Display* (HMD).



**Gambar 1.1 Contoh Penggunaan AR**

## 2 ARToolKit

Perhitungan yang tepat merupakan hal yang sangat penting dalam teknologi AR untuk menempatkan obyek 3D yang dihasilkan komputer sehingga seolah-olah dari sudut pandang *user* berada pada dunia nyata.

ARToolKit adalah salah satu pustaka (*library*) perangkat lunak berbasis C dan C++ yang menggunakan metoda *computer vision tracking* untuk menghitung posisi kamera dan orientasinya yang relatif terhadap *marker*. ARTooKit dikembangkan oleh Dr. Hirokazu Kato dari Universitas Osaka Jepang dan Mark Billinghurst dari *Human Interface Technology Laboratory* (HIT Lab). ARToolKit banyak digunakan untuk mengembangkan aplikasi AR. Marker pada ARToolkit merupakan gambar yang terdiri atas *border outline* dan *pattern image* seperti terlihat pada Gambar 2.1.

**Gambar 2.1 Contoh *marker***

Saat ini ARToolKit bisa berjalan pada Sistem Operasi SGI IRIX, PC Linux, Mac OS X, dan PC Windows (95/98/NT/2000/XP). Fungsi setiap versi ARToolKit adalah sama, tetapi performanya bisa berbeda-beda tergantung dari konfigurasi hardware yang digunakan.

## 2.1 Instalasi

Pada intinya, ARToolKit adalah sebuah software library, yang dirancang untuk dapat dihubungkan ke dalam program aplikasi. Supaya aplikasi yang menggunakan ARToolKit bisa berjalan dengan baik, maka diperlukan beberapa peralatan diantaranya webcam yang sudah terinstall di komputernya, beberapa marker dan juga harus sudah terintsall DirectX minimal versi 9. Library lainya yang harus disiapkan diantaranya glut (*Graphic Library Utility Toolkit*).

Adapun langkah-langkah instalasi ARToolKit diantaranya:

- Buat folder AR di drive C.
- Simpan ARToolkit-2.72.1-bin-win32.zip dan glut-3.7.6-bin.zip ke folder AR.
- Ekstrak ARToolkit-2.72.1-bin-win32.zip ke drive C. Sehingga nanti ada folder ARToolKit di drive C.
- Ekstrak glut-3.7.6-bin.zip pada folder AR sehingga ada folder glut-3.7.6-bin pada folder AR.
- Copy file glut32.dll dari folder glut-3.7.6-bin ke folder C:\ARToolKit\bin.
- Buat folder C:\ARToolKit\include\GL. Copy file glut.h dari folder glut-3.7.6-bin ke folder C:\ARToolKit\include\GL.
- Copy file glut32.lib dari folder glut-3.7.6-bin ke folder C:\ARToolKit\lib.

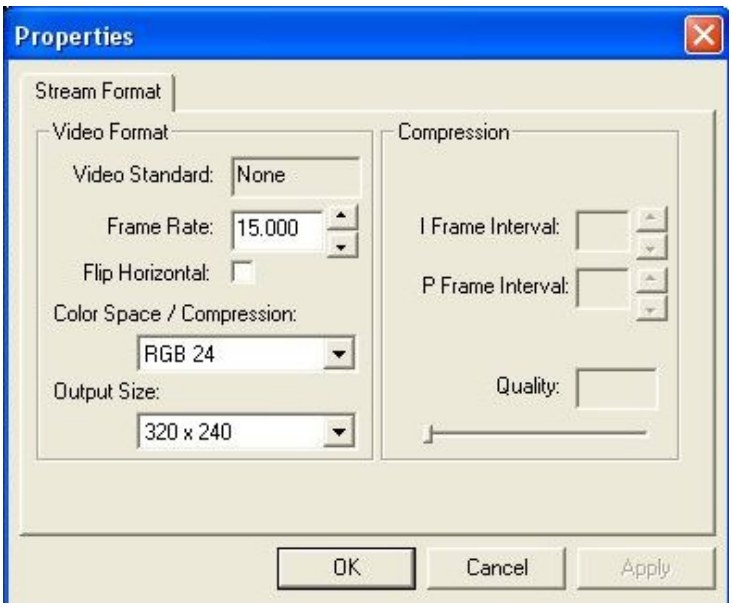
## 2.2 Menjalankan Sample Aplikasi ARToolKit

Sebelum mencoba menjalankan contoh aplikasi ARToolKit, yakinkan bahwa webcam bekerja dengan baik dan marker sudah disiapkan, jika markernya belum ada, bisa di print terlebih dahulu marker yang ada pada folder "pattern". Akan sangat baik jika marker dicetak atau ditempel pada bahan yang kaku dan tidak mudah melipat seperti pada triplek. Sebelum menggunakan marker dan webcam, alangkah baiknya jika dicoba dulu kapabilitas grafis dari ARToolkit yang kita gunakan. Untuk mengetesnya jalankan graphicsTest.exe pada folder bin yang ada di dalam ARToolKit. Jika berfungsi dengan sempurna, maka akan muncul gambar teapot berputar seperti terlihat di bawah ini:

**Gambar 2.2 Teapot berputar**

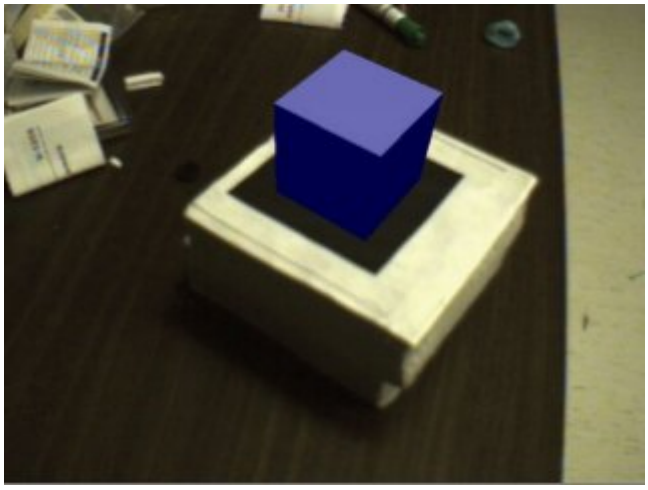
Kemudian tes juga webcam yang terpasang, apakah sudah disupport oleh ARToolKit atau belum dengan menjalankan videoTest.exe yang ada di dalam folder bin.

Jika marker sudah siap, jalankan aplikasi simpleTest.exe. Konsol dos akan muncul, dan ketika kamera terdeteksi, maka akan muncul windows dialog seperti gambar di bawah.




**Gambar 2.3 Memilih konfigurasi webcam**

Pada windows dialog tersebut anda bisa pilih resolusi dan *frame rate* yang diinginkan. Arahkan marker Hiro ke kamera. Jika berjalan dengan baik, akan terlihat kubus yang jika dilihat dalam layar seolah-oleh berada di atas marker. Cobalah putar atau pindahkan posisi marker mendekati atau menjauhi kamera, dan anda bisa lihat apa yang terjadi.

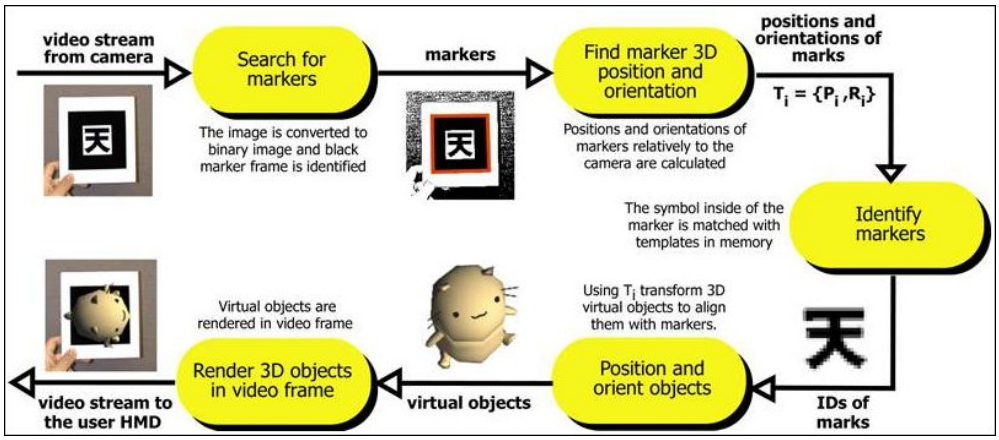


**Gambar 2.4 Kubus virtual terlihat diatas marker**

Supaya kubus virtual terlihat, marker harus berada dalam area yang dapat disorot kamera. Jika kubus virtual terkadang muncul kemudian hilang, bisa jadi dipengaruhi oleh kondisi pencahayaan yang kurang bagus.

## 2.3 Cara Kerja

Cara kerja ARToolKit dapat dilihat pada **Error! Reference source not found.** di bawah ini.




**Gambar 2.5 Cara kerja ARToolKit.**

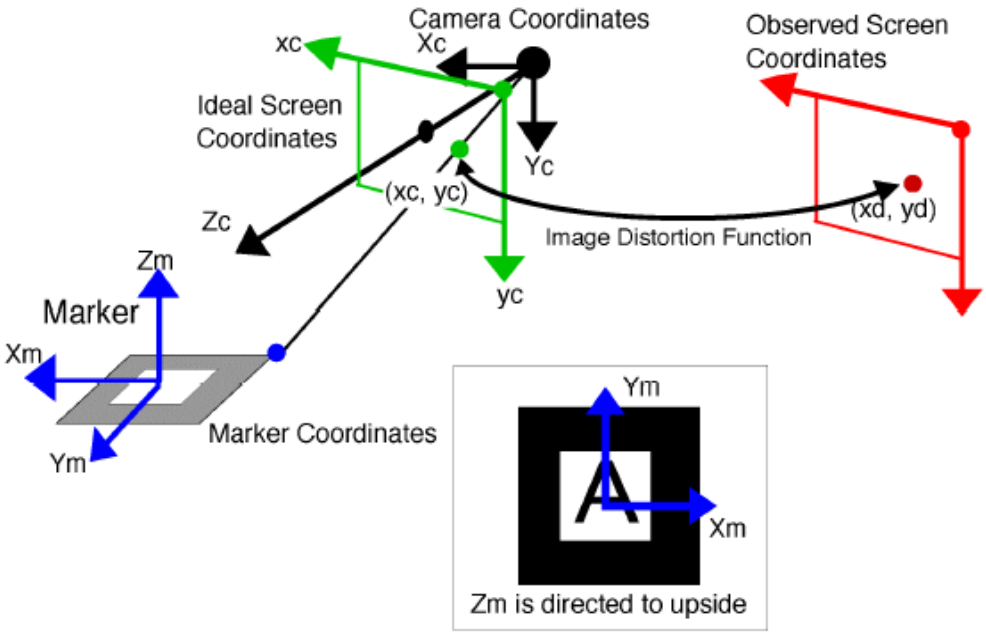
Secara umum prinsip kerja artoolkit adalah sebagai berikut.

1. Kamera menangkap gambar dari dunia nyata secara *live* dan mengirimkannya ke komputer.
2. Perangkat lunak dalam komputer akan mencari *marker* pada masing-masing frame video.
3. Jika *marker* telah ditemukan, komputer akan memproses secara matematis posisi relatif dari kamera ke kotak hitam yang terdapat pada *marker*.
4. Apabila posisi kamera diketahui, maka model tersebut akan digambarkan pada posisi yang sama.
5. Model obyek 3D akan ditampilkan pada *marker*, artinya obyek virtual tersebut ditambahan pada dunia nyata.

Ada beberapa keterbatasan pada sistem AR ini. Objek virtual akan muncul jika markernya ada dalam kawasan yang bisa dilihat oleh kamera. Selain itu, jika ada bagian marker yang tertutup meski sedikit, misalnya terhalang oleh tangan, maka objek virtualnya akan hilang.

Masalah lain adalah masalah jangkauan dan masalah cahaya. Semakin kecil atau semakin jauh marker terhadap kamera, maka semakin kecil kemungkinan marker dapat dideteksi oleh kamera. Pantulan cahaya juga bisa membuat deteksi marker menjadi lebih sulit, oleh karena itu akan lebih baik jia marker dicetak pada media yang tidak memantulkan cahaya.

Ilustrasi sistem koordinat dalam ARToolKit ditunjukan oleh Gambar II-5.



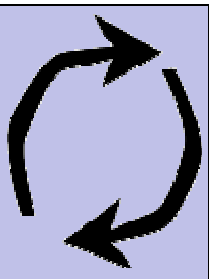

**Gambar 2.6 Sistem koordinat ARToolKit.**

# 3 Membuat Aplikasi AR

## 3.1 Prinsip Pengembangan

Langkah-langkah berikut ini harus diambil dalam aplikasi kode utama:

| | |
|---|---|
| **Inisialisasi** | 1. Menginisialisasi *video capture* dan membaca file marker dan parameter kamera. |
| **Loop** | 2. Ambil video frame input. |
| | 3. Mendeteksi pola marker dan pola gambar dalam kotak hitam pada frame video. |

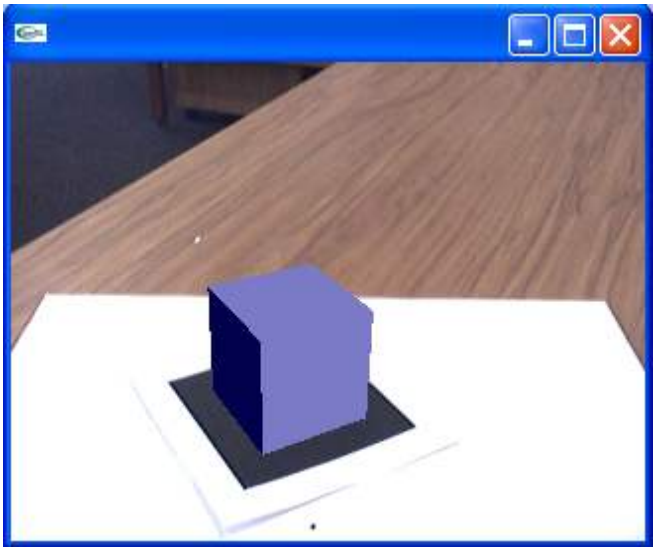
| | |
|---|---|
|  **Utama** | 4. Hitung transformasi kamera relatif terhadap pola yang terdeteksi. |
| | 5. Menggambar objek virtual pada pola yang terdeteksi. |
| **Penutupan** | 6. Tutup video capture. |

Langkah 2 sampai 5 diulang terus menerus sampai aplikasi tersebut berhenti, sementara langkah 1 dan 6 hanya dilakukan pada inisialisasi dan shutdown dari aplikasi masing-masing.

## 3.2 Membuat Aplikasi AR Pertama

### 3.2.1 Pendahuluan

Untuk menunjukkan secara terperinci bagaimana mengembangkan sebuah aplikasi ARToolKit, kita akan mulai membuat project baru pada MS Visual Studio 2003 dan melihat kode sumber simpleTest.c yang ada dalam direktori example/simpleTest/.



**Gambar 3.1 Program SimpleTest**

File yang akan kita lihat adalah simpleTest.c. Program ini hanya terdiri dari main rutin dan beberapa rutin graphic utama.

Fungsi yang sesuai dengan aplikasi enam langkah yang dijelaskan sebelumnya diperlihatkan pada Tabel 3-1. Fungsi yang sesuai dengan langkah 2 sampai 5 ada dalam fungsi mainLoop.

| ARToolKit Step | Fungsi |
|---|---|
| 1. Menginisialisasi aplikasi | `init` |
| 2. Ambil sebuah frame video input | `arVideoGetImage` (dalam mainLoop) |
| 3. Mendeteksi penanda | `arDetectMarker` (dalam mainLoop) |
| 4. Hitung kamera transformasi | `arGetTransMat` (dalam mainLoop) |
| 5. Menggambar objek virtual | `draw` (dalam mainLoop) |
| 6. Tutup video capture bawah | `cleanup` |

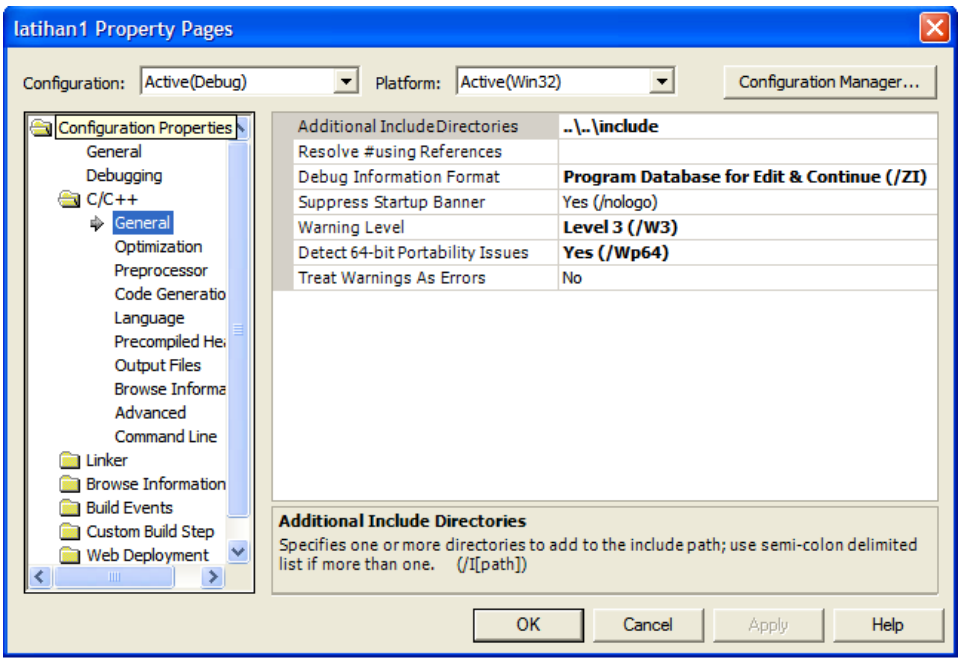
**Tabel 3-1 Fungsi panggilan dan kode yang sesuai dengan langkah-langkah aplikasi ARToolKit.**

Fungsi yang paling penting dalam program ini adalah `main`, `init`, `mainLoop` `draw` dan `cleanup`. Pembahasan fungsi-fungsi tersebut akan dibahsa dalam bagian khusus yang lebih rinci.

Langkah-langkah dalam membuat aplikasi AR menggunakan ARToolKit adalah sebagai berikut:

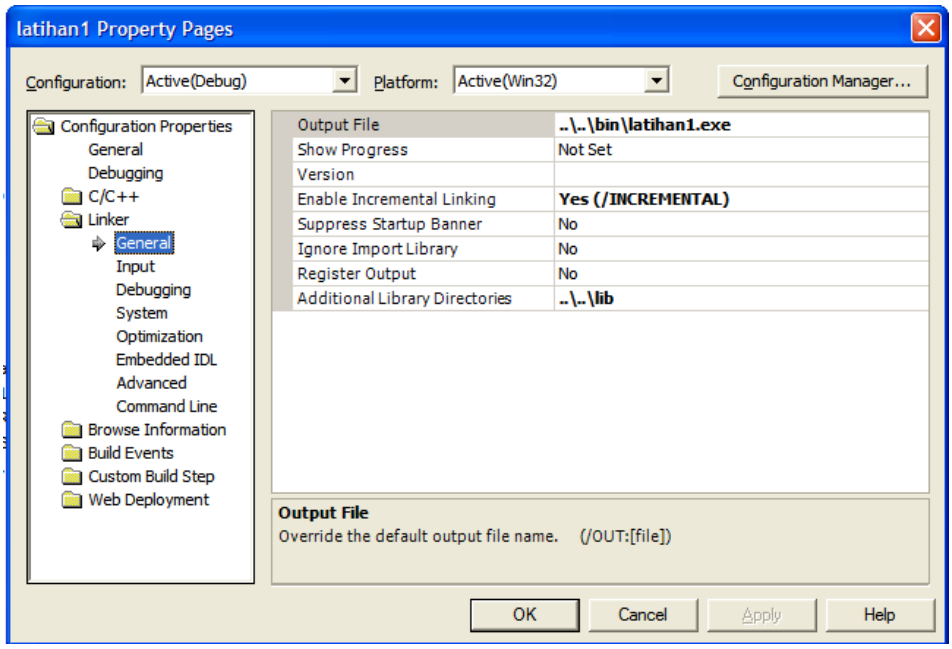
- Buka MS Visual Studio 2003
- Klik File dan Open Project kemudian arahkan ke folder C:\ARToolKit, dan pilih ARToolKit.sln
- Klik kanan pada solution ARToolKit, kemudian Add New Project. Beri nama project yang akan kita buat dengan nama latihan1, dan arahkan Location nya ke dalam folder example. Kemudian klik OK yang diiukuti oleh window Win 32 Application Wizard.
- Pada window Win 32 Application Wizard, pilih Application Setting, kemudian ceklis Empty Project. Kemudian klik Finish.
- Pada Solution ARToolKit akan ada project baru dengan nama latihan1. Klik kanan source file pada project latihan1 kemudian Add New Item. Pilih C++ file kemudian beri nama filenya latihan1.cpp. Klik Open.
- Klik dua kali file simpleTest.c pada folder Source File project simpleTest.
- Copykan isi file simpleTest.c dengan menekan tombol Ctrl + A. Kemudian paste pada file latihan1.cpp
- Kemudian klik kanan pada project latihan1 dan pilih properties.

- Pada menu Configuration Properties, pilih C/C++. Kemudian Pilih General. Di kolom kanan pada Additional Include Directories isikan ..\..\include.



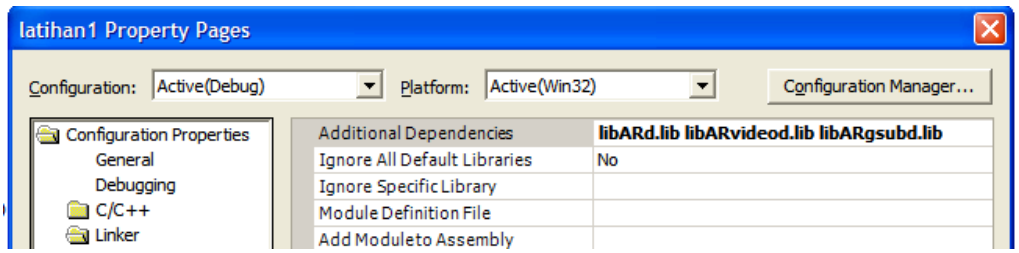

Gambar 3.2 Konfigurasi Project

- Pada menu Configuation Properties, pilih Linker. Kemudian Pilih General. Di kolom kanan pada Output File isikan ..\..\bin. Pada Additional Library Directories isikan ..\..\lib.




Gambar 3.3 Konfigurasi Project

- Masih pada Linker. Pilih Input. Di kolom kanan pada Additional Dependencies isikan **libARd.lib libARvideod.lib libARgsubd.lib**. Kemudian klik OK.




**Gambar 3.4 Konfigurasi Project**

- Klik kanan pada project latihan1, kemudian pilih Project Only dan pilih Rebuild only latihan1

- Jika proses kompilasi berhasil, maka di dalam folder bin akan ada latihan1.exe. Coba jalankan file tersebut untuk melihat hasilnya. Gunakan marker Hiro.

### 3.2.2 main

Coba anda buka file latihan1.cpp, disana ada kode main routine seperti terlihat dibawah:

```
main(int argc, char *argv[])
{
    init();
    arVideoCapStart();
    argMainLoop( NULL, keyEvent, mainLoop );
}
```

Rutin init digunakan untuk inisialisasi video capture, membaca file marker, parameter kamera dan setup graphic window (langkah 1 seperti dalam Tabel 3-1). Selanjutnya fungsi `arVideoCapStart` akan menghidupkan kamera. Fungsi argMainLoop kemudian dipanggil yang akan menjalankan fungsi keyEvent dan fungsi mainLoop yang akan melakukan render secara terus menerus (looping). Definisi fungsi `argMainLoop` ada dalam file gsub.c.

### 3.2.3 init

Rutin init dipanggil dalam main rutin dan digunakan untuk menginisialisasi video capture dan membaca parameter-parameter aplikasi ARToolKit.

Pertama-tama, membuka path video, kemudian mencari ukuran video:

```
/* open the video path */
if( arVideoOpen( vconf ) < 0 ) exit(0);
```

```
/* find the size of the window */
if( arVideoInqSize(&xsize, &ysize) < 0 ) exit(0);
printf("Image size (x,y) = (%d,%d)\n", xsize, ysize);
```

Variable vconf berisi konfigurasi video yang di definisikan pada bagian atas dari file latihan1.cpp. Selanjutnya kita memerlukan parameter yang akan digunakan dalam aplikasi ARToolKit, diantaranya:

- pola yang akan digunakan.
- Karakteristik kamera yang digunakan.

Parameter kamera dibaca dari file Data/camera_para.dat yang merupakan parameter default:

```
/* set the initial camera parameters */
if( arParamLoad(cparaname, 1, &wparam) < 0 ) {
    printf("Camera parameter load error !!\n");
    exit(0);
}
```

Selanjutnya, parameter-parameter tersebut ditransformasikan dengan ukuran gambar yang ditangkap kamera.

```
arParamChangeSize( &wparam, xsize, ysize, &cparam );
```

Kemudian aplikasikan parameter kamera, dan tampilkan parameter tersebut pada konsol:

```
arInitCparam( &cparam );
printf("*** Camera Parameter ***\n");
arParamDisp( &cparam );
```

Kemudian baca definisi pola yang digunakan, dalam hal ini pola yang digunakan adalah Hiro dari file Data/patt.hiro:

```
if( (patt_id=arLoadPatt(patt_name)) < 0 ) {
    printf("pattern load error !!\n");
    exit(0);
}
```

patt_id merupakan id pola (*marker*) yang diidentifikasi.

Buka graphic window:

```
/* open the graphics window */
argInit( &cparam, 1.0, 0, 0, 0, 0 );
```

Argumen kedua fungsi di atas digunakan sebagai fungsi zoom.

### 3.2.4 mainLoop

Rutin ini merupakan rutin dimana banyak fungsi ARToolKit dipanggil dan merupakan langkah 2 sampai langka 5 dari Tabel 3-1.

Pertama-tama melakukan capture frame video menggunakan fungsi `arVideoGetImage`:

```
/* grab a video frame */
if( (dataPtr = (ARUint8 *)arVideoGetImage()) == NULL ) {
    arUtilSleep(2);
    return;
}
```

Frame video yang dicapture tersebut kemudian digambarkan pada layar:

```
argDrawMode2D();
argDispImage( dataPtr, 0,0 );
```

Kemudian fungsi `arDetectMarker` digunakan untuk mencari pola yang sesuai dalam frame video yang tadi dicapture:

```
if(arDetectMarker(dataPtr, thresh, &marker info, &marker num)<0)
{
    cleanup();
    exit(0);
});
```

Pola yang ditemukan berisi variable marker_num, dan marker_info yang merupakan pointer yang berisi informasi koordinat dan confidence value serta object id setiap marker yang ditemukan. Struktur marker_info secara detail dapat dilihat dalam dokumentasi API ARToolKit.

Sampai titik ini, video image ditampilkan dilayar dan dianalisa. Sehingga frame tersebut tidak kita gunakan lagi, sehingga kita bisa panggil frame grabber untuk memulai mengcapture frame video selanjutnya. Untuk memulai frame grabber tersebut kita menggunakan fungsi `arVideoCapNext`:

```
arVideoCapNext();
```

Selanjutnya cek visibility pola marker:

```
/* check for object visibility */
k = -1;

for( j = 0; j < marker_num; j++ ) {
    if( patt_id == marker_info[j].id ) {
        if( k == -1 ) k = j;
        else if( marker_info[k].cf < marker_info[j].cf ) k = j;
```

```
    }
}

if( k == -1 ) {
    argSwapBuffers();
    return;
}
```

Transformasi antara marker dengan kamera bisa ditemukan dengan menggunakan fungsi `arGetTransMat`:

```
/* get the transformation between the marker and the real camera */
arGetTransMat(&marker_info[k],patt_center,patt_width,patt_trans);
```

Posisi ril kamera dan orientasi relative terhadap objek marker i merupakan matriks 3x4, patt_trans.

Selanjutnya, objek virtual dapat kita gambar dengan menggunakan fungsi:

```
draw();
argSwapBuffers();
```

Catatan: langkah optimisasi sederhana dilakukan jika tidak ada pola yang terdeteksi (k==-1), kita dapat langsung men swap buffer tanpa memanggil fungsi draw():

```
if( k == -1 ) {
    argSwapBuffers();
    return;
}
```

### 3.2.5 draw

Fungsi draw dibagi menjadi inisialisasi rendering, setup matrix dan render objek. Inisialisasi rendering 3D dilakukan dengan menggunakan beberapa fungsi pada ARToolKit dang beberapa fungsi OpenGL:

```
argDrawMode3D();
argDraw3dCamera( 0, 0 );
glClearDepth( 1.0 );
glClear(GL_DEPTH_BUFFER_BIT);
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LEQUAL);
```

Yang diperlukan selanjutnya adalah mengkonversi matriks transformasi yang dihasilkan (matriks 3x4) kedalam format (array of 16 values), menggunakan fungsi `argConvGlpara`. Nilai dari array 16 tersebut berupa posisi dan orientasi dari kamera ril, sehingga dengan posisi tersebut diaplikasikan kepada kamera, virtual objek yang dirender seolah-olah posisinya berada pada marker.

```
/* load the camera transformation matrix */
argConvGlpara(patt_trans, gl_para);
glMatrixMode(GL_MODELVIEW);
glLoadMatrixd( gl_para );
```

Posisi kamera virtual di set menggunakan fungsi OpenGL `glLoadMatrixd(gl_para)`. Bagian akhir dari kode dibawah ini merender objek 3D, yakni membuat kubus virtual dengan warna biru dibawah cahaya berwarna putih:

```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
glLightfv(GL_LIGHT0, GL_AMBIENT, ambi);
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightZeroColor);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_flash);
glMaterialfv(GL_FRONT, GL_SHININESS, mat_flash_shiny);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMatrixMode(GL_MODELVIEW);
glTranslatef( 0.0, 0.0, 25.0 );
glutSolidCube(50.0);
```

Selanjutnya, reset OpenGL variable menjadi default:

```
glDisable( GL_LIGHTING );
glDisable( GL_DEPTH_TEST );
```

Langkah-langkah yang dijelaskan di atas berjalan setiap waktu dalam rutin main rendering loop. Ketika program berjalan, mouse event di handle oleh fungsi mouseEvent dan keyboard event ditangani oleh fungsi keyEvent.

### 3.2.6 Cleanup

Fungsi cleanup dipanggil untuk memberhentikan video processing dan menutup video path sehingga device (kamera) bisa digunakan oleh aplikasi lainya:

```
arVideoCapStop();
arVideoClose();
argCleanup();
```

Keterbatasan program ini adalah hanya menggunakan satu pola (marker) yakni marker Hiro.

Langkah selanjutnya kita akan buat program yang akan mendeteksi lebih dari satu pola (marker).

## 3.3 Multimarker

Sekarang kita akan coba dengan menggunakan lebih dari satu pola (marker) dan merender objek 3D yang berbeda untuk masing-masing pola. Untuk keperluan ini kita akan lihat contoh loadMultiple yang ada dalam folder examples. Anda akan lihat dua buah source code yakni

loadmulti.c dan object.c. Program ini dapat mendeteksi multiple marker dan merender objek 3D yang berbeda pada tiap-tiap marker.

Perbedaan utama dengan program latihan1 diantaranya:

- load file dengan deklarasi multiple pattern di dalamnya.
- struktur baru untuk pola yang nantinya digunakan dalam program.
- Redefinisi sintak dan fungsi draw.

Kode yang lainya tetap sama dengan latihan1.

Untuk meload multiple pattern dalam ARToolKit, fungsi baru digunakan, yakni dalam file object.c, fungsi baru tersebut adalah read_ObjData. Dengan fungsi tersebut, meload marker sekarang dilakukan dengan cara:

```
if(  (object=read_ObjData(model_name,  &objectnum))  ==  NULL  )
exit(0);
printf("Objectfile num = %d\n", objectnum);
```

object merupakan pointer ke struktur ObjectData_T, struktur yang spesifik untuk mengelola daftar pola (*marker*). File teks object_data menspesifikasikan objek marker mana yang akan digunakan. File object_data dimulai dengan berapa jumlah objek marker yang akan digunakan kemudian struktur data untuk masing-masing objek. Setiap marker dalam object_data berisi struktur data sebagai berikut:

- Name
- Pattern Recognition File Name
- Width of tracking marker
- Center of tracking marker

Sebagai contoh:

```
#pattern 1
cone
Data/patt.hiro
80.0
0.0 0.0
```

Baris yang dimulai dengan karakter # menandakan komentar yang akan diabaikan.

ARToolKit sekarang dapat mengidentifikasi multiple pattern dalam rutin `arDetectMarker`. Visibility state dan translasinya untuk tiap objek kemudian diperiksa dan disimpan.

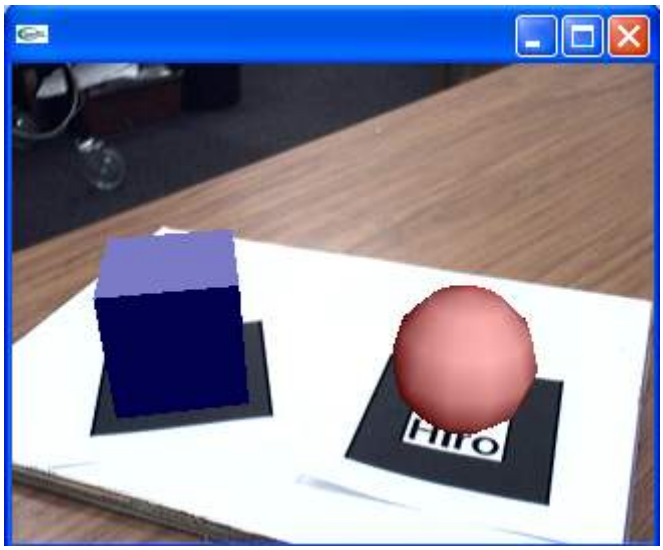
```
/* check for object visibility */
for( i = 0; i < objectnum; i++ ) {
    k = -1;
    for( j = 0; j < marker num; j++ ) {
        if( object[i].id == marker_info[j].id ) {
```

```
            if( k == -1 ) k = j;
            else if( marker_info[k].cf < marker_info[j].cf ) k =
j;
        }
    }

    if( k == -1 ) {
        object[i].visible = 0;
        continue;
    }

    object[i].visible = 1;
    arGetTransMat(&marker_info[k],
    object[i].marker_center, object[i].marker_width,
    object[i].trans);
}
```

Untuk merender objek 3D, tinggal panggil fungsi draw dengan parameter struktur ObjectData_T jumlah objeknya:

```
/* draw the AR graphics */
draw( object, objectnum );
```

Coba kompilasi loadMultiple dan pastikan file yang dibutuhkan ada dalam folder Data. Hasilnya akan terlihat seperti gambar di bawah.




**Gambar 3.5 loadMulti Video Window**

Anda dapat memodifikasi file object_data dan menggunakan pola (marker) yang lainya.

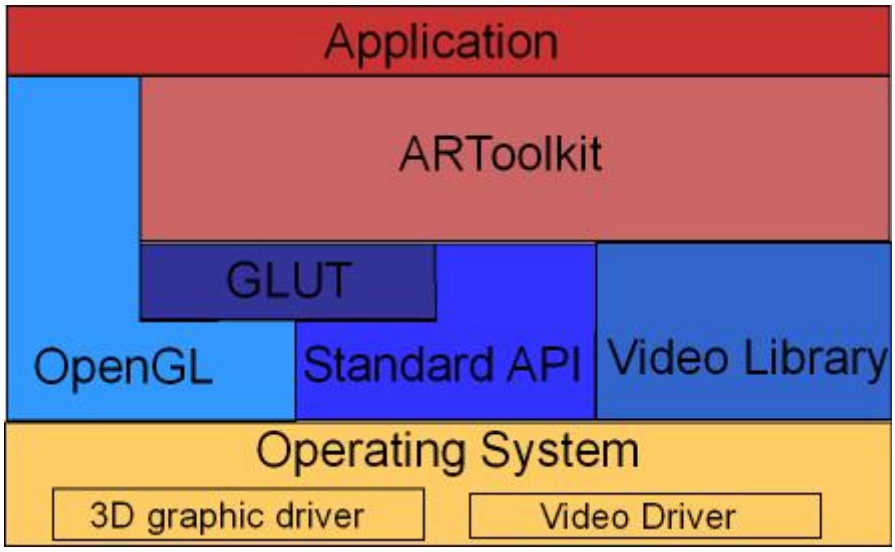
## 3.4 ARToolkit Framework: Deskripsi Umum

### 3.4.1 Pengantar

ARToolKit merupakan software *ToolKit* layaknya GLUT. Di dalamnya terdiri dari predefined functions yang bisa kita panggil (call) dalam mengembangkan aplikasi AR. ARToolKit menggunakan OpenGL untuk rendering, GLUT untuk aspek windows/event handler.

API ARToolKit ditulis dalam bahasa C dan disertakan pula beberapa contoh yang bisa dijadikan acuan dalam mengembangkan aplikasi AR.

Gambar di bawah mengilustrasikan hubungan antara aplikasi yang dibuat dengan ARToolKit dan library yang diperlukan.



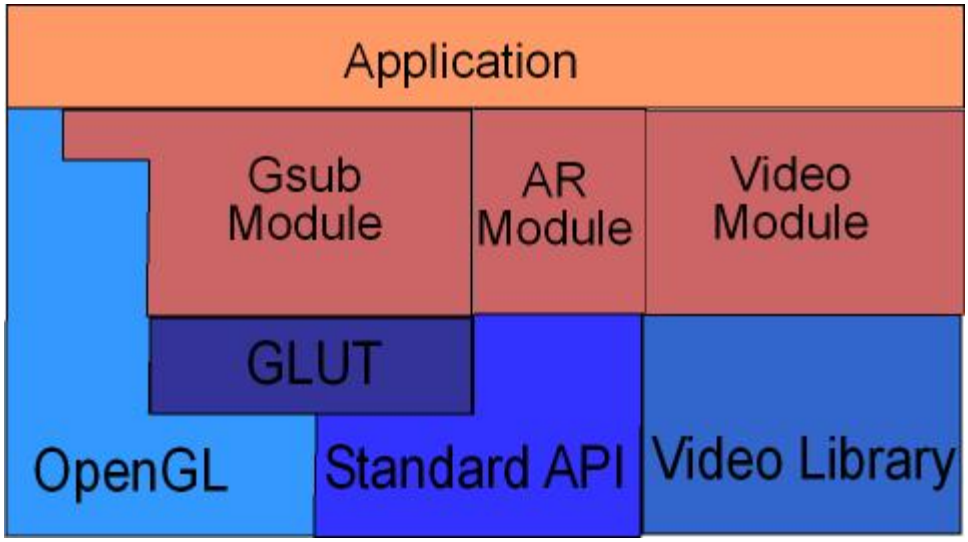

**Gambar 3.6 ARToolKit Architecture.**

### 3.4.2 Structure

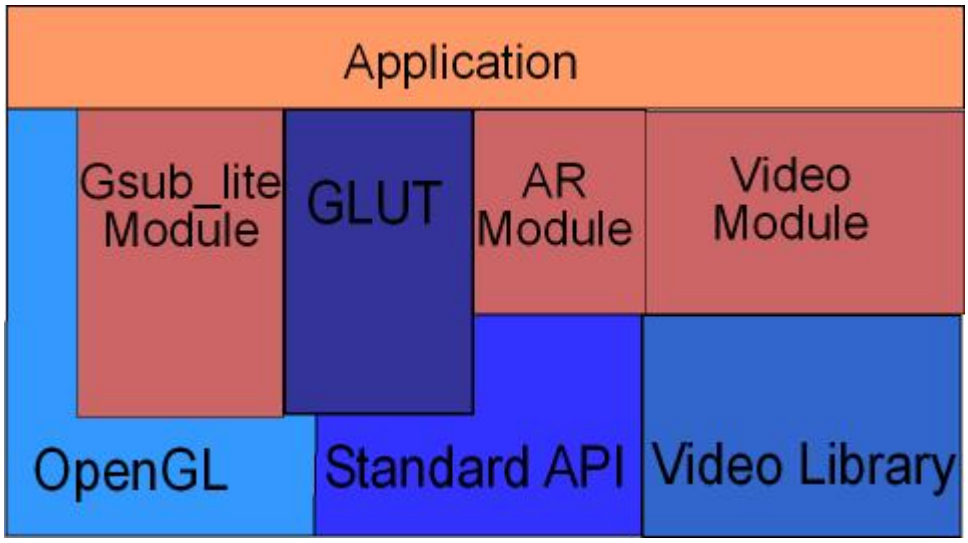
Librari ARToolKit library terdiri dari empat modul:

- AR module: modul utama yang berisi marker tracking routines, calibration dan parameter collection.
- Video module: berisi koleksi rutin untuk mengcapture frame input video.
- Gsub module: koleksi rutin grafik dari OpenGL dan GLUT.
- Gsub_Lite module: suksesor modul Gsub yang lebih efisien.

Gambar berikut memperlihatkan struktur hirarkis dari ARToolKit dan hubunganya dengan dependenci librarinya.



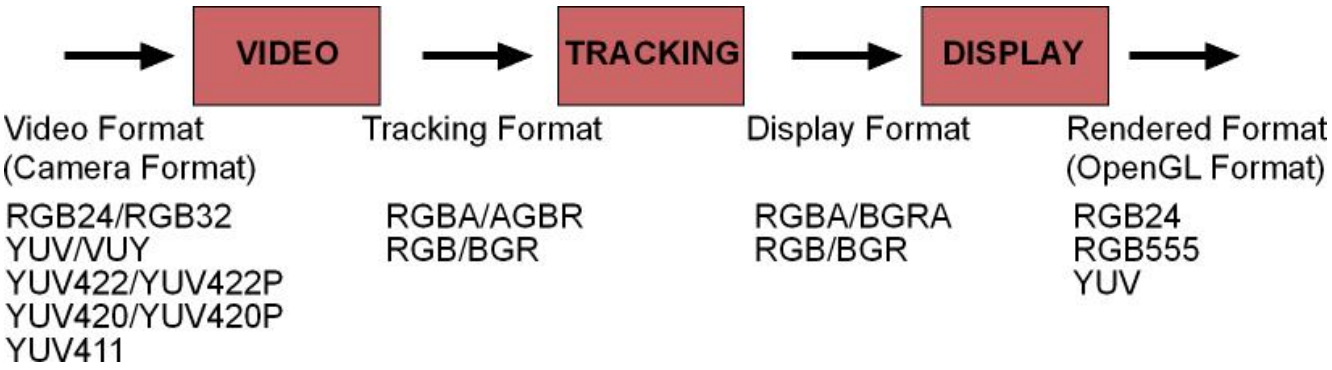

**Gambar 3.7 Struktur hirarkis ARToolKit menggunakan modul gsub**




**Gambar 3.8 Struktur hirarkis ARToolKit menggunakan modul gsub lite**

### 3.4.3 Tipe Data

ARToolKit memanipulasi beberapa variable yang berbeda. ARToolKit menggunakan format image yang berbeda untuk modul yang berbeda. Gambar dibawah mengilustrasikan beberapa format yang didukungnya. Beberapa format hanya tersedia pada beberapa platform dan hardware yang spesifik.




**Gambar 3.9 Data flow ARToolKit**.

Informasi mengenai marker yang terdeteksi berada pada struktur ARMarkerInfo structure yang didefinisikan dalam file ar.h pada direktori include.

### 3.4.4 Augmented Reality Functions

Semua fungsi ARToolKit secara umum dimulai dengan prefix ar, dan prefix spesifik untuk tiap modul (arVideo untuk video, arg untuk graphics, ar untuk main core). Anda dapat memperoleh informasi secara lebih detail pada dokumentasi API nya.

## 3.5 Menggunakan Model VRML dalam aplikasi AR

Selain dapat membuat objek 3D menggunakan fungsi-fungsi graphic OpenGL, dengan ARToolKit kita dapat juga menggunakan model VRML yang bisa dibuat menggunakan 3DMax ataupun Google Sketchup. Coba jalankan file simpleVRML.exe pada folder bin kemudian arahkan pada marker hiro dan marker kanji.
Sebagai contoh, buatlah model 3D menggunakan Google Sketchup. Kemudian simpan filenya dengan export file menjadi format .wrl. Simpan filenya dalam folder bin/Wrl.
Jika filenya sudah disimpan, buka file object_data_vrml yang terdapat pada folder bin/Data.
Kemudian buka file bud_B.dat menggunakan notepad. Edit dan arahkan ke file vrml yang baru saja anda buat. Jalankan kembali simpleVRML.exe (tidak perlu di compile).

Sumber:
http://www.hitl.washington.edu/artoolkit/documentation/index.html


Hiro