Bahan Seminar Tugas Akhir

Perancangan Aplikasi *Viewer* Model 3D Interaktif Berbasis *Web* dengan Teknologi *Augmented Reality*

oleh :

Rizky Zulkarnaen

050402056

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS SUMATERA UTARA

MEDAN

2010

Perancangan Aplikasi *Viewer* Model 3D Interaktif Berbasis *Web* dengan Teknologi *Augmented Reality*

oleh :
Rizky Zulkarnaen
050402056

Tugas Akhir ini diajukan untuk melengkapi salah satu syarat untuk memperoleh gelar Sarjana Teknik

Disetujui oleh :
Pembimbing

Ori Novanda, MT

NIP:1972.1114.2003.121001

Diketahui oleh :
Pelaksana Harian Ketua Departemen Teknik Elektro

Prof. Dr. Ir. Usman S. Baafai

NIP:1946.1022.1973.02001

Abstrak

Dewasa ini, representasi model tiga dimensi (3D) virtual pada komputer sangat populer. Pemodelan model 3D virtual tersebut memerlukan interaksi yang baik dengan pengguna untuk mendapatkan sudut pandang yang jelas dari objek tersebut. Beberapa *software* pemodelan 3D menggunakan *pointer* seperti *keyboard* ataupun *mouse* yang dirasakan masih kurang dalam memberikan derajat kebebasan sudut pandang dari model 3D tersebut. Aplikasi ini dapat menampilkan model 3D virtual dan dapat diakses dengan mudah dari mana saja menggunakan *web browser* tanpa perlu instalasi aplikasinya. Dengan menerapkan teknologi *Augmented Reality*, model 3D dapat disajikan dengan lebih interaktif. Objek di dunia nyata dapat digunakan sebagai *tracker* untuk keperluan interaksi model 3D, sehingga memudahkan dalam menampilkan model 3D tersebut. Hasil pengujian menunjukkan aplikasi dapat menampilkan model 3D yang sederhana dengan baik dalam nilai FPS (*Framerate per Second*) yang mencukupi. Untuk model 3D yang sangat kompleks, aplikasi tidak dapat menampilkannya dengan baik karena keterbatasan aplikasinya yang berbasis *web*.

Kata Kunci : ***augmented reality*, *computer vision*, *3D model viewer*, *Rich Internet Applications***

KATA PENGANTAR

Puji dan syukur kepada Allah SWT penulis ucapkan, karena hanya dengan kehendakNya penulis dapat menyelesaikan Tugas Akhir ini, yang berjudul **"Perancangan Aplikasi *Viewer* Model 3D Interaktif Berbasis *Web* dengan Teknologi *Augmented Reality*"**. Tak lupa pula shalawat dan salam kepada Junjungan kita Nabi Besar Muhammad SAW yang telah membimbing kita kepada Islam. Di samping itu, penulis mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Ayahanda Zulkifli. Y dan Ibunda Nurmawati yang telah membesarkan dan memberikan kasih sayang yang tulus kepada penulis.
2. Kakak dan adik-adik penulis.
3. Bapak Ori Novanda ST, MT, selaku dosen pembimbing penulis yang telah sangat banyak membantu dalam penulisan Tugas Akhir ini.
4. Bapak Ir. Riswan Dinzi selaku dosen wali penulis, atas bimbingannya selama penulis kuliah di Departemen Teknik Elektro, Fakultas Teknik, Universitas Sumatera Utara.
5. Bapak Prof. Dr. Ir. Usman S. Baafai dan Bapak Rahmad Fauzi, ST, MT, selaku Pelaksana Harian Ketua dan Sekretaris Departemen Teknik Elektro, Fakultas Teknik, Universitas Sumatera Utara.
6. Seluruh staf pengajar dan pegawai Departemen Teknik Elektro Fakultas Teknik Universitas Sumatera Utara.

7. Asisten Laboratorium Sistem Pengaturan dan Komputer, Salman Alfarisi dan Fachrurozi Nasution, yang menemani dan banyak membantu penulis dalam pengerjaan Tugas Akhir ini.

8. Teman-teman stambuk 2005, dan teman-teman junior serta senior yang selama ini menjadi teman diskusi di kampus.

Berbagai usaha telah penulis lakukan demi terselesaikannya Tugas Akhir ini dengan baik, tetapi penulis menyadari akan kekurangan dan keterbatasan penulis. Oleh karena itu, saran dan kritik dengan tujuan menyempurnakan dan mengembangkan kajian dalam bidang ini sangat penulis harapkan.

Akhir kata penulis berharap agar Tugas Akhir ini dapat bermanfaat bagi pembaca dan penulis.

Medan, Desember 2010
Penulis,

Rizky Zulkarnaen
NIM:050402056

Daftar Isi

Daftar Tabel

Daftar Gambar

Bab 1

PENDAHULUAN

1.1 Latar Belakang

Pemodelan objek tiga dimensi (3D) sangat diperlukan dalam berbagai aplikasi, baik untuk simulasi maupun untuk pengenalan model dari objek nyata yang sulit disajikan secara fisik dikarenakan keterbatasan ruang dan waktu. Model suatu objek nyata dapat disajikan secara virtual yang dapat dilihat melalui suatu layar atau *display* dengan bantuan komputer sehingga pemodelan suatu objek mudah dilakukan dengan biaya yang murah. Banyak bidang yang memerlukan pemodelan objek virtual 3D ini, misalnya pemodelan organ tubuh yang bermanfaat dalam dunia kedokteran, pemodelan bangunan, pemodelan suatu produk yang akan dijual, dan lain sebagainya.

Pemodelan objek virtual tersebut memerlukan interaksi yang baik dengan pengguna untuk mendapatkan sudut pandang yang jelas dari objek tersebut. Beberapa *software* pemodelan 3D menggunakan *pointer* seperti *keyboard* ataupun *mouse* yang dirasakan masih kurang dalam memberikan derajat kebebasan sudut pandang dari model 3D tersebut. *Augmented Reality* merupakan suatu konsep perpaduan antara *"virtual reality"* dengan *"world reality"*. Sehingga objek-objek virtual 2 Dimensi (2D) atau 3 Dimensi (3D) seolah-olah terlihat nyata dan menyatu dengan dunia nyata. Pada teknologi *Augmented Reality*, pengguna dapat melihat dunia nyata yang ada di sekelilingnya dengan penambahan obyek virtual yang dihasilkan oleh komputer. Dengan teknologi

Augmented Reality, penyajian model 3D dapat disajikan dengan lebih interaktif. Objek di dunia nyata dapat digunakan sebagai *tracker* untuk keperluan interaksi model 3D virtual sehingga memudahkan dalam menampilkan objek virtual tersebut.

Aplikasi komputer (*software*) yang dapat menampilkan model 3D *viewer model 3D* biasanya memerlukan proses instalasi agar dapat dijalankan, dan terkadang hanya dapat berjalan di platform tertentu saja. Dengan berkembangnya teknologi *web*, banyak aplikasi komputer yang dibuat berbasis *web*. Aplikasi yang berbasis *web* sangat mudah diakses, bahkan dapat berjalan di berbagai platform. Dengan *web plug-in* tertentu, aplikasi yang tergolong rumit pun bisa dibuat dan dapat diakses hanya lewat *web browser*. Kebutuhan akan aplikasi yang berbasis *web* ini akan terus meningkat, karena pengguna menyukai aplikasi yang mudah dijalankan dan tanpa perlu instalasi. Begitu pula dengan aplikasi untuk *viewer* model 3D, aplikasi *viewer* 3D berbasis *web* akan memudahkan seseorang yang ingin melihat model 3D secara interaktif melalui *web browser* tanpa perlu instalasi aplikasi atau mungkin bagi perusahaan yang ingin menampilkan produknya dalam bentuk model 3D secara interaktif kepada konsumen.

Berdasarkan uraian tersebut, penulis mengembangkan suatu aplikasi untuk menampilkan objek 3D virtual dan dapat diakses dengan mudah dari mana saja menggunakan *web browser* tanpa perlu instalasi aplikasinya. Pembahasan pengembangan aplikasi ini dibuat menjadi tugas akhir yang diberi judul **"Perancangan Aplikasi *Viewer* Model 3D Interaktif Berbasis *Web* dengan Teknologi *Augmented Reality*"**.

1.2 Tujuan Penulisan

Tujuan dari penulisan tugas akhir ini adalah untuk merancang suatu aplikasi yang memudahkan penyajian objek 3D (*viewer* model 3D) terutama interaksinya dengan menerapkan teknologi *Augmented Reality*. Aplikasi yang akan dibuat ini berbasis *web* dengan konsep *Rich Internet Application (RIAs)*, sehingga aplikasi tersebut dapat menampilkan objek 3D virtual melalui *web browser* dan tentunya dapat diakses di berbagai platform tanpa perlu instalasi

aplikasi di sistem pengguna.

1.3 Rumusan Masalah

Dari latar belakang masalah, maka dapat dirumuskan beberapa permasalahan, yaitu sebagai berikut:

1. Menerapkan teknologi *augmented reality* untuk membuat *viewer* model 3D yang interaktif.
2. Merancang aplikasi berbasis *web* untuk *viewer* objek 3D dengan konsep *Rich Internet Application* (RIAs) dan teknologi *Augmented Reality*.

1.4 Batasan Masalah

Untuk menghindari pembahasan yang terlalu luas, maka penulis akan membatasi tugas akhir ini dengan hal-hal sebagai berikut:

1. Membahas secara ringkas konsep *augmented reality* dan teori yang menunjang perancangan aplikasi viewer 3D dengan *augmented reality*.
2. Tidak membahas pembuatan objek 3D nya, tetapi hanya membahas bagaimana menampilkan objek 3D tersebut dengan bantuan *engine/library* 3D dan teknologi augmented reality.
3. Pembuatan aplikasinya berbasis *web* yang dapat diakses dari *web browser*. Aplikasi tersebut nantinya dapat menampilkan objek 3D dan bisa berinteraksi dengan pengguna menggunakan sebuah *tracker*.

1.5 Metodologi Penulisan

Penulisan dilakukan dalam beberapa tahapan:

1. Tahap penelitian platform dan *framework*
   Pada tahap ini penulis meneliti platform yang akan digunakan, serta juga *framework*/*library* yang cocok untuk pengembangan aplikasi tersebut.

2. Tahap perancangan aplikasi
   Pada tahap ini dirancang aplikasinya dengan menggunakan *framework* yang telah ditentukan di tahap penelitian.

3. Tahap pengembangan aplikasi
   Pada tahap ini penulis membangun aplikasinya sesuai dengan rancangan.

4. Tahap implementasi dan pengujian
   Pada tahap ini penulis mengimplementasikan aplikasi dan melakukan pengujian dari aplikasi yang telah dikembangkan.

1.6 Sistematika Penulisan

Penulisan Tugas Akhir ini disajikan dengan sistematika penulisan sebagai berikut:

BAB I : PENDAHULUAN
Bab ini merupakan gambaran menyeluruh tentang apa yang diuraikan dalam Tugas Akhir ini, yaitu pembahasan tentang latar belakang, tujuan penulisan, batasan masalah, metode penulisan, dan sistematika penulisan.

BAB II : DASAR TEORI
Bab ini membahas tentang *augmented reality*, teori-teori yang mendukung *augmented reality*, teknik *display augmented reality*, konsep aplikasi berbasis *web* dan juga library untuk *augmented reality*.

BAB III : PERANCANGAN DAN IMPLEMENTASI

Bab ini merupakan bab yang membahas perancangan aplikasinya, dimulai dari gambaran umum aplikasi desain aplikasi dan terakhir implementasi aplikasi.

BAB IV : PENGUJIAN DAN ANALISA

Bab ini membahas tentang pengujian dan analisa dari aplikasi yang dibangun.

BAB V : KESIMPULAN DAN SARAN

Bab ini berisi tentang kesimpulan dan saran dari aplikasi yang dirancang.

Bab 2

DASAR TEORI

2.1 *Augmented Reality*

Augmented reality (AR) adalah sebuah istilah untuk lingkungan yang menggabungkan dunia nyata dan dunia virtual yang dibuat oleh komputer sehingga batas antara keduanya menjadi sangat tipis. Ronald Azuma pada tahun 1997 [1] mendefinisikan *Augmented Reality* sebagai sistem yang memiliki karakteristik sebagai berikut:

- Menggabungkan lingkungan nyata dan virtual
- Berjalan secara interaktif dalam waktu nyata
- Integrasi dalam tiga dimensi (3D)

Secara sederhana AR bisa didefinisikan sebagai lingkungan nyata yang ditambahkan objek virtual. Penggabungan objek nyata dan virtual dimungkinkan dengan teknologi *display* yang sesuai, interaktivitas dimungkinkan melalui perangkat-perangkat input tertentu.

AR merupakan variasi dari *Virtual Environments* (VE), atau yang lebih dikenal dengan istilah *Virtual Reality* (VR). Teknologi VE membuat pengguna tergabung dalam sebuah lingkungan virtual secara keseluruhan. Ketika tergabung dalam lingkungan tersebut, pengguna tidak bisa melihat lingkungan

nyata di sekitarnya. Sebaliknya, AR memungkinkan pengguna untuk melihat lingkungan nyata, dengan objek virtual yang ditambahkan atau tergabung dengan lingkungan nyata. Tidak seperti VR yang sepenuhnya menggantikan lingkungan nyata, AR sekedar menambahkan atau melengkapi lingkungan nyata [1].

Tujuan utama dari AR adalah untuk menciptakan lingkungan baru dengan menggabungkan interaktivitas lingkungan nyata dan virtual sehingga pengguna merasa bahwa lingkungan yang diciptakan adalah nyata. Dengan kata lain, pengguna merasa tidak ada perbedaan yang dirasakan antara AR dengan apa yang mereka lihat/rasakan di lingkungan nyata. Dengan bantuan teknologi AR (seperti visi komputasi dan pengenalan objek) lingkungan nyata disekitar kita akan dapat berinteraksi dalam bentuk digital (virtual). Informasi tentang objek dan lingkungan disekitar kita dapat ditambahkan kedalam sistem AR yang kemudian informasi tersebut ditampilkan diatas *layer* dunia nyata secara *real-time* seolah-olah informasi tersebut adalah nyata. Informasi yang ditampilkan oleh objek virtual membantu pengguna melaksanakan kegiatan-kegiatan dalam dunia nyata. AR banyak digunakan dalam bidang-bidang seperti kesehatan, militer, industri manufaktur dan juga telah diaplikasikan dalam perangkat-perangkat yang digunakan orang banyak, seperti pada telepon genggam.

2.2 *Mixed Reality*

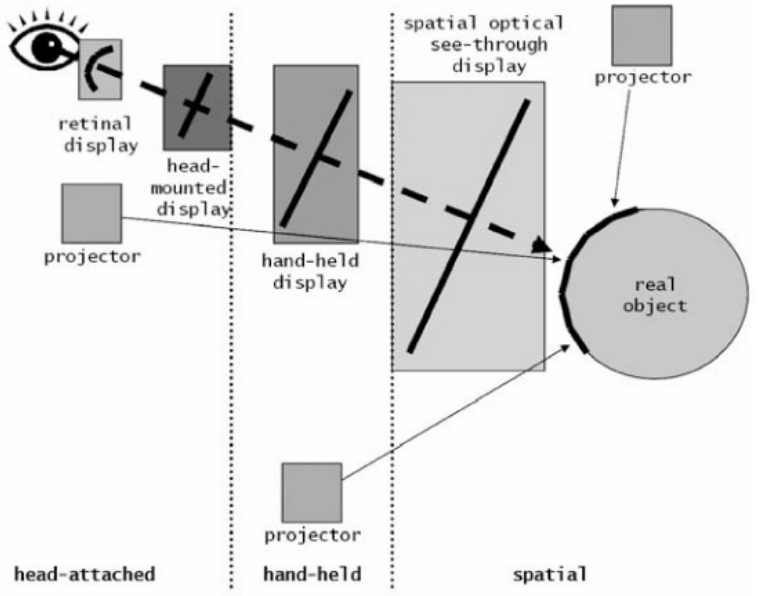
Paul Milgram and Fumio Kishino merumuskan kerangka kemungkinan penggabungan dan peleburan dunia nyata dan dunia maya yang disebut ***Milgram's Reality-Virtuality Continuum*** pada tahun 1994. Dalam Gambar 2.1, sisi yang paling kiri adalah lingkungan nyata yang hanya berisi benda nyata, dan sisi paling kanan adalah lingkungan maya yang berisi benda maya [2].

Dalam *augmented reality*, yang lebih dekat ke sisi kiri, lingkungan bersifat nyata dan benda bersifat maya, sementara dalam *augmented virtuality*, yang lebih dekat ke sisi kanan, lingkungan bersifat maya dan benda bersifat nyata.

Gambar 2.1: Mixed Reality

2.3 Teknik Display *Augmented Reality*

Sistem *display* AR merupakan sistem manipulasi citra yang menggunakan seperangkat optik, elektronik, dan komponen mekanik untuk membentuk citra dalam jalur optik antara mata pengamat dan objek fisik yang akan digabungkan dengan teknik AR. Bergantung kepada optik yang digunakan, citra bisa dibentuk pada sebuah benda datar atau suatu bentuk permukaan yang kompleks (tidak datar)[3]. Gambar 2.2 mengilustrasikan kemungkinan citra akan dibentuk untuk mendukung AR, peletakan display bergantung dari pandangan pengguna dan objek, dan tipe citra seperti apa yang akan dihasilkan (*planar* atau *curved*).



Gambar 2.2: Pembentukan citra untuk display *augmented reality*

Secara garis besarnya ada tiga teknik display AR [3], yaitu sebagai berikut:

1. *Head-Attached Display*
2. *Handheld Display*
3. *Spatial Display*

2.3.1 *Head-Attached Display*

Head-Attached Display merupakan teknik display yang mengharuskan penggunanya untuk memakai sistem ini di kepala pengguna. Berdasarkan teknik citra yang terbentuk, *Head-Attached Display* terbagi tiga, yaitu sebagai berikut:

- *Head-Mounted Display.*
- *Head-Mounted Projectors.*
- *Virtual Retina Display.*

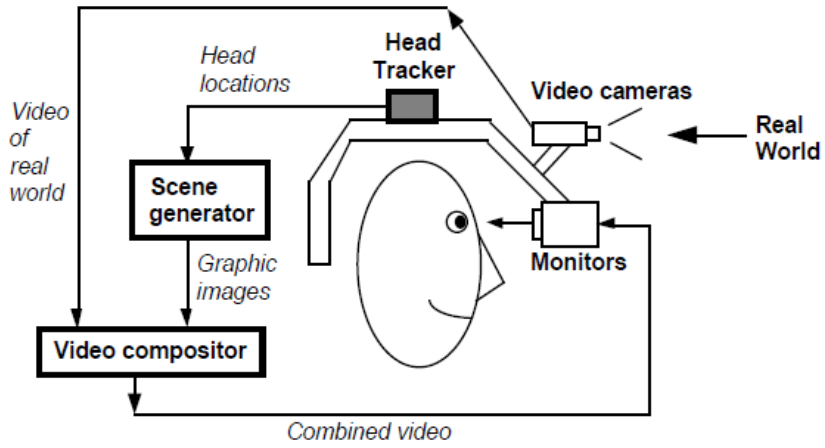
Kelebihan teknik display *Head-Attached Display* ini adalah lebih nyaman ke pengguna, karena citra yang terbentuk mengikuti sudut pandang pengguna.

2.3.1.1 *Head-Mounted Display*

Head-Mounted Display (HMD) menggabungkan citra dari objek virtual dan objek nyata dan menampilkannya langsung ke mata pengguna melalui suatu alat yang dipasang di kepala pengguna. Terdapat dua tipe utama perangkat HMD yang digunakan dalam aplikasi realitas tertambah, yaitu *optical-see-through* HMD dan *video see-through* HMD. Keduanya digunakan untuk berbagai jenis pekerjaan dan memiliki keuntungan dan kerugian masing-masing. Dengan *optical-see-through* HMD, lingkungan nyata dilihat melalui cermin semi transparan yang diletakkan di depan mata pengguna. Cermin tersebut juga digunakan untuk merefleksikan citra yang dibentuk oleh komputer ke mata pengguna, menggabungkan lingkungan nyata dan virtual. Dengan *video see-through*

HMD, lingkungan nyata direkam mengunakan dua kamera video yang terintegrasi ke alat, seperti gambar 2.6, dan citra yang dibentuk komputer digabung dengan video tadi untuk merepresentasikan lingkungan yang akan dilihat pengguna [4].

- *Video-see-through Head-Mounted Display*



Gambar 2.3: Diagram *Opaque* HMD

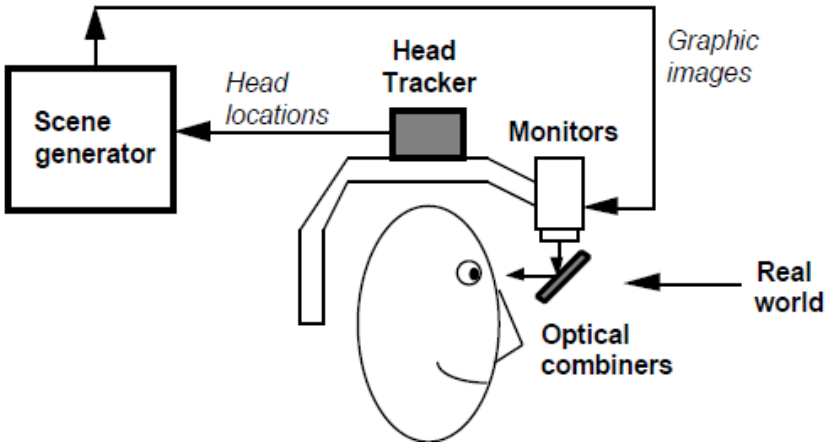
Video see-through HMD bekerja dengan menggabungkan sebuah *closed-view* HMD dengan satu atau dua *head-mounted* kamera video, melalui kamera video tersebut pengguna melihat ke lingkungan nyata. Video dari kamera dikombinasikan dengan citra yang dibuat oleh *scene generator*, dunia nyata dan virtual digabungkan. Hasilnya dikirimkan ke monitor yang terletak di depan mata pengguna. Gambar 2.3 menunjukkan konsep dari *Video see-through* HMD, gambar 2.4 adalah contoh *Video see-through* HMD, dengan dua video terintegrasi di bagian atas Helm [1].

- *Optical see-through Head-Mounted Display*

  Tidak seperti penggunaan *video see-through* HMD, *optical see-through* HMD menyerap cahaya dari lingkungan luar, sehingga memungkinkan pengguna untuk secara langsung mengamati dunia nyata dengan mata (gambar 2.5). Selain itu, sebuah sistem cermin yang diletakkan di

Gambar 2.4: Contoh *Opaque* HMD

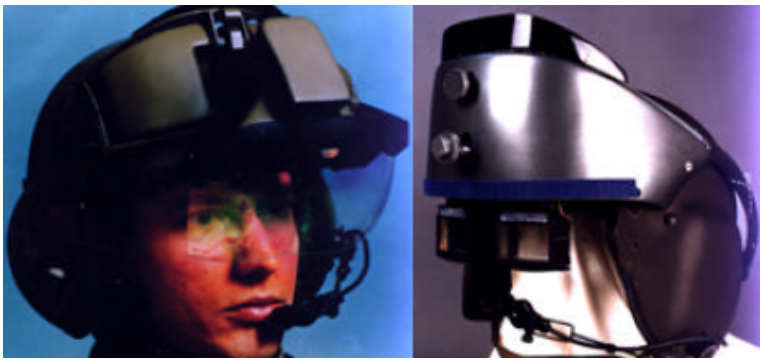
depan mata pengguna memantulkan cahaya dari pencitraan grafis yang dihasilkan komputer. Pencitraan yang dihasilkan merupakan gabungan optis dari pandangan atas dunia nyata dengan pencitraan grafis [1].



Gambar 2.5: Diagram *see-trough* HMD

2.3.1.2 *Head-Mounted Projectors*

Head-Mounted Projectors Menggunakan proyektor atau panel LCD kecil dan mempunyai cahaya sendiri untuk menampilkan citra langsung ke lingkungan nyata [3]. Seperti yang ditunjukkan oleh gambar 2.7.

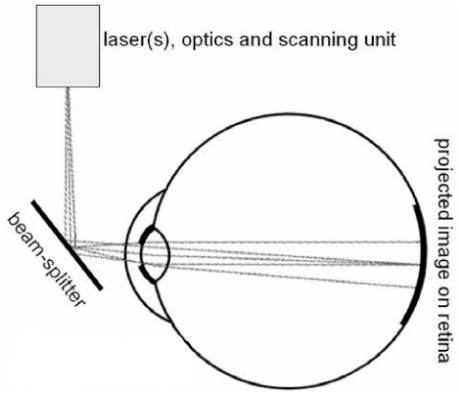


Gambar 2.6: Contoh *see-through* HMD, dibuat oleh Hughes Electronics

Gambar 2.7: Ilustrasi penggunaan dua jenis perangkat HMD yang digunakan untuk menampilkan data dan informasi tambahan

2.3.1.3 *Virtual Retina Display*

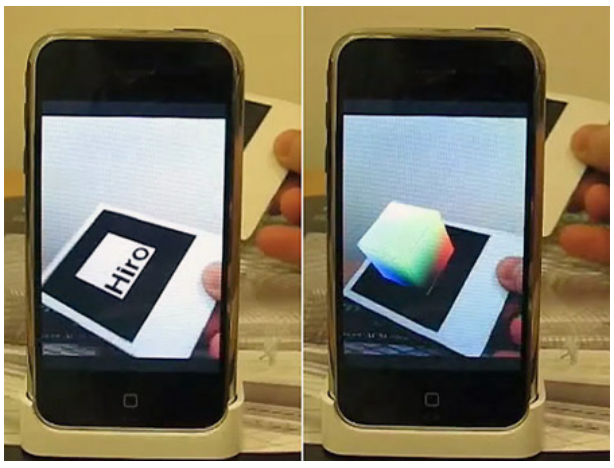
Virtual retina display (VRD), atau disebut juga dengan *retinal scanning display* (RSD), memproyeksikan cahaya langsung kepada retina mata pengguna[5]. VRD dapat menampilkan proyeksi citra yang penuh dan juga tembus pandang tergantung pada intensitas cahaya yang dikeluarkan, sehingga pengguna dapat menggabungkan realitas nyata dengan citra yang diproyeksikan melalui sistem penglihatannya. VRD dapat menampilkan jarak pandang yang lebih luas daripada HMD dengan citra beresolusi tinggi[6]. Keuntungan lain VRD adalah konstruksinya yang kecil dan ringan. Namun, VRD yang ada kini masih merupakan prototipe yang masih terdapat dalam tahap perkembangan, sehingga masih belum dapat menggantikan HMD yang masih dominan digunakan dalam bidang AR. Gambaran sederhana VRD ini dapat dilihat pada gambar 2.8.



Gambar 2.8: Diagram sederhana *virtual retina display*

2.3.2 *Handheld Display*

Teknik ini menggunakan alat dengan display yang dengan mudah dapat digenggam pengguna (Tablet PC, PDA dan telepon genggam) seperti yang ditunjukkan pada gambar 2.9. Sensor dapat berupa GPS, kompas digital ataupun kamera yang ada pada *handheld* tersebut. Semua penerapan AR pada perangkat genggam menggunakan kamera untuk menggabungkan citra digital dengan lingkungan nyata, *Handheld* AR sangat menjanjikan untuk tujuan komersial. Dua kelebihan utama dari *Handheld* AR adalah mobilitas perangkat yang mudah dan salah satu perangkat genggam yang banyak digunakan (telepon genggam) telah banyak dilengkapi kamera.



Gambar 2.9: Contoh augmented reality dengan *handphone*

2.3.3 *Spatial Display*

Dalam *Spatial Augmented Reality* (SAR), objek nyata digabungkan langsung dengan citra yang terintegrasi langsung ke lingkungan nyata. Contohnya, citra diproyeksikan ke lingkungan nyata menggunakan proyektor digital atau tergabung dengan lingkungan menggunakan panel display [7]. Perbedaan utama pada SAR dibanding teknik display sebelumnya adalah displaynya terpisah dengan pengguna. SAR memiliki kelebihan dari HMD dan handheld, sistem ini bisa digunakan oleh banyak orang pada waktu bersamaan tanpa perlu mengenakan suatu alat.

Ada tiga teknik display dalam SAR [3], yaitu sebagai berikut:

1. *Screen-Based Video See-Through Displays*
   Screen-based AR menggabungkan citra dan lingkungan nyata yang ditampilkan ke sebuah monitor, seperti yang ditunjukkan pada gambar 2.10.

Gambar 2.10: Contoh *Screen-Based Video See-Through Displays*

2. *Spatial Optical See-Through Displays*
   Sistem ini menghasilkan citra yang ditampilkan langsung ke lingkungan nyata. Komponen yang penting dalam sistem ini meliputi *spatial optical combiners* (*planar* atau *curved beam combiners*), layar transparan atau hologram.

3. *Projection-Based Spatial Displays*
   Sistem ini memproyeksikan citra secara langsung pada permukaan objek fisik daripada menampilkannya pada sebuah bidang pencitraan dalam penglihatan pengguna. Sistem ini menggunakan banyak proyektor yang digunakan untuk meningkatkan wilayah tampilan serta meningkatkan kualitas citra.

2.4 Model Tiga Dimensi (3D)

Pemodelan Tiga Dimensi (3D) (3D *modeling* atau dikenal juga dengan *meshing*) adalah proses pembuatan representasi matematis permukaan tiga dimensi dari suatu objek dengan *software* tertentu. Produk hasil pemodelan itu disebut model 3D. Model 3D tersebut dapat ditampilkan sebagai citra dua dimensi melalui sebuah proses yang disebut *3D rendering*. Model 3D direpresentasikan dari kumpulan titik dalam 3D, terhubung oleh berbagai macam entitas geometri, seperti segitiga, garis, permukaan lengkung, dan lain sebagainya. Berdasarkan hal tersebut, model 3D bisa dibuat manual (seperti seni memahat), secara algoritma (pemodelan prosedural), atau *scanning*. Hasil akhir dari citra 3D adalah sekumpulan poligon. Model dengan jumlah poligon yang lebih banyak memerlukan waktu yang lebih lama untuk di-*render* oleh komputer, karena setiap permukaan memiliki tekstur dan *shading* tersendiri.

2.5 Aplikasi Komputer Berbasis Web (*Rich Internet Application*)

Pada tahun 90-an, "mengakses *web*" berarti mengakses tulisan dan gambar statis secara *online* . Sejalan dengan berkembangnya koneksi internet, kebutuhan akan suatu konten yang lebih kaya, responsif juga meningkat. Pada tahun 2002, Macromedia menemukan istilah *Rich Internet applications* (RIAs). RIAs menggabungkan fleksibilitas, tingkat responsif, dan kemudahan aplikasi berbasis desktop. RIAs atau dikenal dengan aplikasi berbasis *web* adalah aplikasi yang mempunyai karakteristik seperti aplikasi *desktop*, biasanya didis-

tribusikan atau diakses lewat *browser web* standar, menggunakan *web plug-in*. Contoh RIAs meliputi Ajax, Curl, GWT, Adobe Flash/Adobe Flex/AIR, Java/JavaFX, Mozilla's XUL dan Microsoft Silverlight.

Aplikasi berbasis *web* semakin banyak dikembangkan, karena kelebihan aplikasi *web* yang sangat baik untuk aplikasi dengan model *client-server*. Beberapa kelebihan aplikasi *web* dibanding aplikasi *desktop*:

1. *Running anywhere* (berjalan/beroperasi dimana saja), cukup instalasi pada satu *server* dan tanpa perlu instalasi apapun selama ada *web browser* di sisi *client*, pengguna langsung dapat menjalankan aplikasi tersebut tanpa konfigurasi apapun. Secara default, *web browser* pada sistem operasi apapun telah tersedia sehingga kebutuhan akan *web browser* untuk menjalankan aplikasi bukanlah suatu kendala berarti.

2. *Easy to update* (mudah untuk diubah/diperbaharui), cukup *update* pada sisi *server* maka aplikasi pada *client* akan langsung menggunakan versi ter-*update* tanpa harus instalasi dulu pada *client*.

3. *Requirement* (kebutuhan)pada *client* tidak terlalu besar. Karena *running* aplikasi bersifat *stateless* dan pada sisi *client* hanya sebagai *interface* maka spesifikasi hardware pada *client* tidak harus canggih.

4. Tampilan yang lebih beragam.

2.6 Adobe Flash *Platform*

Kebanyakan *designer* dan *developer* menggunakan Adobe Flash ataupun Adobe Flex, yang merupakan bagian dari platform Adobe Flash, untuk mengembangkan RIAs. Flash merupakan suatu *environment* untuk membuat konten yang interaktif dan kaya fitur dalam dunia *web*. Begitu juga Flex merupakan sebuah *framework cross-platform* untuk mengembangkan RIAs. Konten yang dibuat dengan Flash dan Flex di-*deploy* menggunakan Adobe Flash Player

2.6.1 Adobe Flash

Adobe Flash (dulunya Macromedia Flash) adalah platform multimedia yang aslinya dibuat oleh Macromedia dan saat ini dikembangkan dan didistribusikan oleh Adobe Systems. Sejak pengenalannya pada Tahun 1996, Flash telah menjadi metode yang popular untuk menambahkan animasi dan interaktivitas ke halaman web. Komponen Flash untuk mengintegrasikan video ke halaman web, dan yang terbaru saat ini, untuk mengembangkan RIAs.

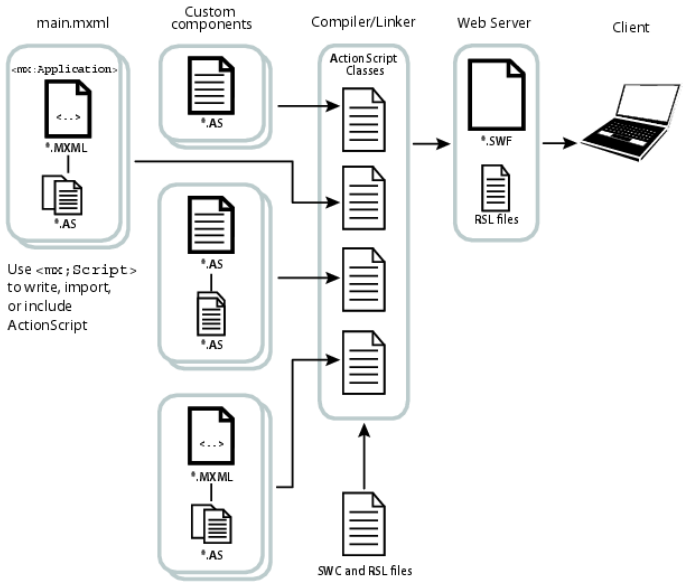
Flash dapat memanipulasi vector dan raster grafik, serta mendukung *streaming* dua arah audio dan video. Flash menggunakan bahasa *script* yang disebut Action Script. Banyak produk *software*, sistem dan *device* dapat menampilkan konten flash, contohnya Adobe Flash player, yang tersedia gratis bagi sebagian besar *web browser*. Beberapa ponsel dan alat elektronik lainnya juga dapat menampilkan konten Flash, menggunakan Flash lite.

File dalam format SWF, biasanya disebut "ShockWave Flash movies", "Flash movies" atau "Flash games", yang biasanya memiliki sebuah ekstensi .swf dan dapat menjadi objek di halaman *web*. *File* tersebut pada dasarnya dijalankan dengan Flash Player itu sendiri atau digabungkan dengan "Projector" (*video flash* yang dapat berjalan sendiri dengan ekstensi .exe di Microsoft Windows atau .hqx untuk Macintosh). *File* Flash Video memiliki ekstensi .flv dan juga digunakan dalam .swf atau dijalankan melalui aplikasi yang dapat menjalankan *file* flv.

2.6.2 Adobe Flex

Adobe Flex adalah paket pengembangan *software* yang dirilis oleh Adobe systems untuk pengembangan dan aplikasi *cross-platform rich internet* berbasis platform adobe flash. Aplikasi flex dapat ditulis menggunakan Adobe Flash Builder sebagai IDE (*Integrated Development Evironment*) dan menggunakan Flex *Software Development Kit* (SDK) yang tersedia gratis dari Adobe.

Flex membuat *workflow* dan model pemrograman yang familiar terhadap *developer* aplikasi berbasis *web*. FLex menggunakan MXML dan ActionScript. MXML merupakan sebuah bahasa yang berbasis XML, menawarkan cara mem-



Gambar 2.11: Diagram Kompilasi Flex

bangun dan menata *user interface*. Interaktivitas dicapai melalui penggunaan ActionScript, yaitu bahasa utama Flash Player.

Flex menyediakan dua kompiler: mxmlc dan compc. Kompiler compc dan mxmlc bisa dijalankan dari Flex Builder atau pun dari *command line*. compc digunakan untuk mengkompilasi komponen, kelas, and file lainnya ke file SWC atau RSL yang digunakan sebagai *library* untuk aplikasi. mxmlc untuk mengkompilasi ActionScript dan SWC ke *file* SWF, seperti ditunjukkan pada gambar 2.11. Setelah aplikasi dikompilasi dan di-*deploy* ke *web server*, pengguna bisa mengakses file SWF tersebut dan dijalankan via *web browser*.

2.6.3 ActionScript

ActionScript merupakan bahasa pemrograman berorientasi objek yang berdasarkan ECMAScript (bahasa yang distandarisasi oleh Ecma International dalam spesifikasi ECMA-262 dan ISO/IEC 16262). ActionScript terutama digunakan untuk pengembangan *website* dan *software* menggunakan Adobe Flash Player (dalam bentuk *file* SWF yang diintegrasikan ke halaman *web*), ActionScript juga digunakan pada beberapa aplikasi untuk *database* (seperti

Alpha Five). ActionScript pada awalnya didesain untuk mengatur animasi vektor 2D sederhana yang dibuat di Adobe Flash, dengan berkembangnya. Versi terakhir dari ActionScript menambahkan kemungkinan penggunaan untuk pembuatan *web* berbasis *game* dan RIAs dengan media *streaming* (seperti video dan audio).

2.7 *Library* Pendukung Augmented Reality Pada Platform Flash

Library atau Pustaka, dalam ilmu komputer adalah koleksi dari rutin-rutin program yang digunakan untuk membangun dan mengembangkan *software*. *Library*, umumnya mengandung kode program dan data pembantu (banyak *programmer* menyebutnya sebagai *helper*), yang menyediakan layanan-layanan kepada program-program independen. Penggunaan *library* sangat diperlukan untuk mengembangkan aplikasi AR secara cepat. *Library* AR yang dikenal luas dalam platform Flash adalah FLARToolKit.

2.7.1 FLARToolKit

FLARToolKit adalah *tracking system library* yang bersifat *open-source* sehingga memungkinkan *programmer* dengan mudah mengembangkan aplikasi AR, FLARToolKit merupakan *porting* (perubahan terhadap *software* untuk menjadikannya dapat digunakan di lingkungan yang berbeda) yang paling terakhir dari ARToolkit, yaitu sebuah libary AR C++ yang awalnya dikembangkan oleh Dr. Hirokazu Kato di Human Interface Technology Lab University of Washington. Dengan datangnya ActionScript 3.0, para pengembang seperti Mario Klingemann dan lainnya mulai bereksperimen dengan teknik analisis image secara *real-time* untuk Flash Player. Saqoosha meneruskan hal ini, dan memporting FLARToolKit dari NYARToolkit (sebuah Java/C-sharp/Android *port* dari ARToolkit).

FLARToolKit hanya merupakan *library* untuk *tracking* pada AR, untuk menampilkan objek 3D di lingkungan Flash, FLARToolKit memerlukan sebuah

library 3D. Beberapa *library* 3D yang didukung oleh FLARToolKit adalah sebagai berikut:

- Alternativa3D
- Away3D
- Away3D Lite
- Papervision3D
- Sandy3D

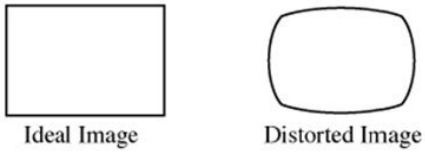
Proses FLARToolKit secara garis besarnya sebagai berikut:

1. Mengambil video dari *webcam*.
2. Binarisasi citra masukan(*thresholding*).
3. Memberi penanda (*labelling*).
4. Deteksi area persegi (*Marker Outline Detection*).
5. Pencocokan pola.
6. Menghitung transform matrix.
7. Merender obyek 3D.

2.7.1.1 Mengambil Video dari *Webcam*

Langkah awal yang harus dilakukan adalah mendapatkan masukan video dari sebuah *webcam*, seperti yang ditunjukkan gambar 2.12. Video yang di-*streaming* secara *real-time* ini akan diolah oleh sistem untuk dianalisa *frame* per *frame*. Sebelum *webcam* digunakan, *webcam* harus dikaliberasi terlebih dahulu. Kaliberasi *webcam* merupakan bagian yang sangat penting dalam proses pengambilan masukan video. Hal ini disebabkan oleh distorsi pada lensa *webcam* yang tiap-tiap kamera berbeda karakteristiknya (gambar 2.13). Tujuan dari kalibrasi *webcam* adalah untuk menghitung tingkat distorsi dari sebuah lensa *webcam* yang digunakan agar citra yang dihasilkan mendekati citra ideal.

Gambar 2.12: Mengambil citra dari *webcam*



Gambar 2.13: Perbandingan antara citra yang ideal dengan citra yang disebabkan oleh faktor distorsi

2.7.1.2 Binarisasi Citra Masukan (*Thresholding*)

Langkah pertama pada aplikasi visi komputer yang terletak pada deteksi tepi adalah untuk men-*threshold* sumber citra atau disebut juga binarisasi seperti yang ditunjukkan gambar 2.14. *Thresholding* mengkonversi citra ke citra binari sehingga memudahkan untuk komputasi. Sebuah citra binari dibuat dengan mengubah pixel yang lebih cerah daripada nilai *threshold* ke suatu warna, dan pixel yang lebih gelap daripada nilai *threshold* ke suatu warna lainnya (didefinisikan sebagai *gray-scale* atau hitam-putih). Nilai *threshold* berada pada angka 0 255 dan secara default, *threshold* bernilai 100. Fungsi dari proses ini adalah untuk membantu sistem agar dapat mengenali bentuk segi empat dan pola di marker pada citra yang diterima. Nilai *threshold* dapat dirubah dan disesuaikan dengan kondisi cahaya disekitar *marker* untuk tetap membuat *marker* terlihat sebagai segi empat, karena ketika cahaya disekitar *marker* berkurang ataupun berlebih pada saat proses *thresholding*, sistem tidak dapat mendeteksi *marker*.

Gambar 2.14: Thresholding

2.7.1.3 Memberi Penanda (*Labeling*)

Langkah berikutnya dari FLARToolKit adalah menemukan area yang berdampingan dalam citra yang di-*treshold*, khususnya dalam area dibawah *threshold* (area yang lebih gelap). Area yang berdampingan diberi tanda dengan warna yang berbeda dengan tujuan untuk mengidentifikasi area, proses *labelling* dapat dilihat pada gambar 2.15.

Gambar 2.15: Setiap area putih ditandai dengan warna yang berbeda.

2.7.1.4 Deteksi Area Persegi (*Marker Outline Detection*)

Langkah selanjutnya, FLARToolKit mencari area yang kemudian ditandai sebagai persegi *(marker outline)*. Setelah citra mengalami proses *thresholding* dan *labelling*, FLARToolkit akan mengenali bentuk dan pola yang ada pada marker. FLARToolkit akan mencari bagian yang memiliki bentuk segi empat dan menandainya. FLARToolKit juga akan menghilangkan area yang tidak berbentuk segi empat sehingga yang akan ditampilkan pada layar hanyalah area yang memiliki bentuk segi empat (gambar 2.16.

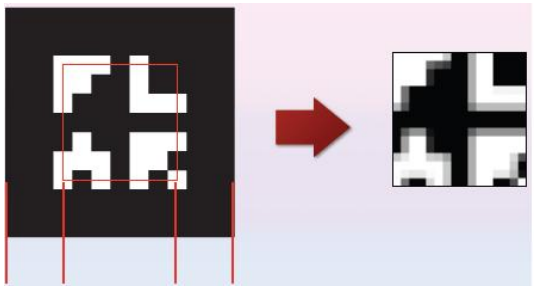
Gambar 2.16: Mencari area persegi (Marker Outline Detection)

2.7.1.5 Pencocokan Pola

Setelah semua area persegi ditandai, FLARToolKit menganalisa citra yang berada di dalam persegi dan membandingkan polanya dengan sekumpulan pola yang telah ditentukan (pencocokan pola). FLARToolKit mengekstrak pola di dalam persegi menggunakan transformasi *homography*. FLARToolKit memberikan sebuah nilai '*confidence*' kepada setiap pola yang cocok, jika kecocokannya di atas nilai yang telah ditentukan maka polanya dinyatakan cocok.

Spesifikasi pola marker (gambar 2.17):

- Harus berupa persegi.
- Hanya 50% dari tengah area yang digunakan untuk proses pencocokan pola.



Gambar 2.17: Spesifikasi pola marker

- Pola marker secara *default*-nya adalah 16 x 16 titik.
- Ukuran pola bisa lebih besar, tapi membutuhkan waktu yang lebih lama untuk diproses.

2.7.1.6 Menghitung Transformasi Matriks

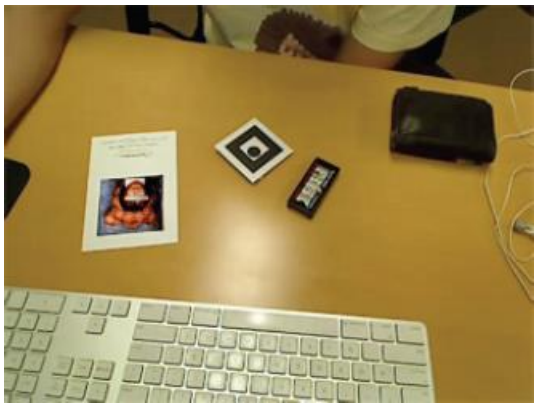
Transformasi matriks dihitung dari titik-titik persegi marker yang dideteksi. Matriks tersebut digunakan untuk proses *render* objek 3D.

2.7.1.7 Me-*render* Objek 3D

FLARToolKit menggunakan transformasi matriks yang dikalkulasikan di step sebelumnya dan menampilkan objek yang sesuai dengan sebuah *library* 3D, seperti yang ditunjukkan gambar 2.18. FLARToolKit menyertakan kelas pendukung yang mengkonversikan transformasi matriks FLARTollKit ke setiap kelas matriks internal *library* 3D tersebut.

2.7.2 Papervision3D

Salah satu *library* 3D (biasa disebut juga dengan *engine* 3D) untuk platform Flash adalah Papervision3D, Papervison3D merupakan *library* 3D yang bersifat *open source*. Dengan Papervision 3D, model 3D bisa dibuat di platform Flash menggunakan kelas-kelas dalam ActionScript ataupun juga dengan meng-*import* model yang dibuat dari *software* pemodelan 3D. Model yang telah dibuat atau di-*import*, bisa dikendalikan dengan ActionScript.



Gambar 2.18: Render objek 3D

2.7.3 FLARManager

Visi komputer dalam konteks *web* memiliki banyak kesulitan untuk diterapkan. Masalah utama ialah kurangnya kendali terhadap kondisi lingkungan *end-user*. Kurangnya atau ketidakteraturan pencahayaan membuat analisis dari sebuah citra sangat sulit, dan permasalahan tersebut berdampak pada FLARToolKit. Selain itu, FLARToolKit sangat sulit digunakan karena minimnya dokumentasi, dan *developer* harus membuat sebuah *manager* untuk mengatur konfigurasi kamera, marker dan juga objek virtual yang akan ditampilkan.

FLARManager adalah sebuah *framework* sederhana yang memudahkan untuk membuat aplikasi AR dengan Flash terutama yang menggunakan *libray* FLARToolKit. *Framework* merupakan kumpulan komponen kelas sekaligus kerangka dalam pemrograman yang memudahkan *programmer* untuk membuat suatu aplikasi yang siap pakai. FLARManager bisa meningkatkan akurasi dan reabilitas dari proses dengan hanya merubah konfigurasinya yang berupa file dengan format xml. FLARManager dapat mendeteksi lebih dari satu *marker* dan pola dalam waktu bersamaan, sehingga bisa menampilkan objek virtual lebih dari satu.

FLARToolkit menyediakan akses ke sumber citra yang diprosesnya, dan juga hasil dari FLARToolkit setelah di analisa. Dengan demikian, FLARManager bisa difokuskan pada fungsionalitas dalam deteksi dan *tracking* dari *mar-*

ker. FLARManager sebisa mungkin menghindari perubahan terhadap FLARToolkit. Dengan tidak tergantungnya FLARManager terhadap FLARToolkit, setiap proyek bisa terus dikembangkan secara terpisah, dan FLARManager secara teori bisa diterapkan pada *libray* Flash AR lainnya.

Sampai saat tulisan ini dibuat, FLARManager telah mendukung *tracking library* sebagai berikut :

- FLARToolkit
- flare*tracker
- flare*NFT

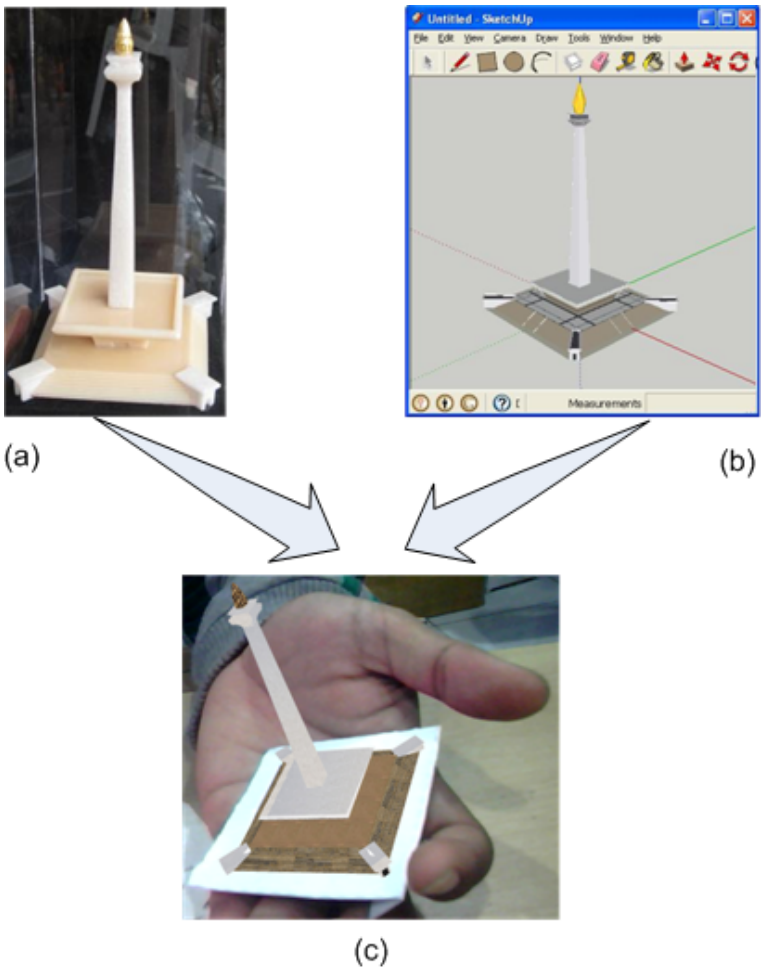
Bab 3

PERANCANGAN DAN IMPLEMENTASI

3.1 Deskripsi

Secara konvensional model suatu objek nyata (misalnya gedung) biasanya disajikan dengan membuat sebuah miniatur atau maket seperti yang ditunjukkan gambar 3.1.a. Dengan adanya model atau maket tersebut, seseorang bisa dengan mudah melihat atau mengetahui objek yang dimodelkan tersebut secara nyata, akan tetapi pembuatan model atau maket tersebut membutuhkan waktu yang lama serta biaya yang mahal. Dengan menggunakan komputer dan *software* tertentu pemodelan objek bisa disajikan secara virtual seperti terlihat pada gambar 3.1.b. Dengan menggunakan *software*, pemodelan objek memang lebih menghemat biaya serta waktu, namun interaksi dengan dunia nyata sangat kurang. Aplikasi *viewer* model 3D dengan teknologi *Augmented Reality* yang penulis rancang ini menggabungkan kedua kelebihan pemodelan secara nyata dan *software*, yaitu kemudahan pembuatan model 3D (secara virtual) dan juga interaksi terhadap dunia nyata seperti diilustrasikan pada gambar 3.1.c.

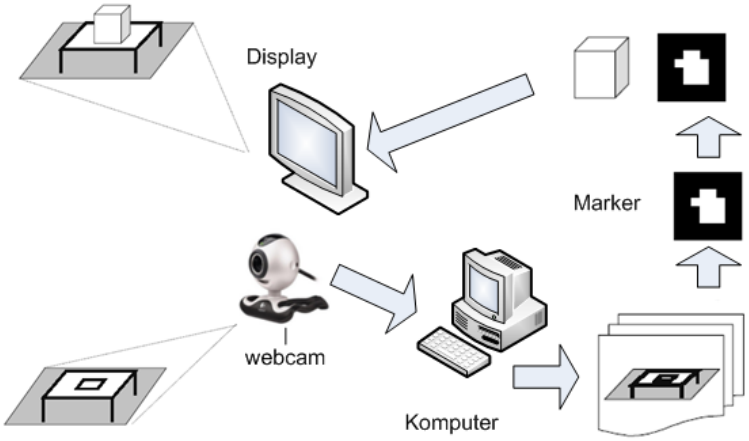
Aplikasi yang dirancang dengan teknologi AR ini, seolah-olah menggabungkan objek virtual dengan objek nyata, dalam hal ini objek nyatanya berupa gambar dengan pola tertentu (disebut *marker*) dan objek virtualnya berupa model 3D.



Gambar 3.1: Gambaran aplikasi *viewer* objek 3D virtual dengan *Augmented Reality*

Sistem menggunakan teknik *spatial display* (pembahasan pada subbab 2.3.3 dengan *screen display* (bisa berupa monitor ataupun proyektor). Secara garis besarnya gambaran sistem dapat dilihat pada gambar 3.2.

webcam berfungsi sebagai media *visi* bagi aplikasi AR untuk mendapatkan video masukan. Kamera mengambil *frame-frame* video untuk dapat diterima oleh komputer. Komputer memproses citra digital yang diakuisisi oleh *webcam*, *frame* demi *frame*. Komputer akan mendeteksi pola yang mirip dengan *marker* dari setiap *frame* video tersebut, yang kemudian lokasi dan rotasi dari *marker* dapat ditentukan. Dengan informasi tersebut, objek *virtual* digabungkan



Gambar 3.2: Gambaran sistem secara umum

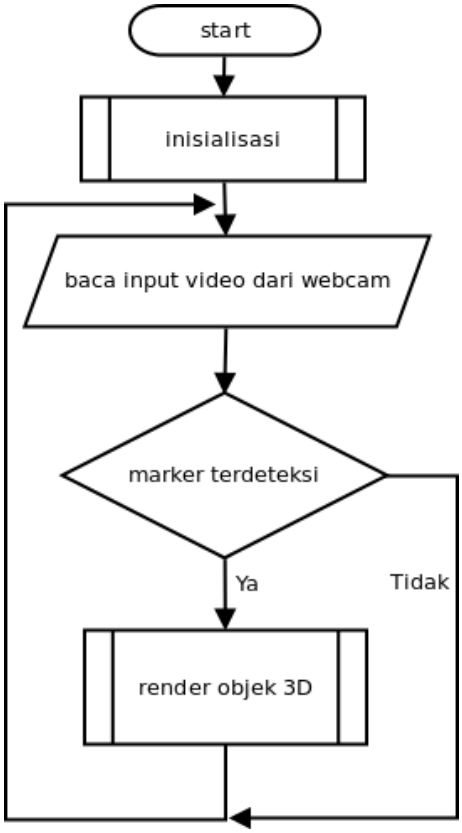
dengan video dari *webcam* dan me-*render*-nya sesuai dengan informasi posisi yang diperoleh dari *marker* tersebut. Dengan demikian, *marker* tersebut seolah-olah berfungsi sebagai *tracker* untuk model 3D virtual. Proses tersebut berlangsung secara *real-time* sehingga model 3D virtual yang tampil di media *display* akan mengikuti pergerakan *tracker*.

3.2 Perancangan Aplikasi

Aplikasi yang dirancang merupakan aplikasi berbasis *web* atau disebut juga *Rich Internet Applications* (RIAs), tujuannya agar mudah diakses dari komputer atau juga *handheld* tanpa perlu instalasi aplikasi. Berdasarkan penjelasan pada subbab 2.6, penulis memilih Adobe Flash sebagai *Framework* RIAs dengan SDK (*software Development Kit*) Adobe Flex. Bahasa script yang digunakan adalah ActionScript yang merupakan bahasa pemrograman berorientasi objek. ActionScript mendukung *event-driven programming* (Pemrograman berbasis *event*). *event-driven programming* merupakan paradigma pemrogaman yang alur program ditentukan dari *event*, *input* sensor atau pesan dari program lain. Paradigma ini sangat cocok untuk pengembangan aplikasi ini, dikarenakan aplikasi menerima input dari kamera atau *webcam*.

Aplikasi ini dibangun menggunakan FLARManager, seperti telah dijelaskan

pada subbab 2.7.3, FLARManager dapat mempermudah dan mempercepat pengembangan aplikasi yang menggunakan *library* FLARToolKit. *Library* FLARToolKit merupakan salah satu *tracking library* AR di lingkungan Flash (subbab 2.7.1). Di dalam FLARManager sudah termasuk pengaturan kamera untuk masukan video, kelas untuk mengatur pola marker yang digunakan, dan juga kelas untuk memudahkan interaksi dengan *library* engine 3D, dengan demikian aplikasi yang dibangun tidak perlu berhubungan langsung dengan *library* FLARToolKit.



Gambar 3.3: Diagram alir aplikasi secara umum

Secara keseluruhan, aplikasi dapat digambarkan dengan diagram alir pada gambar 3.3. Aplikasi melakukan inisialisasi terlebih dahulu sebelum melakukan *tracking marker*, *marker* dideteksi dari masukan video *webcam*, jika *marker* terdeteksi maka objek 3D di-*render*.

Secara garis besarnya, dalam perancangan aplikasi ini ada tiga bagian utama

yaitu sebagai berikut:

- Inisialisasi
  Inisialisasi semua hal yang diperlukan untuk proses aplikasi (*marker*, objek 3D, engine 3D)

- *Tracking marker*
  Marker yang digunakan lebih dari satu pola, setiap *marker* dengan pola tertentu akan menjadi *tracker* objek 3D tertentu pula.

- *Rendering* objek 3D
  Objek 3D dirender dan diatur posisinya sesuai dengan posisi *marker* yang terdeteksi.

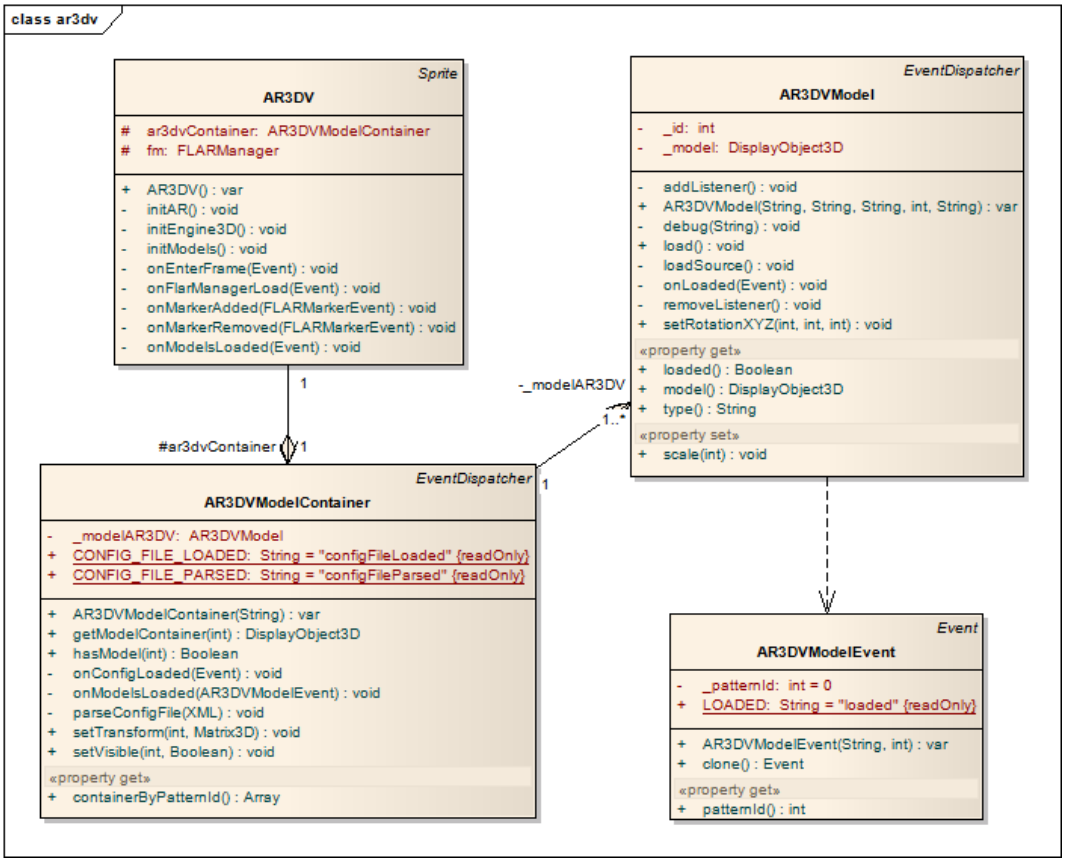
Penulis merancang aplikasi ini atas beberapa kelas (gambar 3.4), agar memudahkan pengembangan aplikasi. Kelas utama adalah **AR3DV**, yang mengatur jalannya aplikasi, mulai dari inisialisasi hingga *render* objek 3D.

Tiga kelas lainnya berhubungan dengan objek 3D dan inisialisasinya, yaitu sebagai berikut:

- AR3DVModelContainer
  Kelas yang mengatur objek 3D apa saja yang di-*load*, serta menyediakan objek 3D yang dapat dipanggil dari kelas utama. **AR3DVModelContainer** menggunakan file konfigurasi dengan format xml untuk model objek 3D yang akan di-*load*.

- AR3DModel
  Kelas yang berfungsi untuk me-*load* objek 3D, serta mengatur rotasi dan skala objek.

- AR3DModelEvent
  Loading model memerlukan waktu yang cukup lama tergantung ukuran *file* objek 3D. Karena itu *load* objek 3D dilakukan secara asinkron. Kelas **AR3DVModelContainer** akan me-*load* semua *file* objek 3D dengan bantuan kelas **AR3DModel**. Agar kelas **AR3DVModelContainer** mengetahui objek 3D tertentu telah di-*load*, diperlukan sebuah pesan

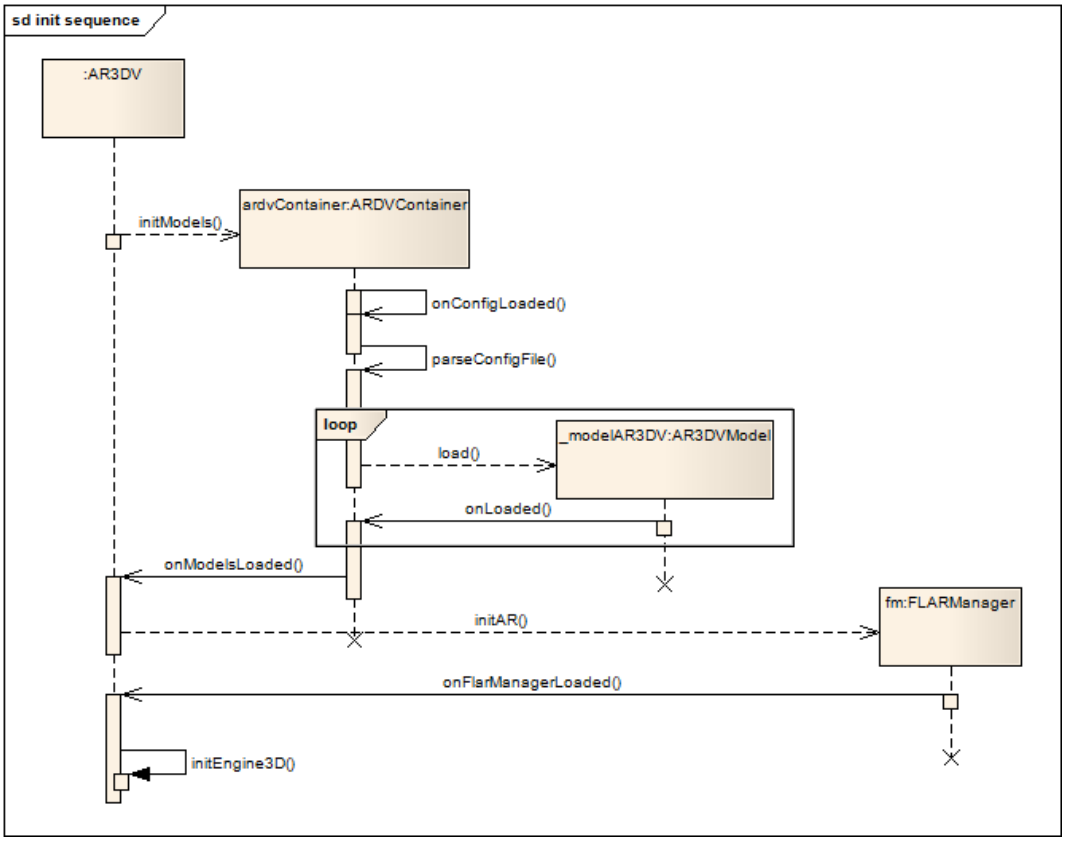


Gambar 3.4: Diagram kelas aplikasi

yang disebut juga **Event**. Penulis membuat kelas **AR3DModelEvent** agar bisa digunakan **AR3DModel** untuk menyampaikan pesan ke kelas **AR3DModelContainer**.

3.2.1 Inisialisasi

Pada tahap ini ditentukan *marker* yang akan digunakan, sumber input video nya, objek 3D yang akan digunakan serta *engine* 3D yang digunakan untuk me-*render* objek 3D. Pada bagian inisialisasi ini, objek 3D diinisialisasi terlebih dahulu karena *loading* objek 3D memerlukan waktu yang cukup lama. Setelah objek 3D di-*load*, kemudian FLARManager dan *engine* 3D diinisialisasi. Deskripsi inisialisasi digambarkan oleh *sequence diagram* gambar 3.5.



Gambar 3.5: *Sequence diagram* inisialisasi

3.2.1.1 Inisialisasi Model 3D

Model 3D yang akan ditampilkan di-*load* terlebih dahulu. Agar aplikasi dapat menampilkan objek 3D tertentu tanpa merubah atau membangun ulang aplikasi, diperlukan sebuah *file* konfigurasi untuk menentukan objek 3D yang akan di-*load* sesuai dengan pola *marker* yang dideteksi. File konfigurasi itu berisi informasi format file objek 3D yang digunakan, skala objek 3D dan juga rotasi terhadap koordinat xyz sehingga penampilan objek 3D bisa lebih proporsional. Dengan adanya file konfigurasi tersebut, objek 3D dan *marker* yang digunakan dapat diatur dengan mudah.

Method untuk inisialisasi model 3D ini adalah **initModels**. Method ini akan membuat objek baru **AR3DVModelContainer**, yang akan me-*load* objek-objek 3D sesuai konfigurasi file xml nya. Objek 3D di-*load* satu persatu dengan

bantuan **AR3DModel**, setelah semua objek ter-*load*, sebuah pesan akan dikirim ke kelas utama (**AR3DV**) dan proses selanjutnya dapat dijalankan.

3.2.1.2 Inisialisasi FLARManager

FLARManager merupakan inti dari aplikasi ini. FLARManager akan mengatur semua hal yang berkaitan dengan AR, mulai dari *marker*, video masukan dari kamera, dan pengenalan *marker* dengan bantuan *library* FLARToolKit. FLARManager diinisialisasi dengan membuat objek baru FLARManager yang memerlukan file konfigurasi berupa file berformat xml dan objek *tracker* (dalam hal ini FLARToolKit), kelas *tracker* yang digunakan untuk objek *tracker* adalah **FLARToolkitManager** yang juga telah tersedia dalam *framework* FLARManager. Inisialisasi FLARManager terdapat pada method **initAR**, setelah FLARManager selesai di inisialisasi, method **onFlarManagerLoad** dijalankan.

3.2.1.3 Inisialisasi *Engine* 3D

Untuk merender objek 3D, terutama objek 3D dari file, diperlukan *library* pendukung atau disebut juga *engine* 3D Flash. Penulis menggunakan Papervision3D (penjelasan mengenai Papervision3D dapat dilihat pada subbab 2.7.2) sebagai *library engine* 3D nya. Sebelum merender objek 3D, Papervision3D juga perlu diinisialisasi terlebih dahulu, inisialisasi engine 3D terdapat pada method **initEngine3D**.

3.2.2 *Tracking* Marker

Bagian ini merupakan inti dari AR, *library* FLARToolKit bekerja pada bagian ini. Marker yang akan digunakan sebagai *tracker* telah ditentukan pada bagian inisialisasi. Di dalam FLARManager telah terdapat kelas tersendiri yang menangani proses *sampling* video, integrasi dengan FLARToolKit untuk pengenalan pola dan menentukan posisi *marker*-nya. Ketika sebuah *marker* terdeteksi, maka method yang mengatur *tracking marker* **onMarkerAdded** dan **onMarkerRemoved** dijalankan. Kedua method tersebut menentukan apa

yang akan dilakukan aplikasi ketika *marker* ditemukan. Jika *marker* terbaca oleh sistem, method *set* **visible** yang berfungsi untuk memunculkan objek 3D pada kelas AR3DModelContaner dijalankan. Begitu juga jika *marker* tidak terbaca sistem method *set* **visible** diisi nilai *false* yang berarti objek 3D disembunyikan.

3.2.3 *Rendering* objek 3D

Setelah *marker* ditemukan maka, objek 3D di-*render* atau dimunculkan diatas *marker*, posisi dan rotasi objek 3D akan mengikuti *tracker*. Method yang berkaitan dengan hal ini adalah **onEnterFrame**. Setiap objek dalam **AR3DModelContainer** akan diatur posisinya sesuai *marker* yang berkaitan dengan objeknya.

3.3 Implementasi Perancangan Aplikasi

Berdasarkan rancangan kelas dan *sequence diagram*, kelas-kelas yang diperlukan dibangun. Penulis menggunakan Adobe Flash Builder sebagai *Integrated Development Tools* (IDE) untuk mengembangkan aplikasinya. SDK yang digunakan adalah Flex versi 4.1. Dengan Adobe Flash Builder, proses *debugging* dan *deploy* aplikasi dapat mudah dilakukan.

Sesuai dengan rancangan pada subbab 3.2.1, berikut ini adalah penjelasan mengenai *source code* yang dibuat yang penulis bangun sesuai dengan rancangan.

3.3.1 Inisialisasi

ActionScript telah menyediakan kelas **Sprite** untuk menampilkan semua objek yang akan digambarkan (*drawing objects*). Kelas utama **AR3DV** dibangun dengan mewarisi kelas **Sprite** tersebut. Semua objek dan komponen lainnya ditambahkan ke kelas utama tersebut.

3.3.1.1 Inisialisasi Model 3D

Inisialisasi FLARManager terdapat pada method **initModels**, seperti ditunjukkan pada *listing* 3.1.

Listing 3.1: Inisialisasi model

```
private function initModels():void{
        // load model source dan konfigurasinya
        this.ar3dvContainer = new AR3DVModelContainer("../resources/ar3dv.
            xml");
        this.ar3dvContainer.addEventListener(AR3DVModelContainer.
            CONFIG_FILE_PARSED,this.onModelsLoaded);
}
```

Objek baru **AR3DVModelContainer** dibuat dan disimpan dalam variabel **ar3dvContainer**. **AR3DVModelContainer** memerlukan file konfigurasi (ar3dv.xml) yang berisi informasi lokasi file 3D, format file, rotasi xyz dan skala objek 3D (lampiran). Pertama kali dibentuk, objek **ar3dvContainer** akan meload file konfigurasi tersebut menggunakan **URLLoader**, seperti ditunjukkan *listing* 3.2.

Listing 3.2: Load Config AR3DV

```
public function AR3DVModelContainer(url:String){
        _configFileLoader = new URLLoader();
        _configFileLoader.addEventListener(IOErrorEvent.IO_ERROR, this.
            onConfigLoaded);
        _configFileLoader.addEventListener(SecurityErrorEvent.
            SECURITY_ERROR, this.onConfigLoaded);
        _configFileLoader.addEventListener(Event.COMPLETE, this.
            onConfigLoaded);
        _configFileLoader.load(new URLRequest(url));
}
```

Sebuah *listener* ditambahkan ke objek **configLoader**, sehingga setelah file selesai di-*load*, method **onConfigLoaded** dapat dijalankan dan file xml dapat diproses oleh method **parseConfigFile**, seperti yang ditunjukkan oleh listing 3.3.

Listing 3.3: Proses config file

```
private function parseConfigFile(data:XML):void{
        var modelList:XMLList = data.models;
        _containerByPatternId = new Array();
```

```
            _modelContainers = new Array();
            for each (var elem:XML in modelList.model) {
                    this.debug("load model :"+elem.@name);
                    var modelAR3DV:AR3DVModel = new AR3DVModel(elem.@name,elem
                        .@source_dir,elem.@source,int(elem.@pattern),elem.
                        @type);
                    if(modelAR3DV.model!=null){
                            modelAR3DV.addEventListener(AR3DVModelEvent.LOADED
                                ,this.onModelsLoaded);
                            modelAR3DV.setRotationXYZ(int(elem.@x),int(elem.@y
                                ),int(elem.@z));
                            modelAR3DV.scale = int(elem.@scale);
                            modelAR3DV.load();
                    }
                    _modelContainers[int(elem.@pattern)] = modelAR3DV;
            }
    }
```

Method **parseConfigFile** akan me-*load* objek 3D dengan bantuan objek **modelAR3DV** yang merupakan objek dari kelas AR3DVModel. Objek model disimpan dalam *array* dengan *key index*-nya berupa nomor pola *marker* yang ditentukan di file konfigurasi model (ar3dv.xml).

3.3.1.2 Inisialisasi FLARManager

Setelah model di-*load*, maka method **initAR** pun dijalankan yang merupakan inisialisasi FLARManager. Seperti ditunjukkan pada potongan *listing* method **initAR** 3.4.

Listing 3.4: Init FLARManager

```
            /* Augmented reality initialisation */
            private function initAR():void {
                    /* Initialise FLARManager */
                    this.fm =  new FLARManager("../resources/flar/flarConfig.
                        xml", new FLARToolkitManager(), this.stage);
```

FLARManager diinisialisasi dengan membuat objek baru **FLARManager** yang disimpan dalam variabel **fm**. FLARManager memerlukan file konfigurasi (flarConfig.xml) dan objek *tracker* (dalam hal ini FLARToolkitManager). Konfigurasi FLARManager menggunakan file xml (flarConfig.xml, isi file konfigurasi dapat dilihat secara lengkap pada lampiran), ada empat hal utama dalam konfigurasi ini, yaitu sebagai berikut:

1. video source
   Terdiri dari konfigurasi dimensi video yang di-*capture* **([sourceWidth] dan [sourceHeight])**, dimensi video yang ditampilkan **([displayWidth] dan [displayHeight])**, *framerate* dari video yang di-*capture*, dan jumlah *downsampling* (*scaling down*)yang di*capture* sebelum diproses.

2. FLARManager Tracker
   konfigurasi yang menyatakan video ditampilkan *mirrored* [mirrorDisplay], pengaturan pergerakan AR [smoothing], kelas yang digunakan untuk *smoothing* [smoother], dan akurasi dari deteksi *marker* terhadap perubahan cahaya (*thresholding*) [thresholdAdapter].

3. Parameter kamera
   Sebuah file telah disediakan oleh FLARManager,atau lebih tepatnya oleh FLARToolKit, untuk mengkompensasi distorsi kamera atau *webcam* yang digunakan [¡cameraParamsFile¿].

4. Pola AR
   lokasi dari file pola *marker* yang dapat dideteksi oleh FLARManager [¡pattern¿]. File tersebut berisi matriks yang bersesuaian dengan pola *marker*, pola *marker* secara lengkap dapat dilihat pada lampiran.

setelah FLARManager selesai diinisialisasi, Event **Event.INIT** di-*dispatch* atau di-*trigger*, sehingga method **onFlarManagerLoad** akan dijalankan. Pada method **onFlarManagerLoad**, *webcam* ditambahkan ke *Sprite*, kemudian method **initEngine3D** yang merupakan inisialisasi Papervision3D diproses.

3.3.2 *Tracking* Marker

listener untuk *method* yang akan dijalankan jika *marker* ditemukan yaitu **(onMarkerAdded)**, dan *listener* ketika *marker* tidak ditemukan lagi yaitu **(onMarkerRemoved)** ditambahkan ke method **initAR**, seperti ditunjukkan oleh *listing* 3.5.

Listing 3.5: Init AR

```
        /* Inisialisasi AR */
```

```
        private function initAR():void {
                /* Inisiliasasi FLARManager */
                this.fm =  new FLARManager("../resources/flar/flarConfig.
                    xml", new FLARToolkitManager(), this.stage);
                /* Event listener ketika sebuah marker dikenali */
                this.fm.addEventListener(FLARMarkerEvent.MARKER_ADDED,
                    this.onMarkerAdded);
                /* Event listener ketika sebuah marker tidak terdeteksi
                    lagi*/
                this.fm.addEventListener(FLARMarkerEvent.MARKER_REMOVED,
                    this.onMarkerRemoved);
                /* Event listener jika inisialisasi selesai */
                this.fm.addEventListener(Event.INIT, this.
                    onFlarManagerLoad);
                /* tampilkan webcam */
                this.addChild(Sprite(this.fm.flarSource));
        }
```

Listing 3.6: Marker

```
    private function onMarkerAdded (evt:FLARMarkerEvent) :void {
                var marker:FLARMarker = evt.marker;
                var patID:int = marker.patternId;
                if(this.ar3dvContainer.hasModel(patID)){
                        this.detectedMarkers[patID] = marker;
                        this.ar3dvContainer.setVisible(patID,true);
                }
        }

        private function onMarkerRemoved (evt:FLARMarkerEvent) :void {
                var marker:FLARMarker = evt.marker;
                var patID:int = marker.patternId;
                if(this.ar3dvContainer.hasModel(patID) && this.
                    detectedMarkers[patID] != null){
                        this.detectedMarkers[patID] = null;
                        this.ar3dvContainer.setVisible(patID,false);
                }
        }
```

ketika *marker* ditemukan maka objek 3D yang tersimpan di dalam objek ar3dvContainer di set *visibility* nya ke nilai **true** yang berarti objek 3D dimunculkan. Begitu juga sebaliknya, ketika *marker* tidak ditemukan lagi maka objek 3D yang tersimpan di dalam objek ar3dvContainer di set *visibility* nya ke nilai **false** yang berarti objek 3D tidak dimunculkan.

3.3.3 *Rendering* Objek 3D

Render objek 3D menggunakan *library* Papervision3D, yang telah diinisialisasi sebelumnya, ada dua jenis *rendering* dalam Papervision3D yaitu **LazyRenderEngine** dan **BasicRenderEngine**. Keduanya sama, hanya memiliki perbedaan dari cara pemakaian, pada **LazyRenderEngine** viewport, scene dan camera ditentukan terlebih dahulu sebelum objek di-*render*. Sedangkan pada **BasicRenderEngine**, ketiga aspek tersebut ditentukan ketika merender objek. Perbedaan keduanya dapat dilihat dengan jelas dari *listing* 3.7.

Listing 3.7: Rendering PV3D

```
var lazy:LazyRenderEngine = new LazyRenderEngine(scene, camera ,
    viewport);
lazy.render();

var basic:BasicRenderEngine = new BasicRenderEngine();
basic.renderScene(scene, camera, viewport);
```

Proses rendering berjalan terus menerus selama aplikasi dijalankan, method yang menangani ini adalah **onEnterFrame**. Method ini akan menentukan posisi dan rotasi dari objek 3D satu-persatu terhadap *marker* dengan menggunakan kelas transformasi matriks **PVGeomUtils**. Kemudian rendering Papervision3D diproses, seperti yang ditunjukkan *listing* 3.8.

Listing 3.8: Rendering AR

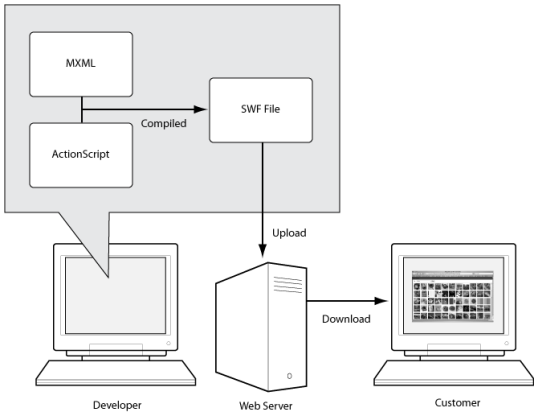
```
private function onEnterFrame (evt:Event) :void {
        var marker:FLARMarker;
        var transMatrix:Matrix3D;

        for(var i:String in this.detectedMarkers){
                marker = this.detectedMarkers[int(i)];
                if(marker!=null){
                        //konversi matriks ke matriks yang
                            bersesuaian dengan PV3D
                        transMatrix = PVGeomUtils.
                            convertMatrixToPVMatrix(marker.
                            transformMatrix);
                        this.ar3dvContainer.setTransform(int(i),
                            transMatrix);
                }
        }

        // render PV3D engine
```

```
                _renderEngine.render();
        }
```

Setelah semua *source code* dibangun, *file source code* aplikasi, FLARManager, FLARToolKit dan *file library* Papervision3D di kompilasi dalam satu *file* SWF. Agar dapat ditampilkan ke *web browser*, Flash Builder telah menyediakan HTML *wrapper* untuk menampilkan *file* swf yang telah dikompilasi (HTML wrapper dapat dilihat secara lengkap pada lampiran). Selanjutnya aplikasi ini di-*deploy* ke *web server* lokal agar mudah diakses, sekaligus sebagai simulasi kondisi sebenarnya karena aplikasi ini bisa diakses dari mana saja lewat koneksi internet. Kemudian pengguna mengakses *web server* yang terhubung ke internet dari komputer menggunakan *web browser* seperti yang ditunjukkan oleh gambar 3.6.



Gambar 3.6: Proses kompilasi, *deploy* dan akses aplikasi

Bab 4

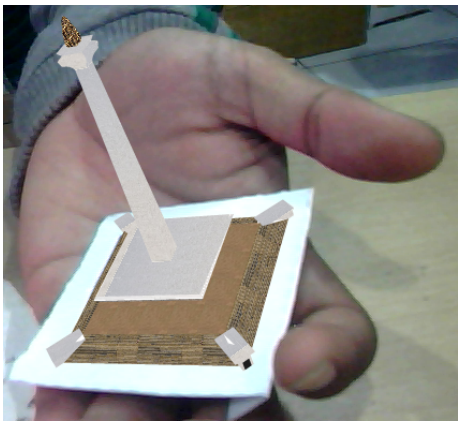
PENGUJIAN DAN ANALISA

Dalam pengujian aplikasi ini, ada dua cara pengujian yaitu dengan PC notebook yang dilengkapi *webcam* internal atau menggunakan PC *Desktop* dengan tambahan *webcam*, konfigurasi sistem dapat dilihat pada gambar 4.1. Untuk menjalankan aplikasi ini, diperlukan *web browser* yang sudah terintegrasi *plug-in* flash player versi 10.

Spesifikasi sistem yang penulis gunakan dalam pengujian ini adalah sebagai berikut.

- Prosesor AMD Phenom^{TM}II X4 955 Processor 3200 Mhz
- Memori 2 Gb
- standar usb 2.0 *webcam*
- *web browser* Mozilla Firefox versi 3.6.12

Hasil pengujian dapat dilihat pada gambar 4.1. Objek 3D dapat dirender dengan baik, meskipun kadang-kadang hilang dari marker, dikarenakan sistem tidak dapat menerima input dengan pergerakan yang marker cepat. Ada dua hal yang akan dianalisa yaitu fps (frame per second) dari video hasil rendering objek 3D dengan model yang berbeda-beda serta juga jarak dan sudut kemiringan marker yang masih dapat diterima aplikasi.



Gambar 4.1: Hasil pengujian

4.1 Analisa *Frame per Second (FPS)*

Frame rate atau frekuensi *frame* merupakan frekuensi sebuah alat atau layar menghasilkan gambar yang disebut *frame*. *Frame rate* lebih sering dikenal dan diekspresikan sebagai frame per second (FPS). FPS merupakan jumlah gambar yang dihasilkan dalam waktu satu detik, mata manusia memerlukan jumlah *frame* tertentu dalam satu detik agar dapat melihat gambar bergerak tanpa adanya *flicker*. Jumlah FPS juga tergantung dari intensitas cahaya dan kecepatan objek bergerak dari video yang dihasilkan.

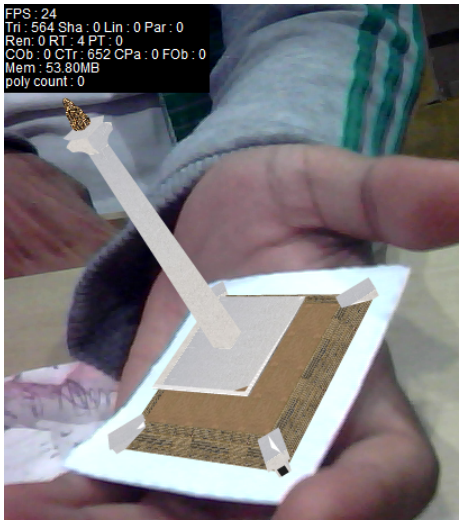
Pengujian untuk menganalisa FPS sangat diperlukan untuk mengetahui sejauh mana performa aplikasi viewer objek 3D ini. Untuk keperluan tersebut, penulis menambahkan informasi FPS tersebut ke aplikasi agar dapat dianalisa. Papervision3D telah menyediakan kelas untuk hal tersebut yaitu **StatsView**. Penggunaannya dapat dilihat dari *listing* 4.1.

Listing 4.1: StatsView

```
_renderEngine = new LazyRenderEngine(_scene3D, _camera3D,
    _viewport3D);
addChild(new StatsView(_renderEngine));
```

Sebuah informasi teks akan ditampilkan dikiri atas tampilan video, seperti yang ditunjukkan oleh gambar **??**. Penulis menggunakan lima macam objek 3D berbeda dalam melakukan pengujian ini. Hasil pengujian dapat dilihat

pada tabel 4.1.



Gambar 4.2: Hasil pengujian dengan statistik

Tabel 4.1: Hasil Pengujian dengan Lima Model 3D

No	Nama	Tipe file	Ukuran file (KB)	FPS	Memory (MB)	Tri
1	airport	3DS	129,7	14	51,76	2870
2	eiffel	DAE	38,7	50	44,93	206
3	monas	DAE	154,5	24	53.80	564
4	scout	DAE	267,2	38	55.28	233
5	tajmahal	DAE	812,9	0	567.68	147.018

Nilai FPS bergantung pada kompleksitas objek 3D yang ditampilkan. Untuk objek 3D yang tidak terlalu kompleks, hasilnya sangat baik, FPS berkisar antara 24 sampai 50, sehingga masih dapat terlihat baik. pada objek 3D kelima, nilai FPS nol atau tidak bergerak sama sekali, walaupun objek tersebut dapat terlihat. Penulis beranggapan bahwa frame yang dihasilkan dalam beberapa detik sekali, sehingga nilai FPS yang didapat adalah nol, hal ini disebabkan karena objek 3D yang di-*render* sangat kompleks sehingga aplikasi tidak sanggup untum menampilkannya dengan baik.

Bab 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pembahasan bab-bab sebelumnya dan didukung oleh hasil pengujian, dapat diambil kesimpulan sebagai berikut.

1. Aplikasi dapat berjalan dengan baik tanpa perlu menginstall aplikasinya, karena aplikasi diakses menggunakan browser yang mempunyai *plug-in* flash.
2. Aplikasi dapat merender objek 3D dengan format DAE dan 3DS sesuai dengan marker yang didteksi, akan tetapi tingkat pengenalan marker sangat dipengaruhi cahaya dan bentuk marker yang digunakan.
3. Aplikasi dapat merender objek 3D sederhana dengan baik dengan nilai FPS yang relatif baik (24 sampai 50).
4. Untuk objek 3D kompleks aplikasi tidak dapat berjalan dengan baik, terjadi *flicker* atau gambar tidak bergerak sama sekali.

5.2 Saran

1. Untuk pengembangan selanjutnya, diharapkan dapat menggunakan *library* yang lebih baik. Marker yang digunakan diharapkan lebih variatif,

dengan pola yang lebih menarik atau mungkin dengan pola yang berwarna sehingga lebih menarik.

2. Perangkat keras yang digunakan tidak lagi komputer tapi berupa *gadget* yang lebih kecil dan bisa menjalankan flash, sehingga aplikasi benar-benar dapat dijalankan dari mana saja.

Daftar Pustaka

[1] R. T. Azuma, "A survey of augmented reality," Presence: Teleoperators and Virtual Environments, vol. 6, no. 4, pp. 355–385, 1997.

[2] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays," IEICE Transactions on Information Systems, vol. E77-D, pp. 1321–1329, December 1994.

[3] O. Bimber and R. Raskar, Spatial Augmented Reality: Merging Real and Virtual Worlds. A K Peters, 2005.

[4] J. P. Rolland, R. L. Holloway, and H. Fuchs, "A comparison of optical and video see-through head-mounted displays," Proceedings of SPIE: Telemanipulator and Telepresence Technologies, p. 293307, 1994.

[5] M. Haller, M. Billinghurst, and B. H. Thomas, Emerging Technologies of Augmented Reality: Interfaces and Design, p. 51. Idea Group Publishing, June 2010.

[6] J. A. Jacko and A. Sears, Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises, p. 454. CRC Press, June 2010.

[7] R. Raskar, G. Welch, and H. Fuchs, "Spatially augmented reality," Presence: Teleoperators and Virtual Environments, Sept. 1998.

[8] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, "Augmented reality: A class of displays on the reality-virtuality continuum," SPIE Proceedings volume 2351: Telemanipulator and Telepresence Technologies, pp. 282–292, October 1994.

[9] P. Milgram, D. Drascic, J. J. Grodski, A. Restogi, S. Zhai, and C. Zhou, "Merging real and virtual worlds," in Proceedings of IMAGINA'95, feb 1995.

[10] W. Barfield and E. B. Nash, "Augmented reality," in International Encyclopedia of Ergonomics and Human Factors (W. Karwowski, ed.), vol. 1, pp. 1029–1032, Boca Raton: CRC Press, second ed., 2006.

[11] W. Barfield and T. A. Furness, Virtual environments and advanced interface design. Oxford University Press, 1995.

[12] O. Bimber, Gatesy, Witmer, R. Raskar, Encarnacao, and L. M. Encarnao, "Merging fossil specimens with computer-generated information," 2002.

[13] R. I. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision. Cambridge University Press, second ed., 2004.

[14] J. P. Rolland and H. Fuchs, "Optical versus video see-through head-mounted displays in medical visualization," in Presence: Teleoperators and Virtual Environments, pp. 287–309, 2000.

[15] H. Takemura and F. Kishino, "Cooperative work environment using virtual workspace," in Proc. Computer Supported Cooperative Work (CSCW'92), pp. 226–232, 1992.

[16] M. Sairio, "Augmented reality."

[17] A. B. Craig, W. R. Sherman, and J. D. Will, Developing Virtual Reality Applications Foundations of Effective Design.

Lampiran A

Listing Source Code Aplikasi

A.1 AR3DV.as

```
1  package ar3dv
2  {
3          /* FLARManager Framework [http://words.transmote.com/wp/
               flarmanager/] */
4          import com.transmote.flar.FLARManager;
5          import com.transmote.flar.camera.FLARCamera_PV3D;
6          import com.transmote.flar.marker.FLARMarker;
7          import com.transmote.flar.marker.FLARMarkerEvent;
8          import com.transmote.flar.tracker.FLARToolkitManager;
9          import com.transmote.flar.utils.geom.PVGeomUtils;
10
11         import flash.display.Shape;
12         import flash.display.Sprite;
13         import flash.events.ErrorEvent;
14         import flash.events.Event;
15         import flash.events.EventDispatcher;
16         import flash.events.IOErrorEvent;
17         import flash.events.SecurityErrorEvent;
18         import flash.geom.Point;
19         import flash.geom.Rectangle;
20         import flash.net.URLLoader;
21         import flash.net.URLRequest;
22         import flash.utils.Dictionary;
23
24         import org.libspark.flartoolkit.support.pv3d.FLARCamera3D;
25         import org.papervision3d.cameras.Camera3D;
26         import org.papervision3d.core.math.Matrix3D;
27         import org.papervision3d.lights.PointLight3D;
28         import org.papervision3d.materials.shadematerials.
               FlatShadeMaterial;
```

```
29          import org.papervision3d.materials.utils.MaterialsList;
30          import org.papervision3d.objects.DisplayObject3D;
31          import org.papervision3d.objects.parsers.DAE;
32          import org.papervision3d.objects.primitives.Cube;
33          import org.papervision3d.render.LazyRenderEngine;
34          import org.papervision3d.scenes.Scene3D;
35          import org.papervision3d.view.Viewport3D;
36          import org.papervision3d.view.stats.StatsView;
37
38          /* Setting output */
39          [SWF(width="640", height="480", frameRate="60", backgroundColor="
                #000000")]
40          public class AR3DV extends Sprite {
41                  /* FLARManager pointer */
42                  protected var fm:FLARManager;
43                  /* model config/source */
44                  protected var ar3dvContainer:AR3DVModelContainer;
45                  /* Array storing references to all markers on screen, key
                        by pattern id */
46                  private var detectedMarkers:Array;
47                  /* Papervision Scene3D pointer */
48                  private var _scene3D:Scene3D;
49                  /* Papervision Viewport3D pointer */
50                  private var _viewport3D:Viewport3D;
51                  /* FLARToolkit FLARCamera3D pointer */
52                  private var _camera3D:Camera3D;
53                  /* Papervision render engine pointer */
54                  private var _renderEngine:LazyRenderEngine;
55                  /* Papervision PointLight3D pointer */
56                  private var _pointLight3D:PointLight3D;
57
58                  public function AR3DV() {
59                          this.initModels();
60                  }
61
62                  private function debug(info:String):void{
63                          trace("[AR3DV] "+info);
64                  }
65
66                  private function onModelsLoaded(evt:Event):void{
67                          this.ar3dvContainer.removeEventListener(
                                AR3DVModelContainer.CONFIG_FILE_PARSED,this.
                                onModelsLoaded);
68                          this.debug("model selesai di-load");
69                          this.initAR();
70                  }
71
72                  private function initModels():void{
73                          // load model source dan konfigurasinya
74                          this.ar3dvContainer = new AR3DVModelContainer("../
                                resources/ar3dv.xml");
```

```
75                          this.ar3dvContainer.addEventListener(
                                AR3DVModelContainer.CONFIG_FILE_PARSED,this.
                                onModelsLoaded);
76                  }
77
78                  private function onFlarManagerLoad(e:Event):void {
79                          /* listener dihapus agar fungsi onFlarManagerLoad
                                tidak dijalankan lagi */
80                          this.fm.removeEventListener(Event.INIT, this.
                                onFlarManagerLoad);
81
82                          this.initEngine3D();
83                          /* event listener untuk setiap frame */
84                          this.stage.addEventListener(Event.ENTER_FRAME,
                                this.onEnterFrame);
85                  }
86
87                  /* Inisialisasi AR */
88                  private function initAR():void {
89                          /* Inisiliasasi FLARManager */
90                          this.fm =  new FLARManager("../resources/flar/
                                flarConfig.xml", new FLARToolkitManager(),
                                this.stage);
91                          /* Event listener ketika sebuah marker dikenali */
92                          this.fm.addEventListener(FLARMarkerEvent.
                                MARKER_ADDED, this.onMarkerAdded);
93                          /* Event listener ketika sebuah marker tidak
                                terdeteksi lagi*/
94                          this.fm.addEventListener(FLARMarkerEvent.
                                MARKER_REMOVED, this.onMarkerRemoved);
95                          /* Event listener jika inisialisasi selesai */
96                          this.fm.addEventListener(Event.INIT, this.
                                onFlarManagerLoad);
97                          /* tampilkan webcam */
98                          this.addChild(Sprite(this.fm.flarSource));
99                  }
100
101                 private function initEngine3D():void{
102                         _scene3D = new Scene3D();
103                         /* Init FLARCamera3D */
104                         _camera3D = new FLARCamera_PV3D(this.fm, new
                                Rectangle(0, 0, this.stage.stageWidth, this.
                                stage.stageHeight));
105
106                         /* Papervision viewport */
107                         _viewport3D = new Viewport3D(this.stage.stageWidth
                                , this.stage.stageHeight);
108                         /* Menambahkan Papervision viewport */
109                         this.addChild(_viewport3D);
110
111                         /* Papervision point light */
```

```
112                         _pointLight3D = new PointLight3D(true, false);
113                         /* light position */
114                         _pointLight3D.x = 1000;
115                         _pointLight3D.y = 1000;
116                         _pointLight3D.z = -1000;
117                         /* Menambahkan light ke Papervision scene */
118                         _scene3D.addChild(_pointLight3D);
119
120                         this.detectedMarkers = new Array();
121
122                         for each(var container:DisplayObject3D in this.
                                ar3dvContainer.containerByPatternId) {
123                                 _scene3D.addChild(container);
124                         }
125                         /* Papervision render engine Init */
126                         _renderEngine = new LazyRenderEngine(_scene3D,
                                _camera3D, _viewport3D);
127                         addChild(new StatsView(_renderEngine));
128                 }
129
130                 private function onMarkerAdded (evt:FLARMarkerEvent) :void
                        {
131                         var marker:FLARMarker = evt.marker;
132                         var patID:int = marker.patternId;
133                         this.debug("marker denga pola "+patID+" terdeteksi
                                ");
134                         if(this.ar3dvContainer.hasModel(patID)){
135                                 this.debug("model  :  "+patID+" ditemukan"
                                        );
136                                 this.detectedMarkers[patID] = marker;
137                                 this.ar3dvContainer.setVisible(patID,true)
                                        ;
138                         }else{
139                                 this.debug("model  :  "+patID+" tidak
                                        ditemukan");
140                         }
141                 }
142
143                 private function onMarkerRemoved (evt:FLARMarkerEvent) :
                        void {
144                         var marker:FLARMarker = evt.marker;
145                         var patID:int = marker.patternId;
146                         if(this.ar3dvContainer.hasModel(patID) && this.
                                detectedMarkers[patID] != null){
147                                 this.debug("marker denga pola "+patID+"
                                        tidak terdeteksi lagi");
148                                 this.detectedMarkers[patID] = null;
149                                 //this.ar3dvContainer.setVisible(patID,
                                        false);
150                         }
151                 }
```

```
152
153                 private function onEnterFrame (evt:Event) :void {
154                         var marker:FLARMarker;
155                         var transMatrix:Matrix3D;
156
157                         for(var i:String in this.detectedMarkers){
158                                 marker = this.detectedMarkers[int(i)];
159                                 if(marker!=null){
160                                         //konversi matriks ke matriks yang
                                                bersesuaian dengan PV3D
161                                         transMatrix = PVGeomUtils.
                                                convertMatrixToPVMatrix(marker
                                                .transformMatrix);
162                                         this.ar3dvContainer.setTransform(
                                                int(i),transMatrix);
163                                 }
164                         }
165
166                         // render PV3D engine
167                         _renderEngine.render();
168                 }
169
170         }
171 }
```

A.2 AR3DVModelContainer.as

```
1  package ar3dv
2  {
3          import flash.events.ErrorEvent;
4          import flash.events.Event;
5          import flash.events.EventDispatcher;
6          import flash.events.IOErrorEvent;
7          import flash.events.SecurityErrorEvent;
8          import flash.net.URLLoader;
9          import flash.net.URLRequest;
10         import flash.utils.Dictionary;
11
12         import org.papervision3d.core.math.Matrix3D;
13         import org.papervision3d.objects.DisplayObject3D;
14         import org.papervision3d.objects.parsers.DAE;
15         import org.papervision3d.objects.parsers.Max3DS;
16
17         public class AR3DVModelContainer extends EventDispatcher
18         {
19                 public static const CONFIG_FILE_LOADED:String = "
                       configFileLoaded";
20                 public static const CONFIG_FILE_PARSED:String = "
                       configFileParsed";
21
22                 private var _configFileLoader:URLLoader;
23                 private var _containerByPatternId:Array;
24                 private var _modelContainers:Array;
25                 private var _modelAR3DV:AR3DVModel;
26
27                 public function AR3DVModelContainer(url:String)
28                 {
29                         _configFileLoader = new URLLoader();
30                         _configFileLoader.addEventListener(IOErrorEvent.
                               IO_ERROR, this.onConfigLoaded);
31                         _configFileLoader.addEventListener(
                               SecurityErrorEvent.SECURITY_ERROR, this.
                               onConfigLoaded);
32                         _configFileLoader.addEventListener(Event.COMPLETE,
                                this.onConfigLoaded);
33                         _configFileLoader.load(new URLRequest(url));
34                 }
35
36                 private function debug(info:String):void{
37                         trace("[AR3DVModelContainer] "+info);
38                 }
39
40                 public function get containerByPatternId():Array{
41                         return _containerByPatternId;
42                 }
43
```

```
44                  public function getModelContainer(id:int):DisplayObject3D{
45                          if(this.hasModel(id)){
46                                  return _containerByPatternId[id];
47                          }else{
48                                  return null;
49                          }
50                  }
51
52                  public function setVisible(id:int,visibility:Boolean):void
                        {
53                          this.getModelContainer(id).visible = visibility;
54                  }
55
56                  public function setTransform(id:int,matrix:Matrix3D):void{
57                          this.getModelContainer(id).transform = matrix;
58                  }
59
60                  public function hasModel(id:int):Boolean{
61                          return _containerByPatternId[id]!=null;
62                  }
63
64                  private function onConfigLoaded (evt:Event) :void {
65                          _configFileLoader.removeEventListener(IOErrorEvent
                                .IO_ERROR, this.onConfigLoaded);
66                          _configFileLoader.removeEventListener(
                                SecurityErrorEvent.SECURITY_ERROR, this.
                                onConfigLoaded);
67                          _configFileLoader.removeEventListener(Event.
                                COMPLETE, this.onConfigLoaded);
68
69                          if (evt is ErrorEvent) {
70                                  var errorEvent:ErrorEvent = new ErrorEvent
                                        (ErrorEvent.ERROR);
71                                  errorEvent.text =  ErrorEvent(evt).text;
72                                  this.dispatchEvent(errorEvent);
73                                  return;
74                          }
75
76                          this.debug("proses file konfigurasi...");
77                          this.parseConfigFile(new XML(_configFileLoader.
                                data as String));
78                          _configFileLoader.close();
79                          _configFileLoader = null;
80                  }
81
82                  private function parseConfigFile(data:XML):void{
83                          var modelList:XMLList = data.models;
84                          _containerByPatternId = new Array();
85                          _modelContainers = new Array();
86                          for each (var elem:XML in modelList.model) {
87                                  this.debug("load model :"+elem.@name);
```

```
88                                  var modelAR3DV:AR3DVModel = new AR3DVModel
                                        (elem.@name,elem.@source_dir,elem.
                                        @source,int(elem.@pattern),elem.@type)
                                        ;
89                                  if(modelAR3DV.model!=null){
90                                          modelAR3DV.addEventListener(
                                                AR3DVModelEvent.LOADED,this.
                                                onModelsLoaded);
91                                          modelAR3DV.setRotationXYZ(int(elem
                                                .@x),int(elem.@y),int(elem.@z)
                                                );
92                                          modelAR3DV.scale = int(elem.@scale
                                                );
93                                          modelAR3DV.load();
94                                  }
95                                  _modelContainers[int(elem.@pattern)] =
                                        modelAR3DV;
96                          }
97                  }
98
99                  private function onModelsLoaded(evt:AR3DVModelEvent):void{
100                         var modelAR3DV:AR3DVModel = _modelContainers[evt.
                                patternId];
101                         var container:DisplayObject3D = new
                                DisplayObject3D();
102                         /*
103                         if(modelAR3DV.type == "DAE"){
104                                 container.addChild(DAE(modelAR3DV.model));
105                         }else{
106                                 container.addChild(Max3DS(modelAR3DV.model
                                        ));
107                         }
108                         */
109                         container.addChild(modelAR3DV.model);
110                         container.visible = false;
111                         _containerByPatternId[evt.patternId] = container;
112
113                         var parsed:Boolean = true;
114                         //cek apakah sudah semua model ter-load
115                         for(var i:String in _modelContainers){
116                                 parsed = parsed && _modelContainers[int(i)
                                        ].loaded;
117                         }
118
119                         if (parsed) {
120                                 this.debug(" PARSED");
121                                 _modelContainers[evt.patternId].
                                        removeEventListener(AR3DVModelEvent.
                                        LOADED,this.onModelsLoaded);
122                                 this.dispatchEvent(new Event(
                                        CONFIG_FILE_PARSED));
```

```
123                                 }
124                             }
125                     }
126     }
```

A.3 AR3DVModel.as

```
1  package ar3dv
2  {
3          /* Flash event package*/
4          import flash.events.ErrorEvent;
5          import flash.events.Event;
6          import flash.events.EventDispatcher;
7          import flash.events.IOErrorEvent;
8          import flash.events.SecurityErrorEvent;
9
10         import org.papervision3d.events.FileLoadEvent;
11         import org.papervision3d.objects.DisplayObject3D;
12         import org.papervision3d.objects.parsers.DAE;
13         import org.papervision3d.objects.parsers.Max3DS;
14
15         public class AR3DVModel extends EventDispatcher
16         {
17                  private var _loaded:Boolean;
18                  private var _source:String;
19                  private var _dir:String;
20                  private var _type:String;
21                  private var _model:DisplayObject3D;
22
23                  protected var _id:int;
24                  public function AR3DVModel(name:String,dir:String,source:
                        String,id:int,type:String="")
25                  {
26                          switch(type){
27                                  case "3DS":
28                                          _model = new Max3DS(name);
29                                          break;
30                                  case "DAE":
31                                          _model = new DAE(true,name,true);
32                                          break;
33                          }
34                          _type = type;
35                          _dir = dir;
36                          _source = source;
37                          _id = id;
38                          _loaded = false;
39                  }
40
41                  private function debug(info:String):void{
42                          trace("[AR3DVModel] "+info);
43                  }
44
45                  private function addListener():void{
46                          _model.addEventListener(FileLoadEvent.
                                LOAD_COMPLETE, this.onLoaded);
47                  }
```

```
48
49                  private function removeListener():void{
50                          _model.removeEventListener(FileLoadEvent.
                                LOAD_COMPLETE, this.onLoaded);
51                  }
52
53                  private function loadSource():void{
54                          switch(type){
55                                  case "3DS":
56                                          Max3DS(this.model).load(_dir+
                                                _source,null,_dir);
57                                          break;
58                                  case "DAE":
59                                          DAE(this.model).load(_dir+_source)
                                                ;
60                          }
61                  }
62
63                  public function get type():String{
64                          return _type;
65                  }
66
67                  public function get loaded():Boolean{
68                          return _loaded;
69                  }
70
71                  public function get model():DisplayObject3D{
72                          return _model;
73                  }
74
75                  public function load():void{
76                          if(_loaded) {
77                                  this.debug(_id+" loaded");
78                                  _loaded = true;
79                                  dispatchEvent(new AR3DVModelEvent(
                                        AR3DVModelEvent.LOADED,_id));
80                                  return;
81                          }
82                          this.addListener();
83                          this.loadSource();
84                  }
85
86                  public function setRotationXYZ(x:int,y:int,z:int):void{
87                          _model.rotationX = x;
88                          _model.rotationY = y;
89                          _model.rotationZ = z;
90                  }
91
92                  public function set scale(scale:int):void{
93                          _model.scale = scale;
94                  }
```

```
95
96                  private function onLoaded (evt:Event) :void {
97                          this.debug(_id+" loaded");
98                          _loaded = true;
99
100                         this.removeListener();
101                         this.dispatchEvent(new AR3DVModelEvent(
                                AR3DVModelEvent.LOADED,_id));
102                 }
103         }
104 }
```

A.4 AR3DVModelEvent.as

```
1  package ar3dv
2  {
3          import flash.events.Event;
4          public class AR3DVModelEvent extends Event {
5
6                  public static const LOADED:String = "loaded";
7
8                  private var _patternId:int = 0;
9
10                 public function get patternId():int {
11                         return _patternId;
12                 }
13
14                 public function AR3DVModelEvent(type:String, id:int){
15                         super(type, true);
16                         _patternId = id;
17                 }
18
19                 public override function clone():Event {
20                         return new AR3DVModelEvent(type, _patternId);
21                 }
22         }
23 }
```