BAB 2
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Suatu aplikasi akan terlihat lebih nyata jika dikembangkan dengan tampilan yang 3D, bahkan dapat dibuat seolah-olah ada penggabungan antara dunia nyata dengan dunia virtual. Teknologi yang menerapkan hal-hal tersebut untuk menjawab kebutuhan pengguna tentang aplikasi yang terlihat nyata tersebut yaitu teknologi *Augmented Reality* (AR). Menurut Hirzer dalam laporannya yang berjudul *Marker Detection for Augmented Reality Applications* (2008), AR dapat dibuat dengan menggunakan modul dari ARToolkit ataupun ARTag. Algoritma yang dipakai adalah algoritma untuk mendeteksi garis dan mendeteksi segi empat. Algoritma-algoritma tersebut digunakan dalam AR tersebut yang kemudian berguna untuk dapat mendeteksi marker yang ada. Teknologi AR tidak sebatas hanya menampilkan model 3D saja, tetapi dapat melakukan interaksi dengan pengguna, salah satunya dengan menggunakan deteksi tangan. Menurut Billinghurst et al dalam laporannya yang berjudul *3D Natural Hand Interaction for AR Applications* (2008), untuk mendeteksi tangan pada AR adalah dengan menggunakan 4 langkah yaitu: melakukan segmentasi pada warna kulit, menentukan jarak antar titik pada tangan, mengetahui arah tangan dengan perhitungan, dan mendeteksi adanya sentuhan/tabrakan antara gambar tangan dengan objek. Implementasi dari AR juga sangatlah banyak, salah satunya adalah untuk membuat aplikasi yang berhubungan dengan musik. Menurut Poupyrev et al, dalam penelitiannya yang dirangkum dalam laporan yang berjudul *Augmented Reality Interface for Electronic Music Performance*, teknologi AR memang dapat dikembangkan untuk aplikasi musik, dan dapat digemari banyak orang jika antarmukanya dibuat semenarik mungkin.

Kesimpulannya, teknologi AR dapat diimplementasikan ke banyak hal, salah satunya ke aplikasi yang berhubungan dengan musik. Menurut penelitian tersebut juga teknologi AR untuk musik dapat memungkinkan pengguna dalam meningkatkan minatnya untuk memainkan musik.

2.2 Dasar Teori

2.2.1 Augmented Reality

Augmented Reality (AR) merupakan sebuah teknologi komputer yang seolah-olah dapat menggabungkan dunia nyata dengan dunia virtual komputer. AR ini merupakan sebuah sistem yang menciptakan tampilan hasil penggabungan dari pandangan nyata dengan obyek virtual dari komputer, termasuk dengan model tiga dimensi ke dalam tampilan. AR menggabungkan kejadian pada dunia nyata dengan kejadian pada virtual komputer pada waktu yang sama atau secara *real time*. Kedua dimensi tersebut digabungkan dengan suatu alat (biasanya *webcam*) lalu mendeteksi suatu benda yang menjadi pemicu munculnya suatu obyek dari sistem tersebut.

Tujuan utama dari AR yaitu untuk menciptakan sistem yang mana pengguna di dunia nyata serasa bersatu dengan dunia virtual. Contoh penggunaan AR dapat dilihat di gambar 2.1. Para perancang kota ingin memvisualisasikan pemandangan yang terlihat ketika sebuah jembatan dibangun. Mereka melihat pemandangan asli di gambar 2.1a, lalu dengan menggunakan AR mereka dapat melihat tampilan jika sudah dibangun jembatan yang baru (gambar 2.1b).

(a) (b)

Gambar 2.1 contoh penggunaan *Augmented Reality*

Secara umum, AR memiliki tahapan sebagai berikut:

1. Menangkap gambar menggunakan kamera (*webcam*).
2. Menemukan pemicu yang biasa disebut *marker*.
3. Mengidentifikasi pola *marker*.
4. Menampilkan obyek 3D sesuai pola yang diidentifikasi.
5. Mengulang langkah pertama sampai keempat.

Tahapan-tahapan tersebut terus menereus diulang secara *real time*, maka dapat dianggap menyatukan dunia virtual dengan dunia nyata.

2.2.2 Marker

Marker dapat dikatakan sebagai pemicu munculnya obyek tiga dimensi pada tampilan dari AR. Setiap aplikasi AR pasti menggunakan marker, dan marker yang digunakan pada setiap aplikasi berbeda. Secara umum, marker ada dua jenis, yaitu marker konvensional dan *markerless*. Marker konvensional yaitu marker dengan warna yang kontras yakni hitam dan putih. Marker ini berbentuk segiempat sama sisi dan segiempat hitam tebal membentuk *list* luar, dengan pola marker berada di dalamnya.

Gambar 2.2 beberapa contoh *marker* konvensional

Pada gambar 2.2 terlihat bahwa segiempat warna hitam tebal dan corak tertentu di dalam segiempat tersebut. Segiempat inilah yang menjadi penanda pada AR bahwa itu adalah marker, dan gambar di dalamnya dianggap sebagai pola marker. Pola inilah yang menjadi identitas tiap marker, sebagai pembeda antara marker yang satu dengan marker yang lain.

Jenis marker lain selain marker konvensional adalah *markerless*. Melalui *markerless* ini, AR dapat mengetahui marker yang disamarkan sebagai gambar tertentu. *Markerless* ini pun memiliki ciri khas tersendiri seperti perpaduan warna dan juga pola unik agar tiap marker dapat dibedakan. Contoh *markerless* dapat dilihat di gambar 2.3.

Gambar 2.3 contoh *markerless*

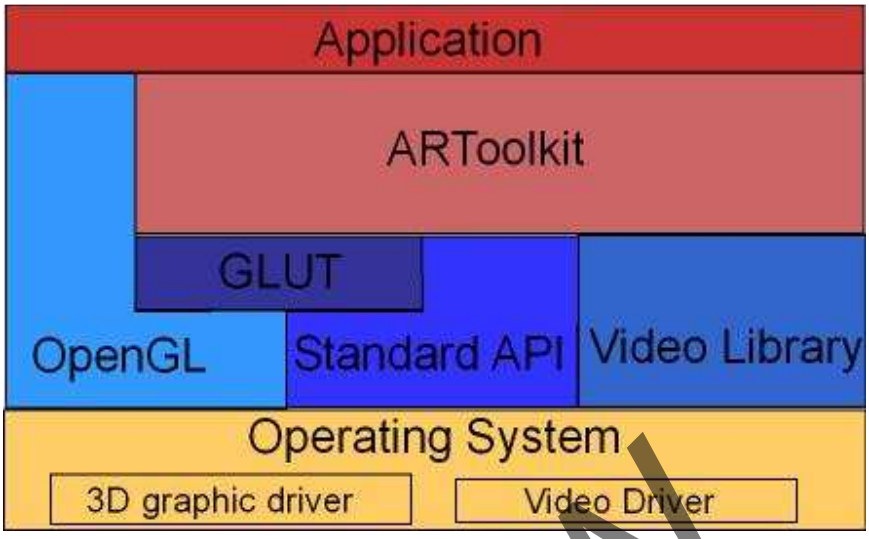
Pada gambar 2.3 terlihat sebuah marker tanpa bingkai persegi hitam dan bagian dalamnya bukan hanya warna putih. Namun dapat dilihat bahwa marker tersebut tetap memiliki pola khusus berupa tulisan ‘CORO’ sebagai identitas marker.

2.2.3 ARToolkit

ARToolkit merupakan sebuah software library untuk membangun aplikasi AR. ARToolkit pada dasarnya dikembangkan oleh Dr. Hirokazu Kato, dan pengembangannya didukung oleh Human Interface Technology Laboratory (HIT Lab) di University of Washington, dengan menggunakan bahasa C++. ARToolkit menggunakan algoritma dari *computer vision* untuk membangun aplikasi tersebut. Library dari ARToolkit dapat menghitung posisi kamera dan sudut pandangnya saat mendeteksi marker, secara *real time*. Kondisi ini memudahkan para programmer untuk mengembangkan aplikasi AR. Beberapa fitur dan kelebihan dari ARToolkit di antaranya:

- Pelacakan posisi dari sebuah kamera yang digunakan.
- Pendeteksi kode marker yang berupa persegi berwarna hitam.
- Kemampuan untuk menggunakan suatu pola marker.
- Pengujian/pencocokan kode marker yang mudah.
- Cukup cepat untuk aplikasi AR yang *real time.*
- Dapat digunakan di SGI IRIX, Linux, MacOS, dan Windows OS.
- Diberikan dengan *source code* yang lengkap.

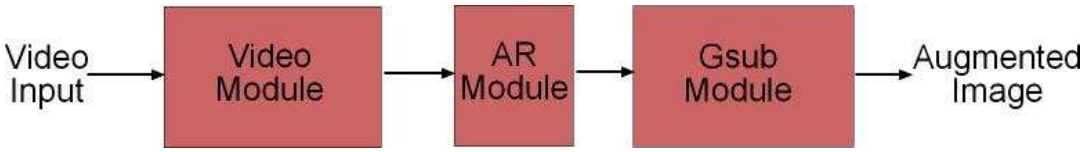
ARToolkit merupakan suatu tool yang berguna dalam melengkapi fungsi-fungsi standar yang dipakai untuk mengembangkan program berbasis AR. Tidak harus sepaket dari ARToolkit semua, dapat juga hanya menggunakan bagian-bagian tertentu darinya secara terpisah. ARToolkit dapat dipakai di berbagai *platform*, dan juga dapat meminimalisir ketergantungan penggunaan library tanpa mengorbankan efisiensi. ARToolkit menggunakan OpenGL untuk *render*, GLUT untuk penanganan OS & *hardware*, dan *Standard API* untuk tiap *platform*. Berikut ini adalah gambaran arsitektur dari ARToolkit.



Gambar 2.4 Arsitektur ARToolkit

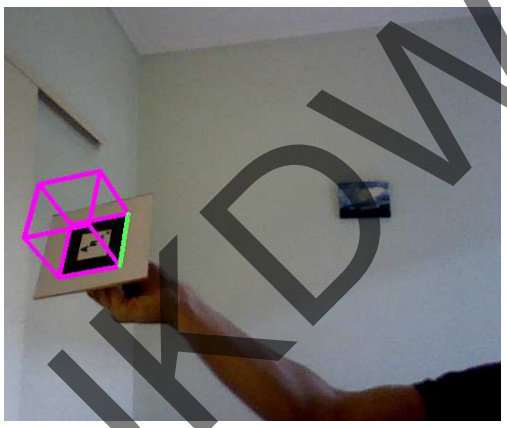
Library pada ARToolkit sendiri terdiri dari empat modul:

- *AR module*: inti modul dengan pelacakan marker, pencocokan, dan kumpulan parameter.
- *Video module*: sebuah kumpulan dari penangkapan input frame-frame video.
- *Gsub module*: kumpulan grafik berdasarkan dari library OpenGL dan GLUT.
- *Gsub_Lite module*: menempatkan kembali GSub dengan kumpulan grafik yang lebih efisien.

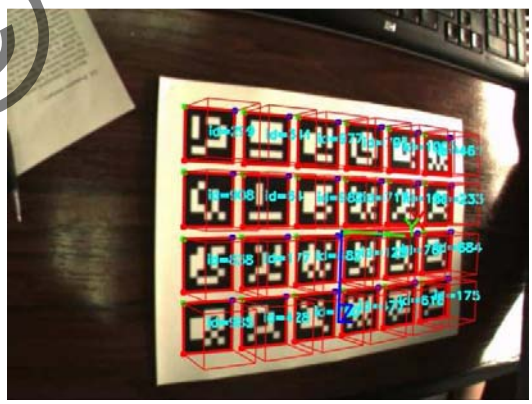


Gambar 2.5 Alur dari ARToolkit

ARToolkit juga mempunyai mendeteksi *single marker* dan *multi marker*. Sedangkan obyek yang ditampilkan oleh ARToolkit digambarkan dalam *environtment* OpenGL. ARToolkit membutuhkan cahaya yang cukup (tidak terlalu gelap dan tidak terlalu terang) agar dapat mengenali marker, karena masih terbatas pada konsep *computer vision* yaitu bagaimana komputer melihat, maksudnya apa yang dilihat oleh komputer adalah bit berupa angka, di mana warna juga dianggap angka oleh komputer. Contoh deteksi *single marker* dan *multiple marker* dapat dilihat pada gambar 2.6 dan gambar 2.7.



Gambar 2.6 Deteksi *single marker*



Gambar 2.7 Deteksi *multiple marker*

2.2.4 OpenGL

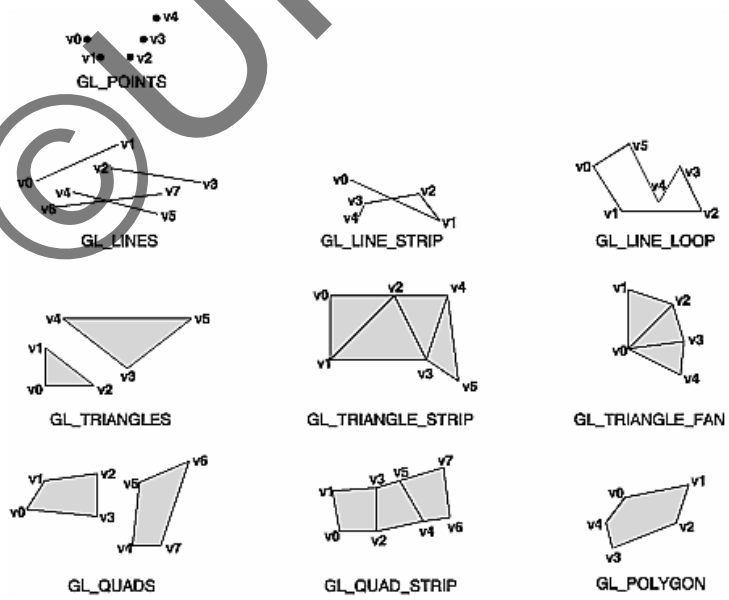
OpenGL yang merupakan singkatan dari Open Graphic Library ini, adalah suatu software antarmuka untuk pembuatan gambar secara digital. OpenGL merupakan standar API, dan berisi library yang dapat membantu membuat gambar pada suatu aplikasi baik 2D (sumbu x dan y) maupun 3D (sumbu x, y, dan z), dan penulis membutuhkannya untuk memvisualisasikan bentuk tuts piano. Model 3D dari OpenGL terbentuk dengan membuat sebuah titik (*vertex*) lalu menghubungkannya dengan *vertex-vertex* yang lain. Fungsi dari OpenGL ini ada banyak, tergantung dari bentuk yang ingin ditampilkan. Pada table 2.1 berikut dicantumkan berbagai fungsi pada OpenGL:

Tabel 2.1 Fungsi-fungsi pada OpenGL

Fungsi	Keterangan
GL_POINTS	Menggambar point pada setiap vertex
GL_LINES	Menggambar garis yang tidak tersambung. Penggambaran dilakukan pada v_0 dan v_1, v_2 dan v_3, dan seterusnya. Apabila sisa, maka verteks akan diabaikan.
GL_POLYGON	Menggambar bangun dengan menghubungkan seluruh verteks yang ada. Bangun dapat digambar dengan minimal 3 verteks.
GL_TRIANGLES	Menggambar bangun segitiga. Verteks harus kelipatan tiga. Apabila tersisa 1 atau 2 verteks, maka akan diabaikan.
GL_LINE_STRIP	Menggambar garis yang tersambung antar verteks, v_0 dihubungkan dengan v_1, v_1 dihubungkan dengan v_2, dan seterusnya. Namun verteks terakhir tidak kembali dihubungkan kembali ke v_0.

GL_LINE_LOOP	Seperti pada GL_LINE_STRIP, namun verteks terakhir disambungkan ke verteks pertama.
GL_QUADS	Menggambar segiempat, yang dapat digambar dengan minimal 4 verteks dan kelipatan empat. Apabila tersisa 1, 2, atau 3 verteks, maka akan diabaikan.
GL_QUADS_STRIP	Menggambar segiempat, namun segiempat tersebut berhubungan satu dengan yang lainnya. Apabila ada verteks tersisa, maka akan diabaikan.
GL_TRIANGLE_STRIP	Menggambar segitiga, namun segitiga tersebut berhubungan satu dengan yang lainnya. Apabila ada verteks tersisa, maka akan diabaikan.
GL_TRIANGLE_FAN	Sama dengan GL_TRIANGLE_STRIP, namun ada satu verteks yang menjadi pusat.

Fungsi-fungsi dari OpenGL tersebut merupakan dasar dari pembuatan bentuk obyek. Contoh dari setiap fungsi tersebut dapat dilihat di gambar 2.8.



Gambar 2.8 Berbagai contoh hasil dari fungsi-fungsi OpenGL

2.2.5 VRML

VRML (Virtual Reality Modeling Language) merupakan suatu bahasa pemrograman dan pemodelan obyek 3D. VRML dapat membuat gambar 3D dan konsepnya tetap menggunakan titik-titik koordinat x, y, dan z. VRML dapat digunakan untuk membantu pembuatan aplikasi di berbagai bidang, termasuk membantu penulis dalam membuat obyek 3D. Dibutuhkan VRML *browser* atau *plug-in* untuk melihat VRML. *Browser* yang paling sering dipakai adalah *Netscape's Live 3D,* yang sudah tersedia di Windows. VRML juga tidak perlu di-*compile*, seperti HTML. VRML dapat dibuat menggunakan *text editor* apapun, bahkan *notepad* sekalipun, asalkan sudah paham *syntax* dari bahasa pemrogramannya. Ada dua versi dari VRML, yaitu VRML 1.0 dan yang terbaru adalah VRML 2.0. Perbedaannya terletak pada suara, interaktivitas, dan strukturnya. Pada penelitian ini penulis menggunakan VRML 2.0.

BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Spesifikasi Perangkat Keras dan Perangkat Lunak

3.1.1 Spesifikasi Perangkat Keras

Pembuatan aplikasi yang dibuat penulis ini menggunakan spesifikasi sebagai berikut:

- Intel® Core™ i3-3217 CPU @ 1.80GHz (4 CPUs)
- RAM 4GB
- 465GB HDD
- VGA 1760 MB
- Monitor 1366 x 768
- Webcam laptop dengan resolusi 1200 x 720
- Webcam eksternal dengan resolusi 1200 x 720
- Mouse & keyboard standart

Adapun spesifikasi minimal yang diperlukan untuk membuat program ini yaitu:

- Processor Intel Celeron 2.13 GHz
- RAM 512 MB
- HDD free 20 GB
- VGA 128MB
- Webcam laptop dengan resolusi 600 x 400
- Mouse & keyboard standart

3.1.2 Spesifikasi Perangkat Lunak

Pembuatan aplikasi ini membutuhkan spesifikasi perangkat lunak sebagai berikut:

- Sistem Operasi: Windows 7 Professional 32bit
- Tools: Microsoft Visual Studio 2008, Blender Foundation: Blender 2.49b
- Berbagai library yang terdiri dari library: ARToolkit 2.7.1, OpenGL, glut 3.7.6, DSVL 0.0.8b-win32.
- OpenVRML-0.14.3-win32
- File msvcp71d.dll, msvcp71.dll, msvcr71d.dll, msvcr71.dll
- DirectX Runtime version 11

Spesifikasi minimal dari perangkat lunak untuk menjalankan aplikasi ini yaitu:

- Sistem Operasi: Windows XP Service Pack 3
- Berbagai library yang terdiri dari library: ARToolkit 2.7.1, OpenGL, glut 3.7.6, DSVL 0.0.8b-win32
- Open VRML-0.14.3-win32
- File msvcp71d.dll, msvcp71.dll, msvcr71d.dll, msvcr71.dll
- DirectX Runtime version 9

3.2 Langkah Kerja

3.2.1 Konfigurasi Sistem

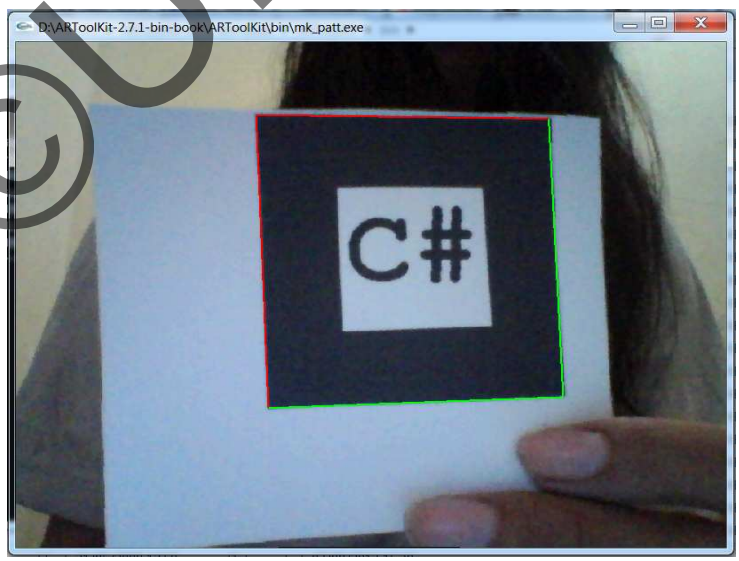
Sistem Augmented Reality dengan menggunakan ARToolkit dapat dibuat dengan melakukan beberapa hal konfigurasi, di antaranya:

1. Menempatkan *file-file* ARToolkit pada lokasi tertentu di dalam komputer.
2. Memasukkan beberapa *file* dan *library* ke lokasi-lokasi tertentu baik di dalam maupun di luar folder ARToolkit.
3. Melakukan konfigurasi pada Visual Studio.
4. Melakukan *converting* dan *building* pada ARToolkit.sln.

3.2.2 Langkah Awal Kerja

Langkah-langkah yang harus dipersiapkan dan dilakukan sebelum membuat program menggunakan ARToolkit antara lain:

1. Membuat *marker* menggunakan *tool* pengolah gambar (Adobe Photoshop, Corel Draw, Photoscape, atau *tool* lain yang sejenis). *Marker* tersebut harus berbingkai hitam, yang mana sudah terdapat pada [ARToolkit]\patterns\blankPatt.gif. *Marker* kosong tersebut lalu ditambahkan suatu pola tertentu pada bagian tengahnya.
2. Mencetak *marker* yang sudah dibuat. *Marker* akan lebih mudah dikenali jika ukurannya semakin besar dan memiliki kontras hitam-putih yang lebih tinggi.
3. Mengenalkan marker yang sudah dibuat dengan menjalankan mk_patt.exe yang ada di [ARToolkit]\bin\mk_patt.exe. Letakkan *marker* di depan kamera. Jika di layar monitor sudah terlihat sisi-sisi *marker* ditandai garis merah dan hijau seperti yang tampak pada gambar 3.1 di bawah ini, maka klik kiri pada gambar dan isikan nama *pattern*/*marker* untuk menyimpan.



Gambar 3.1 Proses pengenalan *marker*

Ulangi langkah 1-3 tersebut untuk mengenali *marker* yang lainnya. Selanjutnya tinggal lanjutkan proses pembuatan sistem berbasis AR tersebut, termasuk membuat obyek animasinya. Jika berhasil, maka ketika marker tersebut diletakkan di depan kamera, pada tampilan akan muncul obyek animasi yang sudah dibuat tersebut.

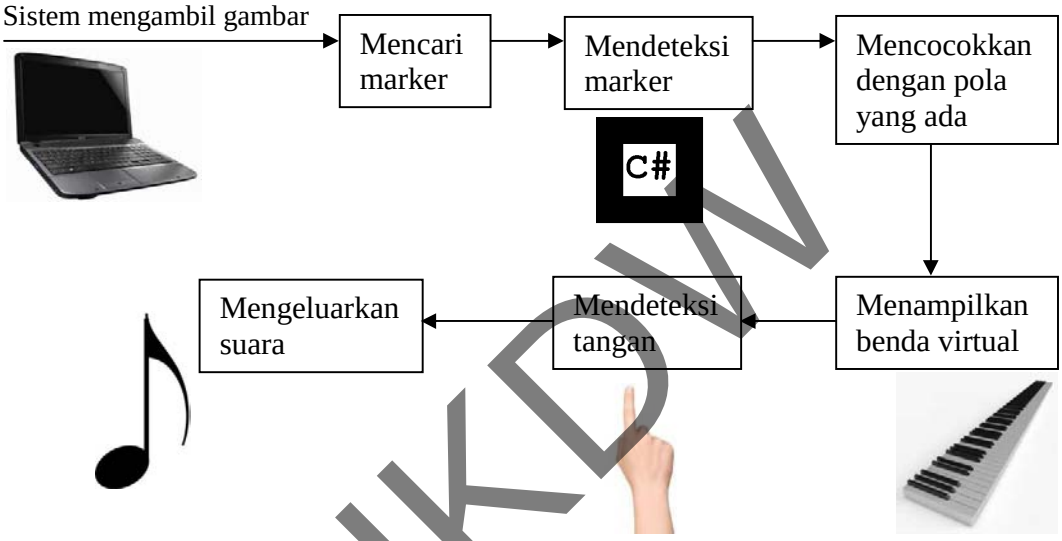
3.2.3 Pseudocode

1. Kamera (webcam) mengambil parameter-parameter video seperti jumlah fps dan resolusi layar.
2. Kamera mengambil gambar video.
3. Sistem melakukan pengecekan apakah terdapat marker atau tidak.
4. Jika ditemukan ada marker, sistem mendeteksi pola yang berada di dalam marker.
5. Jika pola marker sesuai dengan yang ada di sistem, sistem akan mencari benda virtual yang sesuai dengan pola tersebut.
6. Sistem menampilkan benda virtual sesuai dengan pola marker, beserta dengan posisi dan orientasinya.
7. Sistem akan mendeteksi adanya tangan ataupun jari tangan.
8. Jika jari tangan tersebut terdeteksi menyentuh marker, sistem menampilkan suara sesuai dengan marker yang disentuh.
9. Sistem akan mengulangi dari langkah 2, dan baru berhenti jika sudah mendapatkan perintah untuk berhenti dari pengguna.
10. Tampilan sistem ditutup dan kamera berhenti mengambil gambar.

3.3 Gambaran Sistem

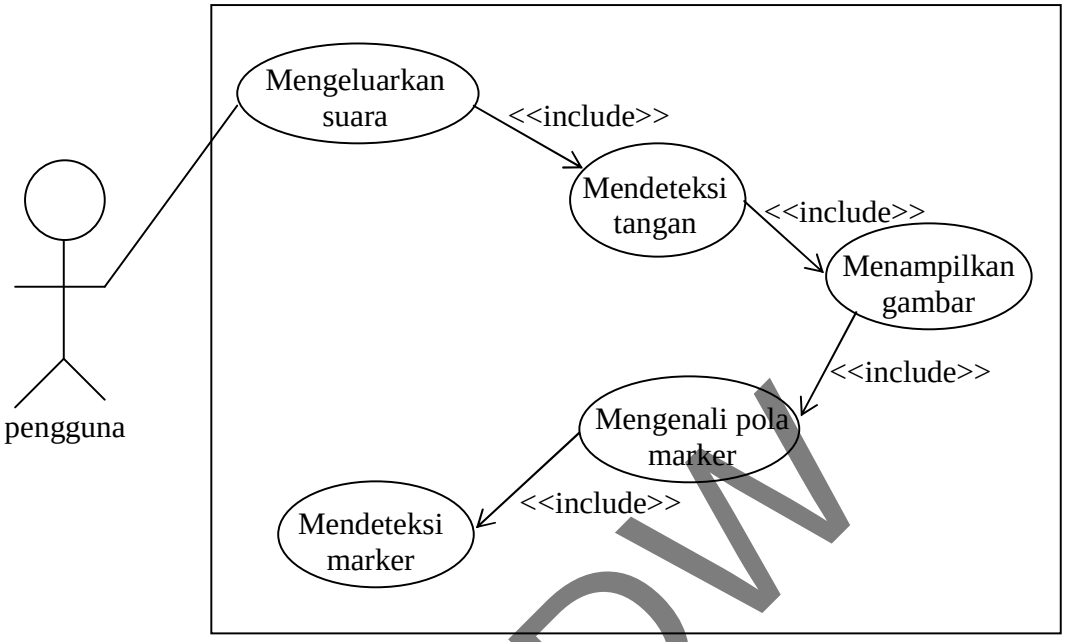
3.3.1 Arsitektur Sistem

Berikut ini adalah penjelasan tentang bagaimana ARToolkit bekerja dan dapat menggabungkan dunia virtual dengan dunia nyata.



Gambar 3.2 Arsitektur Cara Kerja Sistem

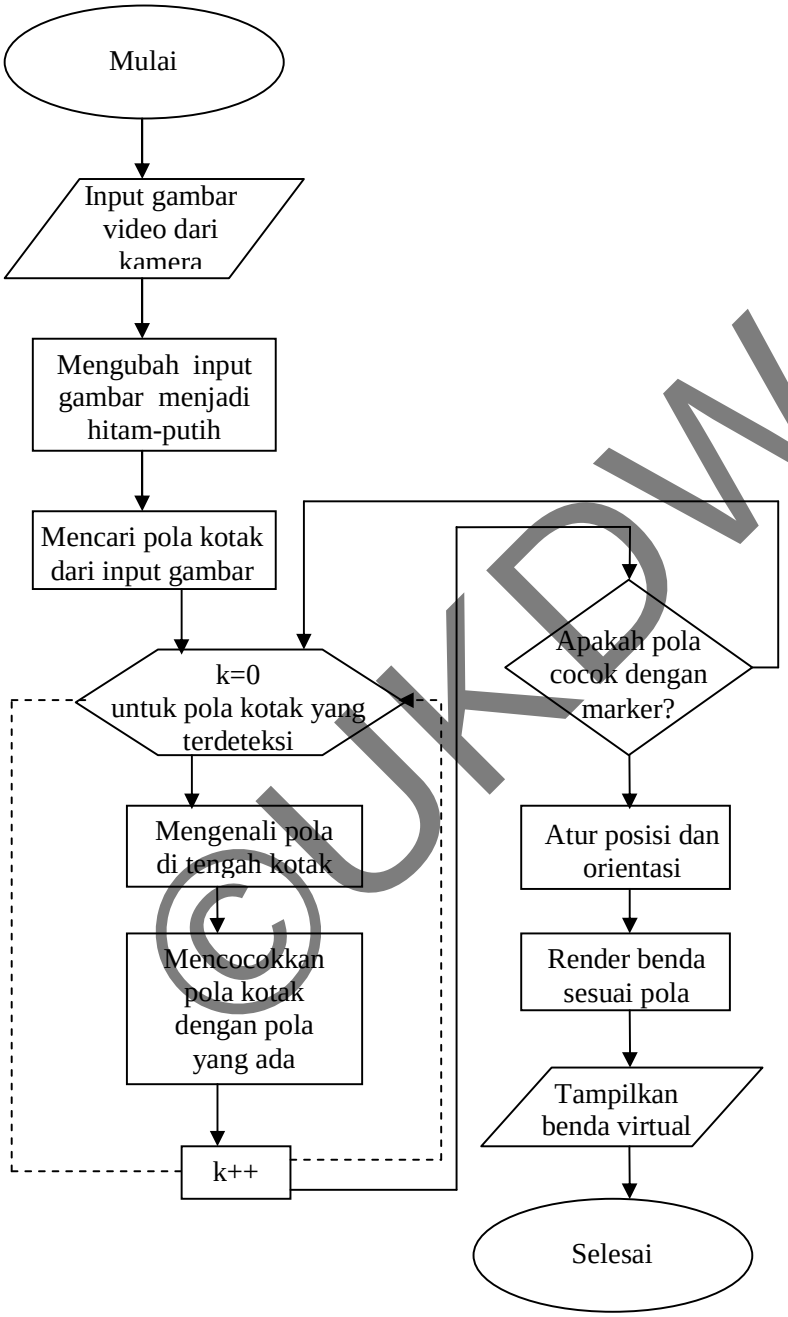
3.3.2 Diagram Use Case



Gambar 3.3 Diagram Use Case Sistem

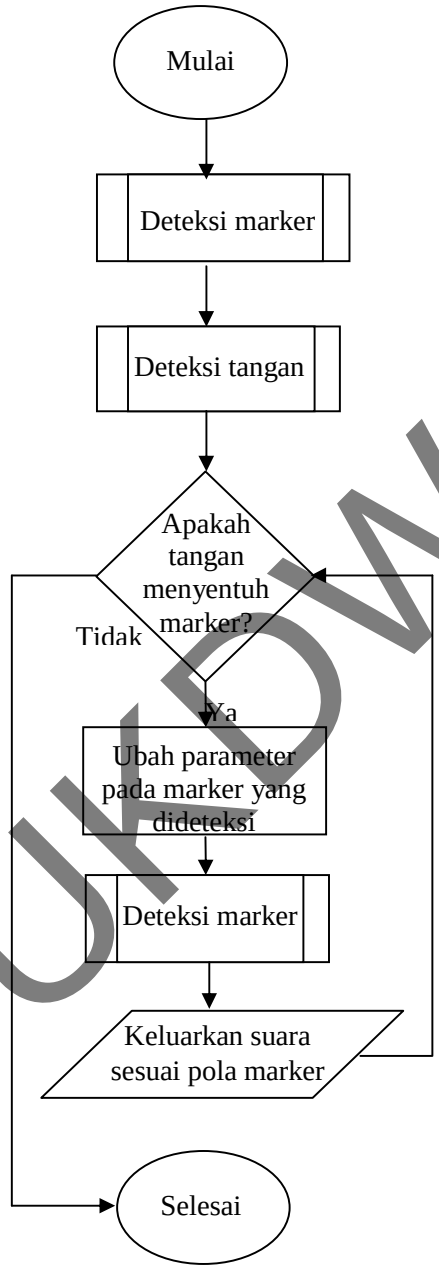
Pada diagram di atas ditunjukkan bahwa pengguna dapat mengeluarkan suara dari sistem AR yang dibangun, dengan sistem harus mendeteksi marker dan mengenali polanya terlebih dahulu, lalu menampilkan gambar virtual, dan selanjutnya dari tangan yang terdeteksi ketika menyentuh marker maka akan keluar suara. Pada kasus ini yang dibangun adalah piano virtual dan suara yang keluar adalah suara piano, dan sistem dibangun menggunakan teknologi AR.

3.3.3 Flowchart Sistem



Gambar 3.4 Flowchart dari sistem pada ARToolkit

Dari Flowchart tersebut dapat dijelaskan bahwa langkah awal yang dilakukan sistem yaitu mengakses kamera (webcam) untuk input gambar video. Lalu input gambar tersebut diubah menjadi gambar hitam-putih dengan nilai *threshold* tertentu. Kemudian sistem akan mencari pola kotak dari input gambar tersebut. Dari pola-pola kotak yang terdeteksi, sistem akan memeriksa lebih lanjut tentang pola yang ada di dalam kotak dan mencocokkan dengan data *marker* yang terdapat di dalam sistem. Jika ternyata pola yang terdeteksi sama dengan *marker* yang ada di dalam sistem, maka selanjutnya sistem akan mengatur posisi dan orientasi *marker* yang terdeteksi. Lalu sistem akan menggambar benda virtual dan akan ditampilkan sesuai letak *marker* tersebut terdeteksi.



Gambar 3.5 Flowchart gambaran kerja sistem

Flowchart yang terakhir tersebut menunjukkan gambaran kerja sistem secara keseluruhan. Setelah sistem mendeteksi adanya marker dan tangan, maka akan dicek lagi apakah tangan tersebut menyentuh marker atau tidak. Jika iya, akan ada suatu perubahan reaksi dari benda virtual yang ditampilkan. Selain itu, dengan fungsi dari ARSound, ketika tangan menyentuh marker maka akan ada suara yang muncul sesuai dengan marker yang disentuh.

3.4 Rancangan Struktur Data

3.4.1 Fungsi-fungsi

Fungsi-fungsi yang terdapat pada ARToolkit antara lain:

Tabel 3.1 Fungsi-fungsi pada ARToolkit

No.	Nama Fungsi	Kegunaan	Letak
1	arVideoOpen	Inisialisasi membuka kamera	setupCamera
2	arVideoInqSize	Mengambil ukuran video yang diambil	setupCamera
3	arParamLoad	Mengambil parameter kamera	setupCamera
4	arParamChangeSize	Mengubah ukuran kamera yang ada di file parameter	setupCamera
5	arParamDisp	Menampilkan parameter kamera	setupCamera
6	arInitCparam	Inisialisasi parameter kamera	setupCamera
7	arVideoCapStart	Mengambil gambar video	setupCamera
8	arDetectMarker	Mendeteksi keberadaan marker	Idle
9	arGetTransMat	Menghitung posisi kamera terhadap marker	Idle
10	arglDispImage	Menampilkan video	Display
11	arVideoCapNext	Memanggil pengambilan frame video selanjutnya	Display
12	arVrmlDraw	Menampilkan benda dalam bentuk	Display

		VRML	
13	arCleanup	Menghapus pengaturan	Quit
14	arVideoCapStop	Menghentikan pengambilan video	Quit
15	arVideoClose	Menutup kamera	Quit

Tabel 3.2 Nama fungsi dan kegunaannya di dalam ARToolkit

No	Nama fungsi	Kegunaan
1	setupCamera	Untuk mengambil informasi tentang kamera yang digunakan
2	setupMarkers	Untuk mengambil informasi tentang *marker* di dalam *file*
3	debugReportMode	Untuk menampilkan laporan sistem
4	Idle	Untuk mendeteksi keberadaan *marker*
5	Visibility	Untuk mengecek apakah tampilan sistem terlihat pada layar monitor
6	Reshape	Untuk melakukan perubahan ukuran tampilan sistem
7	Keyboard	Untuk melakukan suatu *event* saat suatu tombol *keyboard* ditekan
8	Display	Untuk menampilkan benda virtual 3D pada *marker*
9	Quit	Untuk mengentikan semua pengaturan saat tampilan sistem ditutup

3.4.2 File Pendukung

Sistem AR yang akan dibangun ini tidak menggunakan *database* untuk menyimpan data. Namun sistem ini memiliki file untuk menyimpan data-data, yaitu di dalam *file* yang bernama *object_vrml_data*, yang terdapat di [ARToolkit]bin\Data. Struktur isinya adalah sebagai berikut:

#the number of patterns to be recognized

2

#pattern 1
VRML Wrl/gambar1.dat
Data/patt.hiro
80.0
0.0 0.0

#sound 1
SOUNDWrl/Sound/nada1.wav
Data/patt.suara
80.0
0.0 0.0

Tabel 3.3 Penjelasan struktur isi dari object_vrml_data

No	Isi	Keterangan
1	#the number of patterns to be recognized 2	Mendeskripsikan jumlah marker yang akan dideteksi. Pada kasus ini contohnya 2.
2	#pattern 1	Definisi untuk pola marker pertama
3	VRML Wrl/gambar1.dat	Berfungsi untuk menyambungkan ke file .dat, pada kasus ini ke gambar1.dat
4	Data/patt.hiro	Berfungsi menyambungkan ke marker yang dituju, pada kasus ini ke marker patt.hiro
5	80.0	Lebar fisik marker yang digunakan, pada kasus ini lebarnya 80 milimeter.
6	0.0 0.0	Koordinat titik pusat marker, pada kasus ini di 0.0,0.0 yaitu di tengah marker.
7	#sound 2	Definisi untuk pola marker kedua untuk jenis ARSound
8	SOUNDWrl/Sound/nada1.wav	Berfungsi untuk menyambungkan ke file yang bertipe suara dengan ekstensi .wav, pada kasus ini ke nada1.wav

Sistem berbasis AR yang akan dibuat penulis ini dapat menampilkan benda virtual VRML yang berekstensi .wrl. *File* ini terletak pada [ARToolkit]\bin\Wrl\. Penulis membuat obyek benda virtual tersebut menggunakan aplikasi Blender. Obyek VRML tersebut memiliki konfigurasi pada *file*-nya yang berekstensi .dat, dengan struktur isinya sebagai berikut:

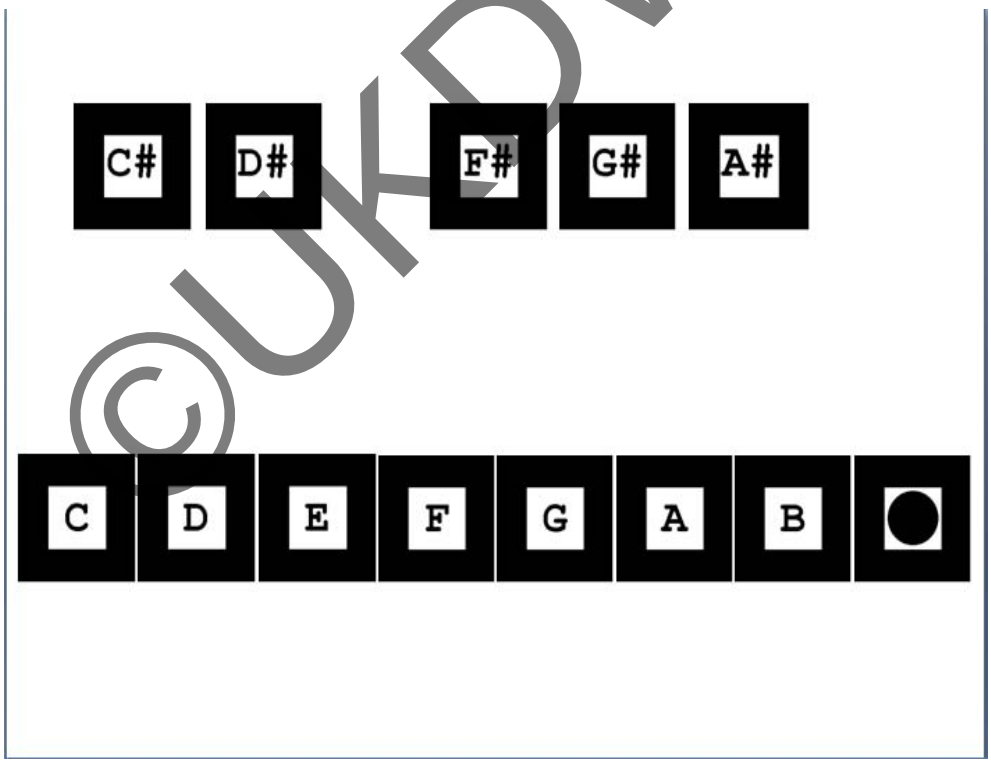
gambar1.wrl
0.0 0.0 0.0 # Translation
0.0 0.0 0.0 # Rotation
10.0 10.0 10.0 #Scale

Tabel 3.4 Penjelasan isi file .dat

No	Isi	Keterangan
1	gambar1.wrl	Penanda obyek VRML yang berhubungan dengan file .dat
2	0.0 0.0 0.0 #translation	Pergeseran obyek VRML berdasarkan sumbu x, y, dan z
3	0.0 0.0 0.0 #rotation	Perputaran obyek VRML terhadap sumbu x, y, dan z, dengan menggunakan nilai derajat
4	10.0 10.0 10.0 #scale	Ukuran skala perbesaran obyek VRML berdasarkan sumbu x, y, dan z, dengan nilai 10.0 berarti obyek berukuran 1000%

3.5 Rancangan Antar Muka

Sistem ini nantinya akan membaca marker-marker yang sudah disiapkan penulis, dan marker-marker tersebut ditempatkan pada selembar kertas dengan gambar tuts piano sepanjang 1 oktaf. Ukuran tuts piano juga sudah disesuaikan dengan ukuran aslinya. Gambar yang akan ditampilkan oleh sistem AR adalah gambar model 3D tuts piano, dengan tiap marker untuk setiap tuts piano. Nada yang dihasilkan oleh tiap tuts juga sudah disesuaikan dengan aslinya, seperti nada C di tuts C, nada D di tuts D, dan seterusnya, serta lengkap dengan nada kromatisnya seperti C#, D#, dan lain-lain. Suara tiap nada tersebut akan keluar jika pengguna menyentuhkan jarinya ke marker. Gambaran rancangan antarmuka sistem dapat dilihat di gambar 3.7 di bawah ini:



Gambar 3.6 Rancangan antarmuka sistem

3.6 Rancangan Evaluasi Sistem

Pada penelitian ini penulis akan mencoba meneliti tingkat keberhasilan sistem dalam mendeteksi marker secara benar, termasuk bunyi output apakah sudah sesuai atau belum. Faktor yang akan menjadi bahan penelitian adalah sumber cahaya yang digunakan dalam menjalankan sistem, dan variabel yang akan diubah adalah jarak antara sumber cahaya dengan obyek marker. Penulis menetapkan variabel jarak yang akan dipakai adalah 1 meter, 1,5 meter, 2 meter, 2,5 meter, dan 3 meter. Selain itu penulis juga akan mencoba penelitiannya di luar ruangan, untuk menggunakan matahari sebagai sumber cahayanya, dan dengan dua kondisi cuaca, yaitu saat cuaca cerah dan saat cuaca mendung. Selain dari sumber cahaya, penulis juga akan meneliti jarak antara obyek dengan kamera, dan variabel jarak yang ditetapkan penulis adalah 10 cm, 20 cm, 30 cm, 40 cm, dan 50 cm.

Penelitian ini menggunakan cahaya sebagai salah satu faktor penelitian, maka dari itu penulis akan menggunakan tempat yang sama untuk meneliti agar hasilnya lebih valid, karena tiap tempat bisa mendapatkan cahaya yang masuk dengan jumlah yang berbeda-beda. Penulis juga akan menggunakan tempat di ruang tertutup yang sinar matahari hampir tidak dapat masuk secara langsung, agar penelitian dapat dilakukan kapanpun entah malam ataupun siang hari. Khusus untuk penelitian yang menggunakan cahaya matahari, penulis melakukannya di luar ruangan pada siang hari. Selain lingkungan yang sama, penulis juga menggunakan sumber cahaya yang sama pula, yakni lampu dengan cahaya putih, dengan daya 20 watt, dan dengan arah sudut datang tepat berada di atas obyek marker dengan jarak tertentu.

BAB 4
IMPLEMENTASI DAN ANALISIS SISTEM

4.1 Implementasi Sistem

4.1.1 Konfigurasi ARToolkit

Ada beberapa hal yang perlu dilakukan sebagai persiapan untuk memulai pembuatan sistem menggunakan ARToolkit dan Microsoft Visual Studio 2008, untuk sistem operasi Windows 7 32 bit:

- ARToolkit versi 2.72.1 untuk Windows
  - Sumber: http://sourceforge.net/projects/artoolkit/files/artoolkit.2.72.1/ARToolKit-2.72.1-bin-win32.zip/download
- DSVideoLib versi 0.0.8b
  - Sumber: http://sourceforge.net/projects/artoolkit/files/artoolkit/2.72.1/DSVL-0.0.8b.zip/download
- OpenVRML versi 0.14.3
  - Sumber: http://sourceforge.net/projects/artoolkit/files/artoolkit/2.72.1/OpenVRML-0.14.3-win32.zip/download
- GLUT versi 3.7.6
  - Sumber: http://user.xmission.com/~nate/glut/glut-3.7.6-bin.zip
  - File ini berfungsi sebagai librari dari OpenGL

- File msvcp71d.dll
  - Sumber: http://www.dll-files.com/dllindex/dll-files.shtml?msvcp71d

- File msvcp71.dll
  - Sumber: http://www.dll-files.com/dllindex/dll-files.shtml?msvcp71
  - File ini berfungsi sebagai librari, yang mana fungsinya sendiri adalah untuk mendukung sistem menjadi real-time

- File msvcr71d.dll
  - Sumber: http://www.dll-files.com/dllindex/dll-files.shtml?msvcr71d
  - File ini berfungsi sebagai librari, yang mana untuk mendukung bagian grafik tampilan.

- File msvcr71.dll
  - Sumber: http://www.dll-files.com/dllindex/dll-files.shtml?msvcr71
  - File ini berfungsi sebagai modul standar bahasa C yang digunakan pada sistem ini.

4.1.2 Instalasi ARToolkit

Langkah-langkah untuk melakukan instalasi ARToolit adalah sebagai berikut:

1. Extract ARToolkit-2.72.1-bin-win32.zip (misalnya ke C:\ )
2. Extract DSVL-0.08b, maka muncul folder DSVL. Masukkan folder tersebut ke dalam folder C:\ARToolkit\ dan *replace all*.
3. Buka folder C:\ARToolkit\DSVL\bin\

4. Copy file DSVL.dll dan DSVLd.dll, lalu *paste* ke dalam C:\ARToolkit\bin\ dan *replace all* juga.
5. Extract OpenVRML-0.14.3-win32.zip ke C:\ARToolkit\
6. Copy js32.dll dari C:\ARToolkit\OpenVRML\bin\ ke C:\ARToolkit\bin
7. Copy file mscvp71d.dll, mscvp71.dll, mscvr71d.dll, mscvr71.dll ke dalam C:\windows\system32\ dan C:\ARToolkit\bin\
8. Extract glut-3.7.6-bin.zip
9. Copy glut32.dll ke C:\windows\system32
10. Copy glut.h ke C:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\include\GL
11. Copy glut32.lib ke C:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\include\GL
12. Buka file C:\ARToolkit\ARToolkit.sln dengan Microsoft Visual Studio 2008, lalu lakukan konversi.
13. Buka Configuration Manager, lalu centang pada *checkbox Build* untuk libARvrml dan simpleVRML.
14. Jalankan *Build Solution*.

4.1.3 Pembuatan Marker

Sistem AR tentunya membutuhkan marker dalam fungsionalitasnya, dan dalam ARToolkit ini marker yang digunakan adalah marker jenis konvensional, yang mana marker tersebut berbentuk kotak dengan warna hitam putih, dan dengan pola unik di tengahnya. Marker cukup mudah dibuat dengan aplikasi pengolah gambar, dan telah ARToolkit telah menyediakan pula *template* marker kosong pada C:\ARToolkit\patterns\blankPatt.gif, sehingga pembuatan marker menjadi semakin mudah.

4.1.4 Pengenalan Pola Marker

Setelah membuat marker, selanjutnya adalah mengenalkan pola ke sistem. Langkah-langkah yang harus dilakukan adalah sebagai berikut:

1. Cetak pola marker yang telah dibuat tersebut.
2. Jalankan C:\ARToolkit\bin\mk_patt.exe lalu masukkan nama file parameter kamera.
3. Letakkan marker di depan kamera, hingga pada tampilan monitor gambar marker tersebut muncul garis merah di sisi kiri dan atas marker, serta garis hijau di sisi kanan dan bawah marker.
4. Klik kiri pada tampilan sistem tersebut lalu beri nama marker dengan format: patt<nama marker>

4.1.5 Pembuatan Obyek 3D dan Obyek Suara

Obyek yang ditampilkan pada sistem ini adalah obyek dengan bentuk 3D. Penulis menggunakan tool Blender versi 2.49 untuk melakukan pemodelan obyeknya, yang mana obyek tersebut lalu di-*eksport* menjadi format VRML97. Setelah itu, copy file obyek tersebut ke dalam C:\ARToolkit\bin\Wrl\, dan copy juga salah satu file .dat yang terdapat di dalam folder tersebut lalu buka dengan menggunakan Wordpad. Sesuaikan namanya menjadi sama dengan obyek 3D yang dibuat, dan jika sudah selesai lalu ganti nama file dari .dat tersebut menjadi sama dengan nama file obyek 3D tadi dengan tanpa mengubah format file.

Adapun obyek 3D yang dibuat penulis dan akan ditampilkan memang cukup sederhana, yaitu berbentuk tuts piano. Adapun panjang tuts piano adalah 1 oktaf, dan obyek 3D yang ditampilkan ada dua jenis, yaitu tuts warna putih untuk nada biasa dan tuts hitam untuk nada kres.

Selain obyek berbentuk gambar 3D, penulis juga membuat obyek suara yang mana dapat dimunculkan menggunakan sistem ini juga. File-file suara tersebut menggunakan format .wav, dan diletakkan di

C:\ARToolkit\bin\Wrl\nada. Terdapat 13 nada, yang terdiri dari satu oktaf nada (dari C sampai B) ditambah 1 nada C tinggi.

4.1.6 Implementasi Tampilan Sistem

Tampilan pada sistem ini adalah diambil dari apa yang dilihat oleh webcam. Apa yang dideteksi oleh sistem, hasilnya ditampilkan pada tampilan utama ini sesuai dengan apa yang sudah dibuat. Misalnya sistem mendeteksi adanya marker dengan simbol F#, lalu jika ada obyek lain menutupi marker tersebut, maka sistem akan mengeluarkan output berupa suara piano dengan nada F#.

Gambar 4.1 Tampilan Sistem

4.2 Analisis Sistem

Pada penelitian ini, penulis melakukannya di dalam sebuah ruangan tertutup dan tidak terganggu cahaya dari luar ruangan, yang mana di dalam ruangan tersebut terdapat lampu dengan daya 20 watt yang terletak pada bagian atas ruangan. Adapun tinggi ruangan adalah 3,2 meter. Media yang digunakan penulis adalah webcam eksternal.

Namun ditemui kendala bahwa sistem tidak dapat membunyikan nada ketika dalam satu tampilan ada lebih dari 1 marker yang terdeteksi, harus yang lain tidak terdeteksi terlebih dahulu barulah sistem dapat mengeluarkan suara dengan nada yang sesuai dengan marker yang disentuh. Maka dari itu, penulis menggunakan alat bantu berupa barang-barang yang dirakit sendiri oleh penulis, yaitu yang berupa rangkaian potongan busa hati dan karton. Alat bantu tersebut akan menutupi marker-marker yang ada, dan ketika ditekan maka akan terangkat yang mana menyebabkan marker dapat terlihat oleh kamera, lalu alat bantu tersebut akan segera turun kembali menyebabkan marker tertutupi kembali, sehingga sistem jadi mengeluarkan suara sesuai marker yang ditutupi tersebut.

4.2.1 Pengukuran Nilai

Nilai yang diukur pada penelitian ini adalah tingkat akurasi atau keberhasilan sistem dalam mendeteksi marker sesuai dengan apa yang telah dibuat. Faktor-faktor yang mempengaruhi penelitian ini adalah jarak antara marker dengan kamera, dan juga cahayanya. Apabila ketika marker disentuh sistem dapat mengeluarkan nada yang sesuai, maka hal itu dianggap berhasil.

4.2.2 Hasil Pengukuran

Pada penelitian ini, sistem akan diuji akurasinya pada 13 marker, yakni marker dengan nada C, C#, D, D#, E, F, F#, G, G#, A, A#, B, dan C tinggi, dengan melakukan 5 kali percobaan pada setiap variabel di setiap markernya. Adapun hasil pengujian itu jumlah yang berhasil adalah sebagai berikut:

Tabel 4.1 Pengujian pada marker C

	10 cm	20 cm	30 cm	40 cm	50 cm	Akurasi
1 m	4	5	2	0	1	48 %
1,5 m	4	4	3	1	1	52 %
2 m	3	5	1	1	2	48 %
2,5 m	3	4	1	0	0	32 %
3 m	2	2	0	1	0	20 %
Akurasi	64 %	80 %	28 %	12 %	16 %	

Tabel 4.2 Pengujian pada marker C#

	10 cm	20 cm	30 cm	40 cm	50 cm	Akurasi
1 m	5	5	5	4	2	84 %
1,5 m	4	5	4	2	2	68 %
2 m	5	4	2	1	1	56 %
2,5 m	3	4	1	2	0	40 %
3 m	3	4	2	0	0	36 %
Akurasi	80 %	88 %	56 %	36 %	20 %	

Tabel 4.3 Pengujian pada marker D

	10 cm	20 cm	30 cm	40 cm	50 cm	Akurasi
1 m	5	5	3	4	1	72 %
1,5 m	4	4	2	2	1	52 %
2 m	4	5	2	0	0	44 %
2,5 m	3	3	0	1	0	28 %
3 m	3	3	2	1	0	36 %
Akurasi	76 %	80 %	40 %	32 %	8 %	

Tabel 4.4 Pengujian pada marker D#

	10 cm	20 cm	30 cm	40 cm	50 cm	Akurasi
1 m	4	5	4	3	2	72 %
1,5 m	4	4	3	2	2	60 %
2 m	3	5	2	1	1	48 %
2,5 m	4	4	1	1	0	40 %
3 m	3	3	1	0	0	28 %
Akurasi	72 %	84 %	44 %	28 %	20 %	

Tabel 4.5 Pengujian pada marker E

	10 cm	20 cm	30 cm	40 cm	50 cm	Akurasi
1 m	3	5	2	3	1	56 %
1,5 m	3	4	3	1	2	52 %
2 m	3	3	3	1	1	44 %
2,5 m	2	4	1	0	1	32 %
3 m	2	3	2	0	0	28 %
Akurasi	52 %	76 %	44 %	20 %	20 %	

Tabel 4.6 Pengujian pada marker F

	10 cm	20 cm	30 cm	40 cm	50 cm	Akurasi
1 m	1	2	0	0	0	12 %
1,5 m	0	1	1	0	0	8 %
2 m	1	1	0	0	0	8 %
2,5 m	0	1	0	0	0	4 %
3 m	0	0	0	0	0	0 %
Akurasi	8 %	20 %	4 %	0 %	0 %	

Tabel 4.7 Pengujian pada marker F#

	10 cm	20 cm	30 cm	40 cm	50 cm	Akurasi
1 m	4	4	4	3	3	72 %
1,5 m	4	5	3	3	2	68 %
2 m	3	5	2	2	1	52 %
2,5 m	2	4	2	1	0	36 %
3 m	2	2	2	0	0	24 %
Akurasi	60 %	80 %	52 %	36 %	24 %	

Tabel 4.8 Pengujian pada marker G

	10 cm	20 cm	30 cm	40 cm	50 cm	Akurasi
1 m	2	2	1	0	0	20 %
1,5 m	1	2	0	0	0	12 %
2 m	1	1	1	0	0	12 %
2,5 m	1	0	1	0	0	8 %
3 m	0	1	0	0	0	4 %
Akurasi	20 %	24 %	12 %	0 %	0 %	

Tabel 4.9 Pengujian pada marker G#

	10 cm	20 cm	30 cm	40 cm	50 cm	Akurasi
1 m	3	5	4	2	2	64 %
1,5 m	4	3	1	2	1	44 %
2 m	3	4	2	0	1	40 %
2,5 m	3	3	1	1	0	32 %
3 m	2	3	2	0	0	28 %
Akurasi	60 %	72 %	40 %	20 %	16 %	

Tabel 4.10 Pengujian pada marker A

	10 cm	20 cm	30 cm	40 cm	50 cm	Akurasi
1 m	2	4	2	1	1	40 %
1,5 m	3	3	1	0	1	32 %
2 m	2	2	1	0	0	20 %
2,5 m	1	1	1	1	0	16 %
3 m	1	1	1	0	0	12 %
Akurasi	36 %	44 %	24 %	8 %	8 %	

Tabel 4.11 Pengujian pada marker A#

	10 cm	20 cm	30 cm	40 cm	50 cm	Akurasi
1 m	4	4	3	3	1	60 %
1,5 m	3	4	2	2	2	52 %
2 m	4	3	2	1	1	44 %
2,5 m	2	5	2	1	0	40 %
3 m	2	3	1	1	0	28 %
Akurasi	60 %	68 %	40 %	32 %	16 %	

Tabel 4.12 Pengujian pada marker B

	10 cm	20 cm	30 cm	40 cm	50 cm	Akurasi
1 m	3	5	2	1	2	52 %
1,5 m	4	4	2	2	1	52 %
2 m	3	4	3	1	0	44 %
2,5 m	3	3	2	1	1	40 %
3 m	3	3	1	1	0	32%
Akurasi	64 %	76 %	40 %	24 %	16 %	

Tabel 4.13 Pengujian pada marker C tinggi

	10 cm	20 cm	30 cm	40 cm	50 cm	Akurasi
1 m	3	4	2	0	1	40 %
1,5 m	2	3	1	1	1	32 %
2 m	2	3	1	0	1	28 %
2,5 m	1	3	2	2	0	32 %
3 m	2	2	0	1	0	20 %
Akurasi	40 %	60 %	24 %	16 %	12 %	

4.2.3 Analisa Hasil Pengukuran terhadap Jarak Obyek dengan Kamera

Berdasarkan penelitian yang dilakukan penulis yaitu dengan mencoba akurasi sistem dengan mengubah-ubah jarak antara obyek dengan kameranya, dapat disimpulkan bahwa jaraknya memang sangat berpengaruh. Penulis menggunakan variabel jarak 10 cm, 20 cm, 30 cm, 40 cm, dan 50 cm antara obyek dengan kamera.

Pada jarak 10 cm, marker mudah untuk dideteksi karena memang jarak tersebut termasuk dekat. Namun pada jarak 20 cm ternyata sistem lebih mudah untuk mengenali marker, yang mana itu dapat terlihat dari jumlah keberhasilannya dalam mengenali yang lebih banyak daripada di jarak 10 cm.

Pada jarak 30 cm, sistem tingkat keberhasilan sistem dalam mengenali mulai agak berkurang lagi, karena sudah cukup jauh dan sistem lebih sulit untuk dapat mengenalinya. Begitu juga untuk jarak-jarak selanjutnya yaitu 40 cm dan 50 cm, sistem semakin sulit untuk dapat mengenalinya, yang mana itu membuat tingkat keberhasilannya dalam mengenali marker sangat berkurang.

Maka dari pengujian tersebut dapat disimpulkan bahwa belum tentu semakin dekat jarak antara obyek dengan kamera akan semakin mempermudah sistem dalam mengenali marker. Berdasarkan pengujian tersebut, jarak yang ideal antara obyek dengan kamera adalah sekitar 20 cm.

4.2.4 Analisa Hasil Pengukuran terhadap Obyek dengan Cahaya

Selain menguji dari jarak antara obyek dengan kamera, penulis juga menguji menggunakan variabel lain yaitu faktor cahaya. Pada kasus ini, penulis meneliti jarak antara obyek dengan sumber cahaya (lampu ruangan), yang mana menggunakan jarak 1 meter, 1,5 meter, 2 meter, 2,5 meter, dan 3 meter. Selain menguji berdasarkan jaraknya, penulis juga mencoba untuk menguji sumber cahaya lain yaitu cahaya matahari, yang mana dalam kasus ini yang diuji bukan jaraknya tetapi cuacanya yaitu saat sedang cerah dan sedang mendung.

Pada pengujian tersebut terlihat bahwa pada jarak 1 meter tingkat keberhasilan masih relatif tinggi, itu karena jaraknya masih dekat dengan sumber cahaya (lampu) sehingga cahaya yang didapat juga cukup terang yaitu dengan tingkat intensitas cahaya 3,7 EV, yang mana hal itu menyebabkan sistem menjadi lebih mudah dalam mengenali marker. Pada jarak 1,5 meter yang tingkat intensitas cahayanya 3,2 EV, tingkat keberhasilan mulai menurun, karena cahaya yang diterima tidak seterang jarak 1 meter. Begitu juga pada jarak 2 meter, 2,5 meter, dan 3 meter, tingkat keberhasilan cenderung semakin menurun. Hal itu terjadi karena semakin jauh dari sumber cahaya, maka cahaya yang didapat juga semakin sedikit atau kurang terang, sehingga semakin mempersulit sistem dalam mengenali marker.

Selanjutnya pada pengujian yang dilakukan di luar ruangan, penulis mencoba pada saat cuaca cerah, tepatnya dengan intensitas cahaya 13,3 EV. Ternyata sistem malah sangat sulit atau bahkan tidak dapat mengenali marker, karena cahaya yang didapat terlalu banyak/terang, sehingga menjadi mengganggu sistem dalam mengenali marker. Lalu pada saat cuaca mendung yaitu dengan intensitas cahaya 9,6 EV, sistem justru dapat mengenali marker dengan mudah, karena cahayanya cukup terang, tapi tidak terlalu terang seperti saat cuaca cerah.

Dari pengujian tersebut dapat disimpulkan bahwa cahaya juga sangat berpengaruh dalam tingkat keberhasilan sistem untuk mengenali marker. Semakin terang cahaya akan semakin mempermudah sistem dalam mengenali marker, tetapi jika terlalu terang juga malah akan mengganggu sehingga justru akan menurunkan tingkat keberhasilannya.

4.2.5 Kelebihan Sistem

Sistem ini tidak hanya berfungsi untuk menampilkan gambar, tapi juga output yang berupa suara, sehingga lebih lengkap daripada sistem-sistem Augemented Reality biasanya.

4.2.6 Kekurangan Sistem

Sistem ini belum dapat mengeluarkan beberapa output suara secara bersamaan, hanya 1 output saja dalam 1 waktu.

LISTING PROGRAM

- Form Main Menu Utama

LAMPIRAN

```
#the number of patterns to be recognized
15

#pattern 0
VEML Wrl/kosong.dat
Data/patt.obyek
80.0
0.0 0.0

#pattern 1
VRML  Wrl/kotak.dat
Data/patt.do
80.0
0.0 0.0

#sound 1
SOUND Wrl/nada/C.wav
Data/patt.c
80.0
0.0 0.0

#sound 2
SOUND Wrl/nada/Cc.wav
Data/patt.cc
80.0
0.0 0.0

#sound 3
SOUND Wrl/nada/D.wav
Data/patt.d
80.0
0.0 0.0

#sound 4
SOUND Wrl/nada/Dd.wav
Data/patt.dd
80.0
0.0 0.0

#sound 5
SOUND Wrl/nada/E.wav
Data/patt.e
80.0
```

```
0.0 0.0

#sound 6
SOUND Wrl/nada/F.wav
Data/patt.f
80.0
0.0 0.0

#sound 7
SOUND Wrl/nada/Ff.wav
Data/patt.ff
80.0
0.0 0.0

#sound 8
SOUND Wrl/nada/G.wav
Data/patt.g
80.0
0.0 0.0

#sound 9
SOUND Wrl/nada/Gg.wav
Data/patt.gg
80.0
0.0 0.0

#sound 10
SOUND Wrl/nada/A.wav
Data/patt.a
80.0
0.0 0.0

#sound 11
SOUND Wrl/nada/Aa.wav
Data/patt.aa
80.0
0.0 0.0

#sound 12
SOUND Wrl/nada/B.wav
Data/patt.b
80.0
0.0 0.0

#sound 13
SOUND Wrl/nada/C1.wav
Data/patt.c1
80.0
0.0 0.0
```

LISTING PROGRAM

- Form Main Menu Utama

LAMPIRAN

```
#ifdef _WIN32
#  include <windows.h>
#endif
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#ifdef __APPLE__
#  include <GLUT/glut.h>
#else
#  include <GL/glut.h>
#endif

#include <AR/config.h>
#include <AR/video.h>
#include <AR/param.h>                   // arParamDisp()
#include <AR/ar.h>
#include <AR/gsub_lite.h>
#include <AR/arvrml.h>

#include "object.h"

// ============================================================
//  Constants
// ============================================================
#define VIEW_SCALEFACTOR            0.025       // 1.0 ARToolKit
unit becomes 0.025 of my OpenGL units.
#define VIEW_SCALEFACTOR_1          1.0               // 1.0
ARToolKit unit becomes 1.0 of my OpenGL units.
#define VIEW_SCALEFACTOR_4          4.0               // 1.0
ARToolKit unit becomes 4.0 of my OpenGL units.
#define VIEW_DISTANCE_MIN           4.0               // Objects
closer to the camera than this will not be displayed.
#define VIEW_DISTANCE_MAX           4000.0            // Objects
further away from the camera than this will not be displayed.


// ============================================================
//  Global variables
// ============================================================

static int prefWindowed = TRUE;
static int prefWidth = 640;
static int prefHeight = 480;
static int prefDepth = 32;
static int prefRefresh = 0;
```

```
static ARUint8          *gARTImage = NULL;

static int              gARTThreshhold = 100;
static long             gCallCountMarkerDetect = 0;

static int              gPatt_found = FALSE;

static ARParam          gARTCparam;
static ARGL_CONTEXT_SETTINGS_REF gArglSettings = NULL;

static ObjectData_T     *gObjectData;
static int              gObjectDataCount;

// ============================================================
//    Functions
// ============================================================

static int setupCamera(const char *cparam_name, char *vconf,
ARParam *cparam)
{
    ARParam             wparam;
      int                     xsize, ysize;

    if (arVideoOpen(vconf) < 0) {
            fprintf(stderr, "setupCamera(): Unable to open
connection to camera.\n");
      return (FALSE);
      }

    if (arVideoInqSize(&xsize, &ysize) < 0) return (FALSE);
      fprintf(stderr, "SOUND\n");
    fprintf(stdout, "Camera image size (x,y) = (%d,%d)\n", xsize,
ysize);

    if (arParamLoad(cparam_name, 1, &wparam) < 0) {
            fprintf(stderr, "setupCamera(): Error loading
parameter file %s for camera.\n", cparam_name);
        return (FALSE);
    }
    arParamChangeSize(&wparam, xsize, ysize, cparam);
    fprintf(stdout, "*** Camera Parameter ***\n");
    arParamDisp(cparam);

    arInitCparam(cparam);

      if (arVideoCapStart() != 0) {
      fprintf(stderr, "setupCamera(): Unable to begin camera data
capture.\n");
            return (FALSE);
      }

      return (TRUE);
}
```

```
static int setupMarkersObjects(char *objectDataFilename)
{
    if ((gObjectData = read_VRMLdata(objectDataFilename,
&gObjectDataCount)) == NULL) {
        fprintf(stderr, "setupMarkersObjects(): read_VRMLdata
returned error !!\n");
        return (FALSE);
      }

    printf("Object count = %d\n", gObjectDataCount);

      return (TRUE);
}

static void debugReportMode(void)
{
      if(arFittingMode == AR_FITTING_TO_INPUT ) {
            fprintf(stderr, "FittingMode (Z): INPUT IMAGE\n");
      } else {
            fprintf(stderr, "FittingMode (Z): COMPENSATED
IMAGE\n");
      }

      if( arImageProcMode == AR_IMAGE_PROC_IN_FULL ) {
            fprintf(stderr, "ProcMode (X)   : FULL IMAGE\n");
      } else {
            fprintf(stderr, "ProcMode (X)   : HALF IMAGE\n");
      }

      if (arglDrawModeGet(gArglSettings) ==
AR_DRAW_BY_GL_DRAW_PIXELS) {
            fprintf(stderr, "DrawMode (C)   : GL_DRAW_PIXELS\n");
      } else if (arglTexmapModeGet(gArglSettings) ==
AR_DRAW_TEXTURE_FULL_IMAGE) {
            fprintf(stderr, "DrawMode (C)   : TEXTURE MAPPING
(FULL RESOLUTION)\n");
      } else {
            fprintf(stderr, "DrawMode (C)   : TEXTURE MAPPING
(HALF RESOLUTION)\n");
      }

      if( arTemplateMatchingMode == AR_TEMPLATE_MATCHING_COLOR ) {
            fprintf(stderr, "TemplateMatchingMode (M)   : Color
Template\n");
      } else {
            fprintf(stderr, "TemplateMatchingMode (M)   : BW
Template\n");
      }

      if( arMatchingPCAMode == AR_MATCHING_WITHOUT_PCA ) {
            fprintf(stderr, "MatchingPCAMode (P)   : Without
PCA\n");
      } else {
            fprintf(stderr, "MatchingPCAMode (P)   : With PCA\n");
      }
```

```
}

static void Quit(void)
{
	arglCleanup(gArglSettings);
	arVideoCapStop();
	arVideoClose();
#ifdef _WIN32
	CoUninitialize();
#endif
	exit(0);
}

static void Keyboard(unsigned char key, int x, int y)
{
	int mode;
	switch (key) {
		case 0x1B:
		case 'Q':
		case 'q':
			Quit();
			break;
		case 'C':
		case 'c':
			mode = arglDrawModeGet(gArglSettings);
			if (mode == AR_DRAW_BY_GL_DRAW_PIXELS) {
				arglDrawModeSet(gArglSettings,
AR_DRAW_BY_TEXTURE_MAPPING);
				arglTexmapModeSet(gArglSettings,
AR_DRAW_TEXTURE_FULL_IMAGE);
			} else {
				mode = arglTexmapModeGet(gArglSettings);
				if (mode == AR_DRAW_TEXTURE_FULL_IMAGE)
	arglTexmapModeSet(gArglSettings,
AR_DRAW_TEXTURE_HALF_IMAGE);
				else arglDrawModeSet(gArglSettings,
AR_DRAW_BY_GL_DRAW_PIXELS);
			}
			fprintf(stderr, "*** Camera - %f (frame/sec)\n",
(double)gCallCountMarkerDetect/arUtilTimer());
			gCallCountMarkerDetect = 0;
			arUtilTimerReset();
			debugReportMode();
			break;
		case '?':
		case '/':
			printf("Keys:\n");
			printf(" q or [esc]    Quit demo.\n");
			printf(" c             Change arglDrawMode and
arglTexmapMode.\n");
			printf(" ? or /        Show this help.\n");
			printf("\nAdditionally, the ARVideo library
supplied the following help text:\n");
			arVideoDispOption();
			break;
		default:
```

```
                    break;
        }
}

static void Idle(void)
{
        static int ms_prev;
        int ms;
        float s_elapsed;
        ARUint8 *image;

        ARMarkerInfo    *marker_info;
        ARMarkerInfo    *marker_info2;
      int             marker_num;
      int             i, j, k;


              ms = glutGet(GLUT_ELAPSED_TIME);
        s_elapsed = (float)(ms - ms_prev) * 0.001;
        if (s_elapsed < 0.01f) return;
        ms_prev = ms;

        arVrmlTimerUpdate();

        if ((image = arVideoGetImage()) != NULL) {
              gARTImage = image;
              gPatt_found = FALSE;

              gCallCountMarkerDetect++;
              if (arDetectMarker(gARTImage, gARTThreshhold,
&marker_info, &marker_num) < 0) {
                    exit(-1);
              }
                          // Detect the markers in the video frame.
              if (arDetectMarker(gARTImage, gARTThreshhold,
&marker_info2, &marker_num) < 0) {
                    exit(-1);
              }

              for (i = 0; i < gObjectDataCount; i++) {
                    k = -1;
                    for (j = 0; j < marker_num; j++) {
                          if (marker_info[j].id ==
gObjectData[i].id) {
                                if( k == -1 ) k = j;
              else if (marker_info[k].cf < marker_info[j].cf) {
                                      k = j;
                                }
                          }
                    }

                    if (k != -1) {

                          if (gObjectData[i].visible == 0) {
                                fprintf(stderr,
                                      gObjectData[i].name );
```

```
                              arGetTransMat(&marker_info[k],
gObjectData[i].marker_center, gObjectData[i].marker_width,
gObjectData[i].trans);
                        } else {
      fprintf(stderr,
                                    gObjectData[i].name);
                              arGetTransMatCont(&marker_info[k],
gObjectData[i].trans,
gObjectData[i].marker_center, gObjectData[i].marker_width,
gObjectData[i].trans);
                        }
                        if (i>=2){
      PlaySound(gObjectData[i].name,NULL,SND_ASYNC);
                              fprintf(stderr,
                                    gObjectData[i].name);
                        }
                        else{
                              marker_info[i].status=1;
                              gObjectData[i].visible=1;
                        }
                        gPatt_found = TRUE;
                  } else {
                        marker_info[i].status = 0;
                        gObjectData[i].visible = 0;
                  }
            // Tell GLUT to update the display.
            glutPostRedisplay();
      }
}
}

static void Visibility(int visible)
{
      if (visible == GLUT_VISIBLE) {
            glutIdleFunc(Idle);
      } else {
            glutIdleFunc(NULL);
      }
}
```

```
static void Reshape(int w, int h)
{
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      glViewport(0, 0, (GLsizei) w, (GLsizei) h);

      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();

      }

static void Display(void)
{
      int i;
    GLdouble p[16];
      GLdouble m[16];

      glDrawBuffer(GL_BACK);
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      arglDispImage(gARTImage, &gARTCparam, 1.0, gArglSettings);
      arVideoCapNext();
      gARTImage = NULL;

      if (gPatt_found) {
                        arglCameraFrustumRH(&gARTCparam,
VIEW_DISTANCE_MIN, VIEW_DISTANCE_MAX, p);
            glMatrixMode(GL_PROJECTION);
            glLoadMatrixd(p);
            glMatrixMode(GL_MODELVIEW);

                        glLoadIdentity();



            for (i = 0; i < gObjectDataCount; i++) {
                  if ((gObjectData[i].visible != 0) &&
(gObjectData[i].vrml_id >= 0))
                        arglCameraViewRH(gObjectData[i].trans, m,
VIEW_SCALEFACTOR_4);
                        glLoadMatrixd(m);

                        arVrmlDraw(gObjectData[i].vrml_id);
                  }
            }
      }

      glutSwapBuffers();
}

int main(int argc, char** argv)
{

      int i;
      char glutGamemode[32];
```

```
	const char *cparam_name = "Data/camera_para.dat";
#ifdef _WIN32
	char			*vconf = "Data\\WDM_camera_flipV.xml";
#else
	char			*vconf = "";
#endif
	char objectDataFilename[] = "Data/DataObjectAndSound";

	// -----------------------------------------------------------
-------------------
	// Library inits.
	//

	glutInit(&argc, argv);

	// -----------------------------------------------------------
-------------------
	// Hardware setup.
	//

	if (!setupCamera(cparam_name, vconf, &gARTCparam)) {
		fprintf(stderr, "main(): Unable to set up AR
camera.\n");
		exit(-1);
	}

#ifdef _WIN32
	CoInitialize(NULL);
#endif

	// -----------------------------------------------------------
-------------------
	// Library setup.
	//

		glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA |
GLUT_DEPTH);
	if (!prefWindowed) {
		if (prefRefresh) sprintf(glutGamemode, "%ix%i:%i@%i",
prefWidth, prefHeight, prefDepth, prefRefresh);
		else sprintf(glutGamemode, "%ix%i:%i", prefWidth,
prefHeight, prefDepth);
		glutGameModeString(glutGamemode);
		glutEnterGameMode();
	} else {
		glutInitWindowSize(gARTCparam.xsize,
gARTCparam.ysize);
		glutCreateWindow(argv[0]);
	}


	if ((gArglSettings = arglSetupForCurrentContext()) == NULL)
{
		fprintf(stderr, "main(): arglSetupForCurrentContext()
returned error.\n");
		exit(-1);
```

```
      }
      debugReportMode();
      arUtilTimerReset();

      if (!setupMarkersObjects(objectDataFilename)) {
            fprintf(stderr, "main(): Unable to set up AR objects
and markers.\n");
            Quit();
      }

    fprintf(stdout, "Pre-rendering the VRML objects...");
      fflush(stdout);
    glEnable(GL_TEXTURE_2D);
    for (i = 0; i < gObjectDataCount; i++) {
            arVrmlDraw(gObjectData[i].vrml_id);
    }
    glDisable(GL_TEXTURE_2D);
      fprintf(stdout, " done\n");

      glutDisplayFunc(Display);
      glutReshapeFunc(Reshape);
      glutVisibilityFunc(Visibility);
      glutKeyboardFunc(Keyboard);

      glutMainLoop();

      return (0);
}
```