ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – KI091391

Permainan Menembak Dengan Perintah Melalui Gerakan Tangan dan Suara.

HAQQI ANNAZIL
NRP 5108 100 605

Dosen Pembimbing
Imam Kuswardayan, S.Kom., M.T.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2011

FINAL PROJECT – KI091391

SHOOTING GAME USING HAND MOTION AND SPEECH CONTROL

HAQQI ANNAZIL
NRP 5108 100 605

Advisor
Imam Kuswardayan, S.Kom., M.T.

DEPARTMENT OF INFORMATICS
Faculty Of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2011

LEMBAR PENGESAHAN

LEMBAR PENGESAHAN

PERMAINAN MENEMBAK DENGAN PERINTAH MELALUI GERAKAN TANGAN DAN SUARA

TUGAS AKHIR
Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Teknik Informatika
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
HAQQI ANNAZIL
NRP. 5108 100 605

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Imam Kuswardayan, S.kom, M.T. .....
NIP: **197612152 003121 001** (pembimbing 1)

SURABAYA
JANUARI, 2011

PERMAINAN MENEMBAK DENGAN PERINTAH MELALUI GERAKAN TANGAN DAN SUARA

Nama Mahasiswa : **Haqqi Annazil**
NRP : **5108 100 605**
Jurusan : **Teknik Informatika FTIF-ITS**
Dosen Pembimbing I : **Imam Kuswandayan, S.Kom., M.T.**

Abstrak

Seiring perkembangan teknologi permainan yang sangat cepat diikuti dengan meningkatnya teknologi dan berkembangnya *hardware* untuk mempercepat proses kecepatan visual yang sangat tinggi dapat membuat permainan jauh lebih menarik. Dan sebagian permainan yang ada masih menggunakan *joystick, keyboard* dan *mouse* sebagai kontrol pada permainan. Namun diantaranya ada beberapa permainan dan *console* yang cukup menarik dengan melibatkan lebih banyak gerakan pemain dalam permainan, sehingga mampu lebih interaktif dan nyata.

Pada tugas akhir ini penulis mengusulkan untuk membuat permainan interaktif yang memanfaatkan gerakan tangan dan suara pemain sebagai kontrol permainan yaitu permainan menembak. Pemainan menembak ini akan terus diperbarui dengan menambahkan fitur-fitur agar lebih menarik. Kontrol permainan dilakukan dengan mengenali gerakan tangan menggenggam menggunakan fungsi *training haar cascade* pada *library opencv*, sedangkan kontrol suara dilakukan dengan pengenalan ucapan menggunakan fungsi *speech to text.* Permainan ini merupakan perkembangan dari pengolahan citra dan pengolahan suara menggunakan library API windows.

Kata Kunci : Permainan, Speech API, OpenCV, Haar Cascade.

(halaman ini sengaja dikosongkan)

SHOOTING GAME USING HAND MOTION AND SPEECH CONTROL

Student Name : **Haqqi Annazil**
Student ID : **5108 100 605**
Department : **Informatics FTIF-ITS**
Main Supervisor : **Imam Kuswandayan, S.Kom., MT.**

1 Abstract

Along with the rapid improvement of game technology followed by the increase in graphics technology and the development of hardware to accelerate the process of visual speed is very high to make the game becomes much more interactive. While most games are still using a joystick, keyboard and mouse as a control on the game, there are some games and the console which is quite attractive by engaged the movement of the players in the game. Therefore, the way of using this movement preserve to be more interactive and real among others.

The thesis proposes to create an interactive game that uses hand movements and speech as a control to play the shooting games as in case. This Shooting Game will continue to be updated by adding features to make it more attractive. Control game made by detecting the movement of the hand grasping using haar cascade training function in opencv libraries, while the speech control is done by using Pengenalan Suara speech to the text function. This shooting game is the development of image processing and sound processing using the API library windows.

Keyword: : Game, Speech API, OpenCV, Haar Cascade.

(*halaman ini sengaja dikosongkan*)

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas segala limpahan nikmat dan hidayah – Nya sehingga penulis dapat menyelesaikan tugas akhir ini yang berjudul :

"PERMAINAN MENEMBAK DENGAN PERINTAH MELALUI GERAKAN TANGAN DAN SUARA"

Pada kesempatan ini penulis ingin menyampaikan penghormatan dan terimakasih yang sebesar – besarnya kepada :

1. Bapak dan Ibu penulis yang selalu memberikan bimbingan , dukungan, doa, kasih sayang dan kata - kata bijak yang tak henti – hentinya menaungiku .
2. Adikku Rizky Amalia, terima kasih atas motivasi ,dan saran nya
3. Imam Kuswardayan, S.Kom, M.T, selaku dosen pembimbing yang telah mengarahkan penulis dalam penyelesaian tugas akhir ini.
4. Bapak Yudhi Purwananto, M.Kom selaku Ketua Jurusan Teknik Informatika beserta. seluruh dosen di jurusan Teknik Informatika ITS yang telah memberikan banyak ilmu, arahan, dan pengelamannya kepada penulis selama menempuh masa perkuliahan.
5. Bapak-bapak/Ibu-ibu dosen Jurusan Teknik Informatika FTIF-ITS, tempat di mana penulis belajar dan menimba ilmu. Bekal ilmu yang telah diberikan kepada penulis tidak dapat diukur nilainya, semoga di masa yang akan datang bisa membawa manfaat bagi semuanya.
6. Seluruh Staf dan Karyawan FTIF ITS yang banyak memberikan kelancaran administrasi akademik kepada penulis.
7. Teman – teman ALJ 2009 : Dimas F Putranto, R. Agung H, Cahya Purnama, Yohanes Kosasi, Dinna Mauiza, mas Nico,

mbak Umi C, mbak Ratri, mas Didit, pak Radit, terima kasih atas kebersamaannya.

8. Luthfi R N yang sudah perhatian sekali.
9. Yusuf "The Partners" terima kasih banyak bah...
10. Semua pihak yang telah banyak membantu dan tidak dapat disebutkan satu per satu..

Tiada untaian kata yang cukup yang untuk penulis sampaikan sebagai ucapan terimakasih hanya harapan semoga Allah SWT membalas semua amal kebaikan tersebut.

Semoga tugas akhir ini dapat bermanfaat bagi pembaca dan dapat digunakan sebagai referensi pengembangan perangkat lunak yang lebih baik. Penulis menyadari bahwa tugas akhir ini jauh kesempurnaan. Penulis mengucapkan maaf apabila terdapat kesalahan didalamnya. Kritik dan saran yang membangun sangat diharapkan sebagai perbaikan selanjutnya.

Surabaya, Januari 2011

Haqqi Annazil

DAFTAR ISI

DAFTAR ISI

DAFTAR GAMBAR

(halaman ini sengaja dikosongkan)

DAFTAR TABEL

(*halaman ini sengaja dikosongkan*)

BAB 1
PENDAHULUAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Seiring perkembangan permainan yang akhir-akhir ini sangat cepat dan diikuti dengan meningkatnya teknologi grafika serta berkembangnya *hardware* untuk mempercepat proses kecepatan *visual* yang sangat tinggi dapat membuat permainan jauh lebih interaktif. Hingga saat ini para pecinta permainan masih menggunakan beberapa permainan interaktif yang ada seperti playstation, xbox, java game di hp dan WII-nintendo. Dan sebagian permainan yang ada masih menggunakan *joystick, keyboard* dan lain lain. Namun diantaranya ada beberapa permainan yang cukup menarik sehingga mampu berinteraksi secara nyata dengan pemain. Diantaranya yaitu *Just Dance2*, *Call of Duty : Black Ops* milik WII-nintendo dan *PlayStation Eye* yang oleh PlayStation digunakan untuk melacak gerakan tubuh pemain yang berada dalam tampilan kamera.

Pada tugas akhir ini penulis mengusulkan untuk membuat permainan menembak. Permainan menembak ini akan terus diperbarui dengan menambahkan fitur-fitur agar lebih menarik sehingga dikembangkan permainan yang memanfaatkan gerakan tangan dan suara pemain sebagai kontrol permainan. Kontrol permainan dilakukan dengan mengenali gerakan tangan menggenggam menggunakan fungsi *training haar cascade* pada *library opencv*, sedangkan kontrol suara dilakukan dengan pengenalan suara menggunakan fungsi *speech to text*. Permainan ini merupakan perkembangan dari pengolahan citra dan pengolahan suara menggunakan *library API windows*.

1.2 Perumusan Masalah

Permasalahan yang diangkat dalam perancangan dan pembuatan implementasi tugas akhir ini meliputi :

- Bagaimana mengaplikasikan aturan-aturan permainan agar dapat dilaksanakan dengan baik.
- Bagaimana membedakan antara bentuk tangan yang sedang menggenggam dengan bentuk tangan yang lain.
- Bagaimana mengenali arah gerakan tangan pemain dan suara pemain dengan cepat sehingga diperoleh kondisi real time.

Berikut ini adalah beberapa batasan masalah dan ruang lingkup dalam pengerjaan tugas akhir ini :

- Kata yang diucapkan terbatas pada data yang tersedia.
- Bentuk tangan dan suara pemain harus pada kondisi yang ideal.
- Kondisi pencahayaan ruangan pada kondisi ideal/ terang.

1.3 Tujuan dan Manfaat

Tujuan dari pembuatan tugas akhir ini adalah untuk membuat permainan yang dapat dikontrol menggunakan gerakan tangan dan suara pemain.

1.4 Metodologi

Metodologi yang dilakukan dalam perancangan dan pembuatan aplikasi tugas akhir ini menggunakan langkah-langkah sebagai berikut :

1. Studi Pustaka

Tahap ini merupakan langkah awal dalam perancangan dan penyusunan tugas akhir ini yaitu mengumpulkan berbagai informasi untuk membangun perangkat lunak berupa buku–buku referensi, paper maupun literatur dokumentasi internet. Proyek akhir ini terinspirasi oleh permainan yang telah ada seperti :

- *Nintendo WII*[1].
  Produk nintendo yang dapat dimainkan dengan menggunakan sensor pada benda yang digerakkan pemain. Seperti terlihat pada gambar 1.1 dibawah ini.

Gambar 1.1 WII Bermain menggunakan Joystik.

- *CamGame*[2].
  Bermain menggunakan webcam. Seperti terlihat pada gambar 1.2 dibawah ini.

Gambar 1.2 Game Menggunakan Webcam

Serta mengumpulkan informasi untuk melakukan beberapa proses yang mendukung permainan antara lain :

a. Mempelajari *Library Opencv* beserta teknologi yang berkaitan.
b. Mempelajari *Windows Speech API.*
c. Mempelajari *Training Haar Cascade.*

2. Perancangan Perangkat Lunak

Pada tahapan ini mempunyai urutan dalam perancangan perangkat lunak, yaitu :

a. Mengenali bentuk tangan menggenggam
b. Mengenali perintah suara pemain
c. Membuat perancangan antar muka(*interface*) permainan.

3. Pembuatan Perangkat Lunak

Tahap ini merupakan proses dimulainya pembuatan perangkat lunak yang berdasarkan hasil perancangan sebelumnya. Dalam pembuatan perangkat lunak dibutuhkan beberapa *tool* dan *library* untuk mendukung aplikasi ini, antara lain

a. Microsoft Visual Studio .NET 2008
b. Library Opencv
c. SDK windows API(Speech to text).

4. Pengujian dan Evaluasi Perangkat Lunak

Tahap ini akan dilakukan uji coba sekaligus evaluasi perangkat lunak. Ujicoba dan evaluasi perangkat dilakukan untuk mencari masalah yang mungkin timbul, dan mengevaluasi jalannya program dan mengadakan proses perbaikan.

5. Penyusunan Laporan Tugas Akhir
   Pada tahap ini dilakukan penyusunan buku tugas akhir sebagai laporan dan dokumentasi mulai dari tahap awal hingga tahap akhir pengerjaan tugas akhir.

1.5 Sistematika Penulisan

Penulisan buku tugas akhir ini akan dibahas secara sistematis yang dibagi beberapa bab, yaitu

1. BAB 1, Pendahuluan, berisi latar belakang, perumusan masalah, tujuan dan manfaat, metodologi, dan sistematika penyusunan tugas akhir.
2. BAB 2, Dasar Teori, akan dibahas dasar ilmu yang mendukung pembahasan tugas akhir ini yaitu *Action Game dan Design Game, Library Opencv, Training Haar Cascade dan Speech Application Programming Interface(SAPI).*
3. BAB 3, Metodologi, akan membahas tentang analisis kebutuhan perancangan perangkat lunak yang akan dibuat meliputi : arsitektur, perancangan antarmuka perangkat lunak, dan perancangan desain interface
4. BAB 4, Pembuatan dan Implementasi, menjelaskan tentang penerapan fungsi dan aturan permainan dalam suatu bentuk aplikasi yang disertai *use case, flowchart,* dan *diagram algoritma* dari aplikasi permainan ini.
5. BAB 5, Uji Coba dan Implementasi, berisi tujuan pengujian yang merupakan jawaban permasalahan dalam permainan ini. Yang meliputi lingkungan uji coba, scenario ujicoba, dan hasil uji coba.
6. BAB 6, Penutup, berisi kesimpulan yang merupakan jawaban dari objektif atau tujuan dari pembuatan tugas akhir ini, dan saran-saran yang dibutuhkan untuk pengembangan permainan selanjutnya.
7. Daftar Pustaka dan Lampiran, berisi daftar pustaka yang dijadikan sebagai literature dalam tugas akhir ini, dan lampiran yang berisi data pendukung tugas akhir ini.

(halaman ini sengaja dikosongkan)

BAB 2
DASAR TEORI

BAB 2
DASAR TEORI

Pada bab ini akan dibahas mengenai dasar-dasar teori yang akan digunakan sebagai acuan dalam pengerjaan tugas akhir ini.

2. 1 Game

Sebuah *Game*[3] adalah kegiatan terstruktur, biasanya dilakukan untuk kesenangan dan terkadang dilakukan sebagai alat pendidikan. Permainan berbeda dari pekerjaan, yang biasanya dilakukan untuk *rumerisasi*, dari beberapa seni, yang lebih peduli dengan ekspresi ide. Namun, perbedaan tersebut tidak jelas, dan banyak permainan juga dianggap bekerja atau seni. Komponen kunci dari permainan adalah tujuan, aturan, tantangan dan interaksi. Permainan biasanya melibatkan stimulasi mental atau fisik, dan terkadang kedua-duanya. Banyak permainan membantu mengembangkan keterampilan praktis, berfungsi sebagai bentuk latihan, atau melakukan peran pendidikan, simulational atau psikologis.

Menurut perancang permainan komputer (crawford, 2003)[4], definisi istilah permainan menggunakan serangkaian dikotomi:

1) Ekspresi kreatif adalah seni jika dibuat untuk kepuasan sendiri, dan hiburan jika dibuat untuk uang.
2) Hiburan adalah sebuah mainan yang jika dapat melakukan interaksi. Film dan buku-buku dianggap sebagai contoh hiburan non-interaktif.
3) Jika tidak ada tujuan yang terkait dengan permainan, itu adalah mainan(*toy*). Jika memiliki tujuan, permainan adalah tantangan.
4) Jika tantangan tidak memiliki "*Active agent against whom you compete*" maka disebut teka-teki. Jika ada, itu adalah konflik.

5) Terakhir, jika para pemain hanya dapat mengalahkan lawan tetapi tidak menyerang mereka mengganggu kinerja mereka, konflik adalah sebuah kompetisi. Namun, jika serangan diizinkan kemudian konflik memenuhi syarat sebagai sebuah permainan.

Permainan dapat dicirikan oleh "apa yang pemain dapat lakukan". Hal ini sering disebut sebagai *gameplay*. Elemen kunci umum diidentifikasi dalam konteks ini adalah alat dan aturan yang mengidentifikasikan konteks keseluruhan permainan dan yang pada gilirannya menghasilkan keterampilan, strategi dan kesempatan.

Permainan sering diklasifikasikan oleh komponen yang diperlukan untuk memainkannya (misalnya miniatur, bola, *keyboard and pieces,* atau komputer). Permainan seperti petak umpet tidak memanfaatkan alat apapun, melainkan interaktivitas mereka ditentukan lingkungan.

Selain yang sering dikarakteristikan dengan alat-alat yang digunakan, permainan sering ditentukan oleh aturan yang digunakan. Ketika aturan yang dikenakan variasi dan perubahan, biasanya menghasilkan permainan baru. Aturan umumnya menentukan urutan gilirannya, hak-hak dan tanggungjawab pemain, dan tujuan masing-masing pemain.

2. 2 Action Game dan Design Game

Definisi *Action game*[5] adalah aliran permainan video yang menekankan tantangan fisik, termasuk koordinasi tangan-mata dan reaksi-waktu. Dan dalam sebuah permainan terdapat sebuah aliran yang mencakup beragam tipe seperti permainan pertempuran, permainan menembak, dan permainan *platform*. Dalam sebuah permainan aksi, pemain biasanya mengontrol sebuah *avatar protagonis*. *Avatar* harus menavigasi tingkat permainan, mengumpulkan benda-benda, menghindari rintangan,

dan memerangi musuh dengan berbagai serangan. Pada akhir tingkat atau kelompok tingkat, pemain sering harus mengalahkan musuh bos besar yang lebih besar dan lebih menantang daripada musuh-musuh yang lainnya. Serangan musuh dan rintangan menguras tenaga *avatar,* dan permainan akan berakhir ketika *avatar* telah kehabisan tenaga.

Definisi *Design Game*[6], adalah proses merancang isi dan aturan permainan dalam tahap pra-produksi. Dalam mendesain sebuah permainan membutuhkan beberapa komponen serta dokumentasi yang berisi konsep, daftar fitur, aturan, cerita dan target. Tahap pertama yang biasanya dimulai dengan ide, dan sering dimodifikasi dari konsep yang telah ada. Para *designer* sering kali bereksperimen dengan campuran beberapa aliran permainan. Perubahan ini sering kali berdampak positif pada permainan itu sendiri.

Menurut (Brenda Brathwaite; Ian Schreiber, 2009)[10] dalam mendesain permainan terdapat beberapa tipe, yaitu :

- *World Design* : Menciptakan sebuah *backstory,* pengaturan, dan tema untuk permainan yang sering dilakukan oleh seorang *lead desainer.*
- *System Design* : Aturan permainan dan pola matematis yang mendasari.
- *Content Design* : Penciptaan karakter, item, teka-teki dan misi.
- *Game Writting* : Penulisan permainan melibatkan menulis dialog, teks dan cerita.
- *Level Design* : Tingkatan permaianan dan fitur-fiturnya.
- *User Interface Design* : Membangun interaksi dengan pengguna/pemain.

2. 3 Library Opencv

Pengertian *Opencv*[8] adalah suatu library gratis yang dikembangkan oleh developer-developer Intel Corporation. Library ini terdiri dari fungsi-fungsi *computer vision* dan *image processing* tingkat tinggi. *OpenCV* sangat disarankan untuk *programmer* yang akan berkutat pada bidang *computer vision*, karena library ini mampu menciptkan aplikasi yang handal dan kuat dibidang *digital vision*.

Salah satu tujuan *OpenCV* untuk menyediakan penggunaan yang mudah infrastruktur *computer vision* yang dapat membantu orang dengan fungsi yang canggih pada aplikasi vision secara cepat. *Library OpenCV* memiliki lebih dari 500 fungsi yang dapat menjangkau banyak area dari didalam *computer vision*, termasuk Pabrik inspeksi produk, pencitraan medis, keamanan, *user interface*, kalibrasi kamera, *stereo vision*, *robotika*.

Saat ini para developer dari Intel Corporation telah membuat berbagai macam versi, yaitu:

- openCV untuk bahasa pemograman C/C++,
- openCV untuk bahasa pemograman C# (masih dalam tahap pengembangan), dan
- openCV untuk bahasa pemograman Java.

untuk bahasa pemograman C# dan Java, karena masih dalam tahap pengembangan, maka kita membutuhkan library lain sebagai pelengkap kekurangan yang ada. Namun untuk bahasa pemograman C/C++ tidak memerlukan library lainnya untuk pemrosesan pada *computer vision*.

2.3.1 Struktur Opencv

Struktur *OpenCV* menggunakan struktur IplImage untuk menciptakan dan menangani gambar. Maka dari itu, semua hal yang berhubungan dengan gambar/image harus ditampung pada variable tersebut. Kelebihan dari variable IplImage adalah dapat menampung semua format gambar, baik itu bitmap, png ataupun

jpeg, sehingga tidak membutuhkan konversi dari suatu format ke format yang lainnya. Sebagai contoh :

- .width adalah lebar dari Gambar.
- .height adalah tinggi dari /Gambar.
- .nChannels jumlah channel (1 jika gray level image dan 3 jika color level image).

Setiap fungsi yang ada pada library OpenCV menggunakan cv pada awal permulaan nama fungsinya. misalnya, cvSeq, cvCreateImage, cvNamedWindow, dan lain sebagainya. Library Opencv dibagi ke dalam 3 bagian, yaitu :

- CV, Didalamnya terdapat fungsi-fungsi yang berkaitan dengan *computer vision*, misalnya *histogram, template matching, optical flow*, dan sebagainya.
- CXCORE, Pada bagian ini terdapat fungsi-fungsi yang menangani hal-hal yang berkaitan dengan penulisan dan pembacaan data dari suatu file, penanganan tentang penggunaan memori dan lain sebagainya.
- HighGUI, Merupakan kumpulan fungsi-fungsi yang berkaitan dengan *windowing*, loading file video, dan lain sebagainya yang berkaitan dengan *Graphical User Interface(*GUI).

2.3.2 Fitur Pada Library Opencv

Adapun yang menjadi fitur-fitur yang sudah tersedia pada library ini, antara lain adalah sebagai berikut:

- manipulasi data gambar, contohnya alokasi memory penyimpanan, menyalin Gambar, *setting*, konversi dari Gambar satu ke Gambar lainnya, dan lain-lain yang berhubungan dengan Gambar/image.
- Gambar dan video input/output, contohnya untuk pengaksesan kamera, media membuka file video, dan lain sebagainya.

- Matriks dan perhitungan nilai vector dan juga aljabar linear, contohnya *dot* dan *cross products, eigenvalues, SVD.*
- Dasar-dasar *image processing*, misalnya *filtering*, deteksi tepi, deteksi sudut, *template matching, hough transform*, dan lain sebagainya.
- Camera calibration, misalnya *stereo correspondence*.
- *Motion analysis*, contohnya *optical flow, motion segmentation* dan *tracking*.
- Pengenalan obyek, contohnya *eigen-methods*.
- GUI dasar, contohnya menampilkan Gambar/video

2.4 Pengenalan Obyek

Metode yang digunakan untuk melakukan pengenalan obyek pada proyek akhir ini adalah *Haar Classifier*[7], yaitu metode yang membangun sebuah *boosted rejection cascade*, yang akan membuang data training negatif sehingga didapat suatu keputusan untuk menentukan data positif.

Suatu *Haar Classifier* merupakan metode *supervised learning*, yaitu membutuhkan data training untuk dapat mengenali obyek-obyek tertentu. Untuk itu, *Haar Classifier* membutuhkan data positif (obyek yang akan dikenali) dan data negatif (bukan obyek yang akan dikenali).

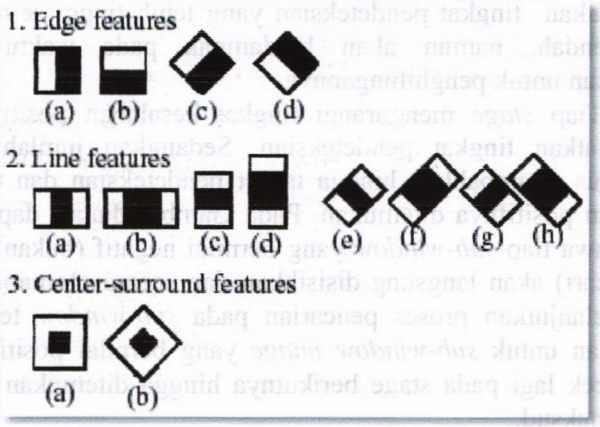
2.4.1 Fitur Haar

Setiap fitur haar terdiri dari gabungan kotak-kotak hitam dan putih. Tiga tipe kotak(*rectangular*) fitur adalah

- *Tipe two-rectangle feature* .
- *Tipe three-rectangle feature.*
- *Tipe four-rectangle feature.*

Nilai *haar-like feature* adalah perbedaan antara jumlah nilai-nilai piksel gray level dalam daerah kotak hitam dan daerah kotak putih. Kotak *haar-like feature* dapat dihitung secara cepat

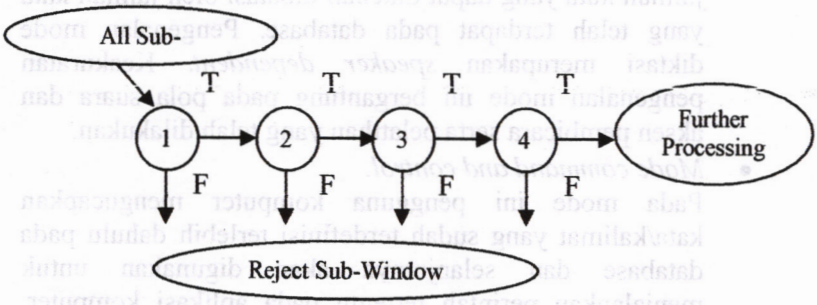
menggunakan "*integral image*". Seperti terlihat pada gambar 2.1 dibawah ini:



Gambar 2.1 Fitur Haar.

2.4.2 Cascade Classifier

Definisi dari cascade classifier adalah sebuah rantai *stage classifier*, dimana setiap *stage classifier* digunakan untuk mengenali apakah di dalam *image sub window* terdapat obyek yang diinginkan (*object of interest*). Seperti terlihat pada gambar 2.2 dibawah ini:



Gambar 2.2 Cascade Classifier.

Untuk mendapatkan nilai cascade, perlu dilakukan *training cascade*. *Classifier* dengan banyak fitur akan mendapatkan tingkat pendeteksian yang lebih tinggi serta error yang rendah, namun akan berdampak pada waktu yang dibutuhkan untuk penghitungannya.

Tiap *stage* mengurangi tingkat kesalahan positif serta meningkatkan tingkat pendeteksian. Sedangkan jumlah *stage* akan terus ditambahkan hingga target pendeteksian dan tingkat kesalahan positifnya ditemukan. Pada Gambar diatas, dapat kita lihat bahwa tiap *sub-window* yang bernilai negatif (bukan obyek yang dicari) akan langsung disisihkan dan secara otomatis tidak akan melanjutkan proses pencarian pada *sub-window* tersebut. Sedangkan untuk *sub-window image* yang bernilai positif akan terus dicek lagi pada stage berikutnya hingga ditemukan obyek yang dimaksud.

2.5 Pengenalan Suara

Terdapat dua macam mode pada sistem pengenalan pembicaraan yaitu:

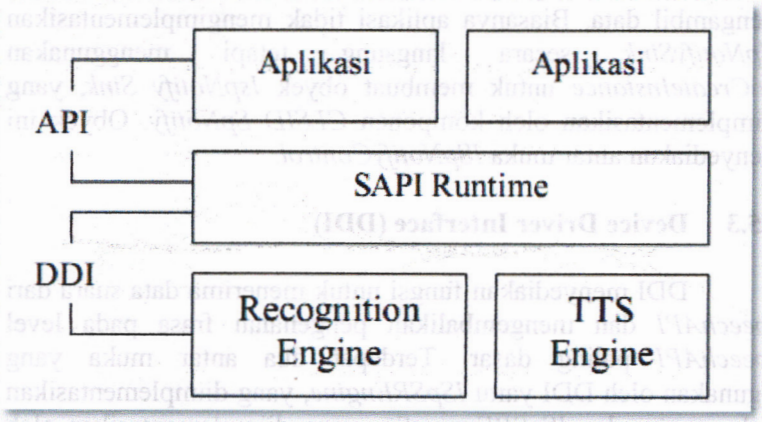
- *Mode diktasi.*
  Pada mode ini pengguna komputer dapat mengucapkan kata/kalimat yang selanjutnya akan dikenali oleh komputer dan diubah menjadi data teks. Kemungkinan jumlah kata yang dapat dikenali dibatasi oleh jumlah kata yang telah terdapat pada database. Pengenalan mode diktasi merupakan *speaker dependent.* Keakuratan pengenalan mode ini bergantung pada pola suara dan aksen pembicara serta pelatihan yang telah dilakukan.
- *Mode command and control.*
  Pada mode ini pengguna komputer mengucapkan kata/kalimat yang sudah terdefinisi terlebih dahulu pada database dan selanjutnya akan digunakan untuk menjalankan perintah tertentu pada aplikasi komputer. Jumlah perintah yang dapat dikenali tergantung dari

aplikasi yang telah mendefinisikan terlebih dahulu pada database jenis-jenis perintah yang dapat dieksekusikan. Mode ini merupakan *speaker independent* karena jumlah kata yang dikenali biasanya terbatas sekali.

2.5.1 Speech Application Programming Interface (SAPI)

Definisi *SpeechAPI*[9] adalah sebuah *API* yang dikembangkan oleh Microsoft yang digunakan sebagai pengenal suara di dalam pemrograman aplikasi Windows. Secara arsitektur, pemrograman SAPI dapat dilihat sebagai sebuah *middleware* yang terletak di antara aplikasi dan *speech engine*.

Sebuah *SpeechAPI* terdiri dari 2 antar muka yaitu *application programming interface*(API) dan *device driver interface*(DDI). Seperti terlihat pada gambar 2.3 dibawah ini:



Gambar 2.3 Blok Diagram Arsitektur SAPI.

Komponen utama di dalam SAPI ada tujuh yaitu

- *Voice Command.*
- *Voice Dictation.*
- *Voice Talk.*
- *Voice Telephony.*

- *Direct Pengenalan Suara.*
- *Direct Text to Speech.*
- *Audio Object.*

2.5.2 Application Programming Interface (API)

Pada sistem pengenalan suara, aplikasi akan menerima even pada saat suara yang diterima telah dikenali oleh *engine*. Komponen *SpeechAPI* yang akan menghasilkan even ini diimplementasikan oleh antar muka *ISpNotifySource*. Lebih spesifik, *SpeechAPI* menggunakan *SetNotifySink*, yaitu aplikasi akan meneruskan *pointer IspNotifySink* ke *ISpNotifySource: :SetNotifySink.ISpNotifySource::SetNotifySink* ini akan menerima pemanggilan melalui *IspNotifySink::Notify* ketika terdapat satu atau lebih even yang menyatakan bahwa aplikasi dapat mengambil data. Biasanya aplikasi tidak mengimplementasikan *ISpNotifySink* secara langsung tetapi menggunakan *CoCreateInstance* untuk membuat obyek *IspNotify Sink*, yang diimplementasikan oleh komponen *CLSID_SpNotify*. Obyek ini menyediakan antar muka *ISpNotifyControl.*

2.5.3 Device Driver Interface (DDI)

DDI menyediakan fungsi untuk menerima data suara dari *SpeechAPI* dan mengembalikan pengenalan frasa pada level *SpeechAPI* paling dasar. Terdapat dua antar muka yang digunakan oleh DDI yaitu *ISpSREngine*, yang diimplementasikan oleh *engine* dan *ISpSREngineSite* yang diimplementasikan oleh *SpeechAPI.*

Antar muka *SpeechAPI* yaitu *ISpSREngineSite* juga menyediakan metode untuk memberikan informasi lebih detail mengenai apa yang dikenali oleh *engine*. *Grammars* dan *speakers* menyediakan informasi ke *engine* yang dapat membantu *engine* untuk melakukan pengenalan pembicaraan lebih baik, disamping juga merupakan bagian penting komunikasi

yang menghubungkan *SpeechAPI* dan *speech engine*. *Engine* menyediakan layanan ke *SpeechAPI* melalui antar muka *ISpSREngine*.

Semua fungsi pengenalan terjadi melalui *Isp SREngine::RecognizeStream*. Ketika *SpeechAPI* memanggil *ISpSREngine::SetSite*, maka *SpeechAPI* memberikan pointer ke antar muka *ISpSREngineSite* dimana kemudian engine dapat berkomunikasi dengan *SpeechAPI* selama *ISpSREngine: :RecognizeStream* dieksekusi.

Sebuah *SpeechAPI* memisahkan pembuat *engine* dari kerumitan untuk mengatur peralatan suara secara detail. *SpeechAPI* menjaga *logical stream* dari raw audio data dengan membuat indeks posisi *stream*. Dengan menggunakan indek posisi *stream*, *engine* dapat melakukan pemanggilan terhadap *ISpSREngineSite:: Read* untuk menerima buffer dari raw audio data selama *ISpSREngine::RecognizeStream* dieksekusi.

Pemanggilan ini akan terjadi sampai semua data yang dibutuhkan tersedia. Jika *ISpSREngineSite::Read* menghasilkan data yang lebih sedikit dari yang dibutuhkan, yang berarti tidak ada data lagi, maka *engine* akan menghentikan eksekusi *ISpSREngine::RecognizeStream*.

(halaman ini sengaja dikosongkan)

BAB 3
METODOLOGI

BAB 3
METODOLOGI

Pada bab ini akan dibahas mengenai desain perangkat lunak dari aplikasi "permainan menembak dengan perintah melalui gerakan tangan dan suara". Aplikasi permainan ini diberi nama " permainan menembak koruptor". Pembahasan desain yang akan digunakan meliputi analisis kebutuhan, perancangan aplikasi, dan perancangan desain permainan.

3.1 Deskripsi Umum Perangkat Lunak

Kecenderungan dalam perkembangan permainan telah mengalami perkembangan yang cukup pesat dari yang berupa stick pada *game console* kemudian keyboard pada game pc. Dan akhir-akhir ini kontrol permainan dibuat lebih interaktif dan realtime seperti pada *console nintendo wii* yang berupa wiimote. Dan teknologi terbaru yang dikembangkan oleh xbox360 berupa kontrol permainan yang dikendalikan dengan bantuan webcam menggunakan deteksi citra. Inilah kontrol permainan yang sedang menjadi bahan perbincangan saat ini.

Pengembang-pengembang permainan sekarang telah memulai membangun permainan maupun simulasi yang dapat dikendalikan oleh deteksi citra secara *realtime* pada webcam untuk xbox360 dan PC. Maka pada tugas akhir ini akan didesain sebuah permainan interaktif dengan aliran *shooting game* yang mampu dikendalikan oleh gerakan tangan dan kontrol suara pemain. Diharapkan aplikasi permainan ini mampu menambah aplikasi pengembangan permainan yang ada di indonesia.

Didalam permainan menembak koruptor ini pemain diharuskan untuk menembak karakter koruptor dan karakter penegak hukum. Dengan menggerakkan tangan kanan menggenggam yang nantinya dikenali oleh fungsi pengenalan obyek sebagai pengganti kursor. Mengucapkan perintah-perintah tembak yang nantinya dikenali oleh fungsi pengenalan suara

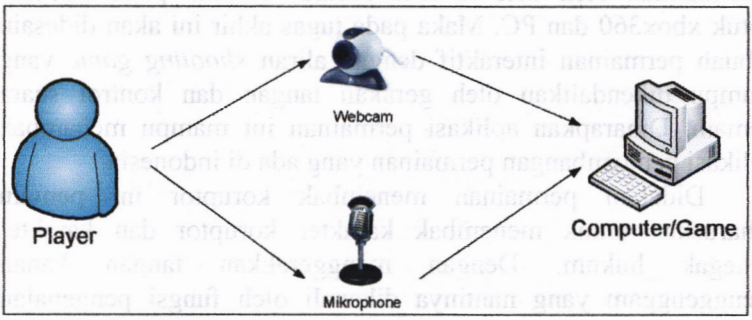
sebagai pengganti tombol perintah. Pada akhirnya, aplikasi permainan ini diharapkan mampu memberikan jawaban atas permasalahan, baik dalam hal pengenalan obyek tangan maupun pengenalan suara pemain.

2 Arsitektur Sistem

Sistem yang dibangun menggunakan arsitektur sistem berbasis desktop. Agar perangkat lunak dapat berjalan maka dibutuhkan sebuah perangkat pendukung yaitu webcam dan mikropon.

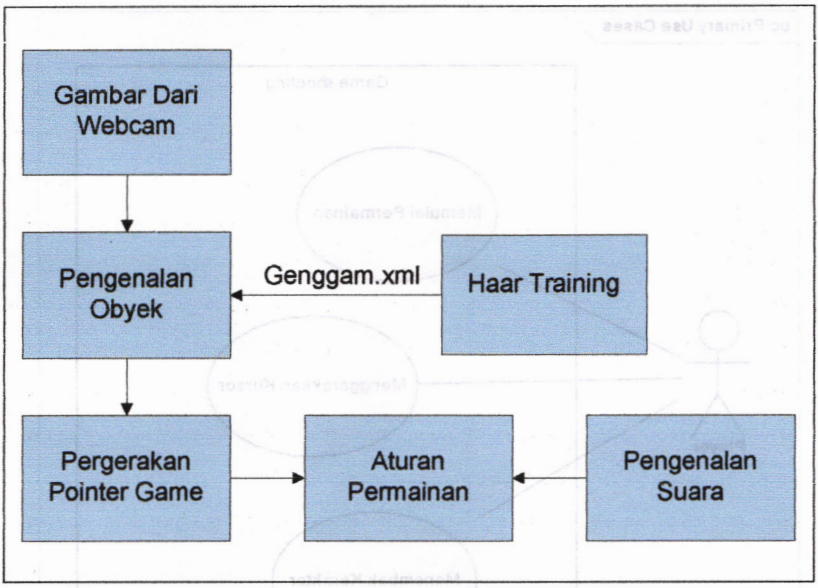
Webcam berfungsi untuk menangkap gerakan tangan pemain yang digunakan sebagai kontrol pengganti mouse. Pada saat tangan berada didepan kamera, maka tangan tersebut akan ditangkap oleh kamera. Kemudian data gambar tangan tersebut oleh kamera diteruskan ke permainan. Oleh permainan dengan *operating system windows* data gambar tersebut diolah oleh fungsi pengenalan obyek.

Mikropon berfungsi untuk mengambil suara pemain yang digunakan sebagai pengganti keyboard/kontrol tombol perintah. Pada saat pemain mengucapkan perintah, maka akan dikenali oleh mikropon kemudian diolah oleh fungsi pengenalan suara. Seperti terlihat pada gambar 3.1 dibawah ini:



Gambar 3.1 Arsitektur Permainan Menembak Koruptor.

Secara garis besar, gambaran sistem "Permainan menembak dengan perintah melalui gerakan tangan dan suara" dapat dipresentasikan seperti yang tampak pada gambar 3.2 dibawah ini.



Gambar 3.2 Sistem Diagram Permainan Menembak.

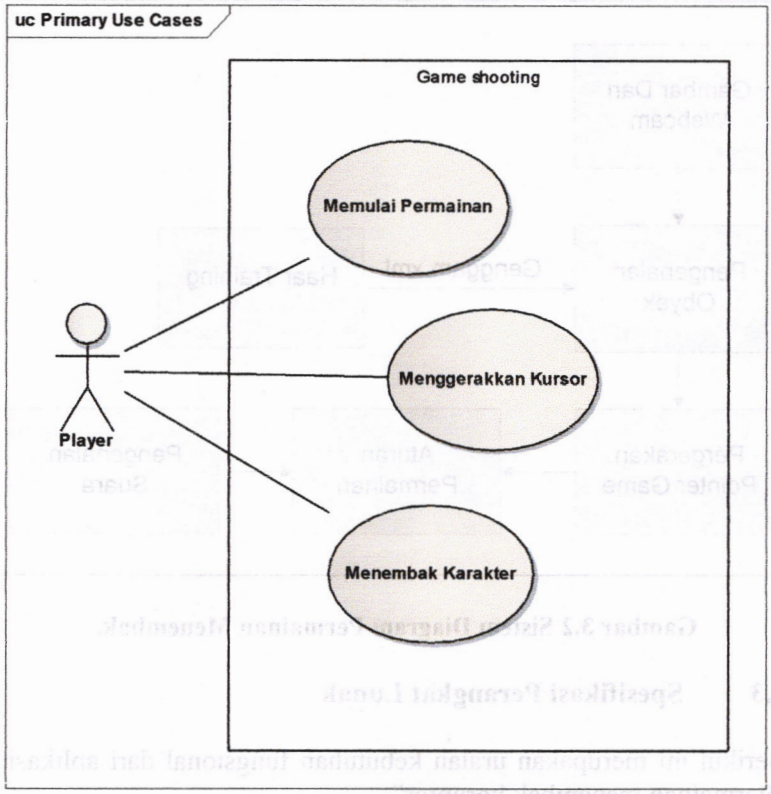
3.3 Spesifikasi Perangkat Lunak

Berikut ini merupakan uraian kebutuhan fungsional dari aplikasi "Permainan menembak koruptor".

1. Memulai permainan
2. Menggerakkan kursor
3. Mengucapkan perintah permainan
4. Mendapatkan nilai
5. Mengakhiri permainan

3.3 Pembuatan Use Case.

Secara garis besar, skenario use case sistem "Permainan Menembak Koruptor" dapat dipresentasikan oleh gambar .



Gambar 3.3 Use Case Permainan Menembak koruptor.

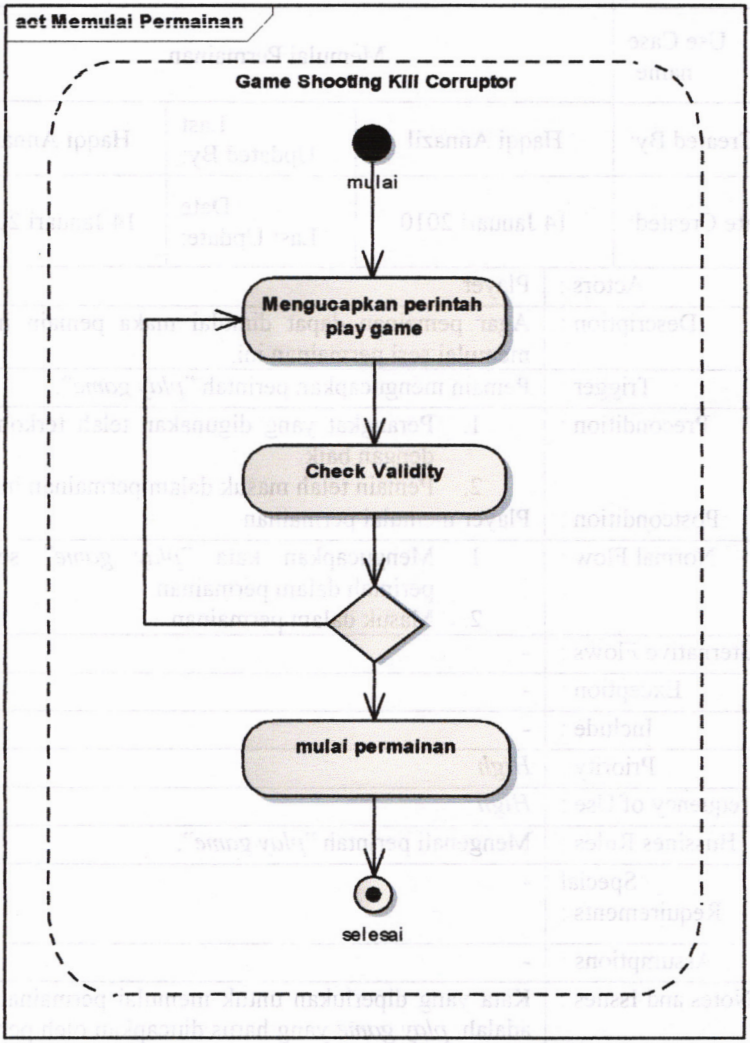
Adapun penjelasan dari masing-masing use case dipaparkan pada subbab-subbab berikut :

3.3.1 Use Case Memulai Permainan.

Use Case ID:	UC-SG-1		
Use Case name:	Memulai Permainan		
Created By:	Haqqi Annazil	Last Updated By:	Haqqi Annazil
Date Created:	14 Januari 2010	Date Last Update:	14 Januari 2010

Actors :	Player
Description :	Agar pemainan dapat dimulai maka pemain harus memulai sesi permainan ini.
Trigger :	Pemain mengucapkan perintah "*play game*".
Precondition :	1. Perangkat yang digunakan telah terkoneksi dengan baik. 2. Pemain telah masuk dalam permainan ini.
Postcondition :	Player memulai permainan.
Normal Flow :	1. Mengucapkan kata "*play game*" sesuai perintah dalam permainan. 2. Masuk dalam permainan.
Alternative Flows :	-
Exception :	-
Include :	-
Priority :	*High*
Frequency of Use :	*High*
Bussines Rules :	Mengenali perintah "*play game*".
Special Requirements :	-
Assumptions :	-
Notes and Issues :	Kata yang diperlukan untuk memulai permainan ini adalah *play game* yang harus diucapkan oleh pemain dengan benar..

Aktivitas yang dilakukan untuk memenuhi use case ini tergambar pada diagram aktivitas di gambar 3.4 berikut ini.



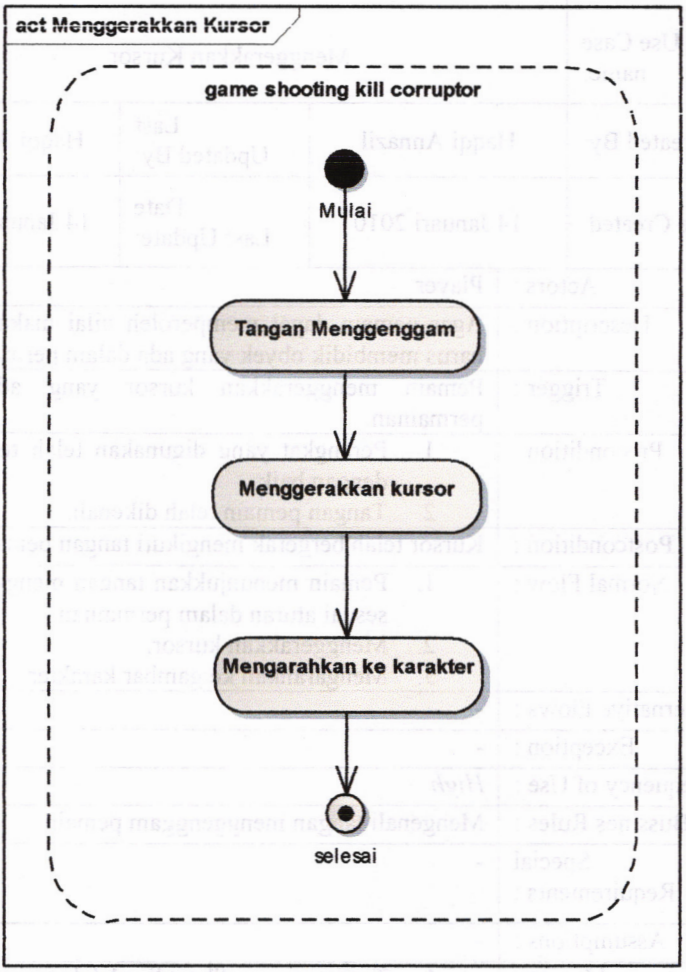
Gambar 3.4 Activity Diagram Memulai Permainan.

3.3.2 Use Case Menggerakkan Kursor.

Use Case ID:	UC-SG-2		
Use Case name:	Menggerakkan Kursor		
Created By:	Haqqi Annazil	Last Updated By:	Haqqi Annazil
Date Created:	14 Januari 2010	Date Last Update:	14 Januari 2010

Actors :	Player
Description :	Agar pemain dapat memperoleh nilai maka pemain harus membidik obyek yang ada dalam permainan.
Trigger :	Pemain menggerakkan kursor yang ada pada permainan.
Precondition :	1. Perangkat yang digunakan telah terkoneksi dengan baik. 2. Tangan pemain telah dikenali.
Postcondition :	Kursor telah bergerak mengikuti tangan pemain.
Normal Flow :	1. Pemain menunjukkan tangan menggenggam sesuai aturan dalam permainan. 2. Menggerakkan kursor. 3. Mengarahkan ke gambar karakter.
Alternative Flows :	-
Exception :	-
Frequency of Use :	*High*
Bussines Rules :	Mengenali tangan menggenggam pemain
Special Requirements :	-
Assumptions :	-
Notes and Issues :	1. Tangan yang dikenali adalah tangan kanan menggenggam 2. Jarak antara kamera dan tangan haurs ideal.

Aktivitas yang dilakukan untuk memenuhi use case ini tergambar pada diagram aktivitas di gambar 3.5 berikut ini.



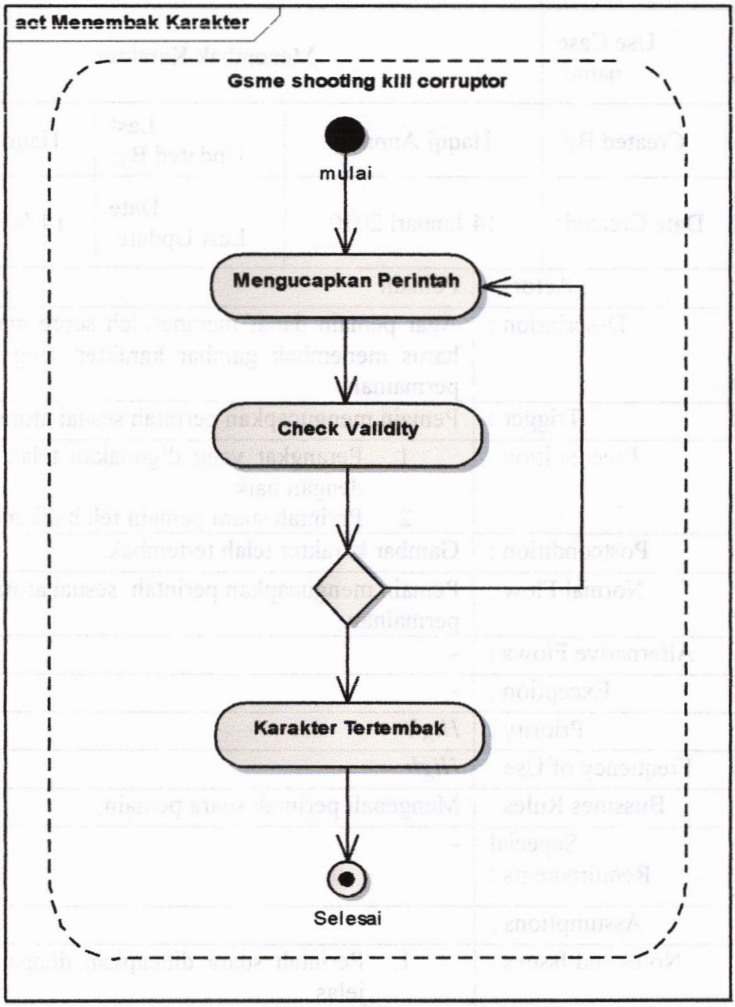
Gambar 3.5 Activity Diagram Menggerakkan Kursor.

3.3.3 Use Case Menembak Karakter.

Use Case ID:	UC-SG-3		
Use Case name:	Menembak Karakter		
Created By:	Haqqi Annazil	Last Updated By:	Haqqi Annazil
Date Created:	14 Januari 2010	Date Last Update:	14 Januari 2010

Actors :	Pemain
Description :	Agar pemain dapat memperoleh score maka pemain harus menembak gambar karakter yang ada dalam permainan.
Trigger :	Pemain mengucapkan perintah sesuai aturan.
Precondition :	1. Perangkat yang digunakan telah terkoneksi dengan baik. 2. Perintah suara pemain telah dikenali.
Postcondition :	Gambar karakter telah tertembak.
Normal Flow :	Pemain mengucapkan perintah sesuai aturan dalam permainan.
Alternative Flows :	-
Exception :	-
Priority :	*High*
Frequency of Use :	*High*
Bussines Rules :	Mengenali perintah suara pemain.
Sepecial Requirements :	-
Assumptions :	-
Notes and Issues :	1. Perintah suara diucapkan dengan baik dan jelas. 2. Perintah suara yang dikenali adalah fire, reload, dor dan kpk.

Aktivitas yang dilakukan untuk memenuhi use case ini tergambar pada diagram aktivitas di gambar 3.6 berikut ini.



Gambar 3.6 Activity Diagram Menembak Karakter.

3.4 Perancangan Antar Muka Grafis.

Pada bab ini akan dijelaskan perancangan terhadap perangkat lunak. Terdapat 2 jenis antarmuka yang menggambarkan aplikasi permainan ini yaitu Halaman menu, dan Halaman permainan. Perancangan yang akan dibuat terdiri :

3.4.1 Rancangan Antar Muka Halaman Menu

Halaman menu merupakan halaman yang menampilkan tentang aturan-aturan dalam permainan ini. Menu ini digunakan untuk menampilkan bagaimana cara bermain dan aturan-aturan yang harus dipatuhi oleh pemain. Di menu ini juga terdapat aplikasi *try voice* yang dapat menampilkan ucapan pemain apabila dikenali dengan benar.

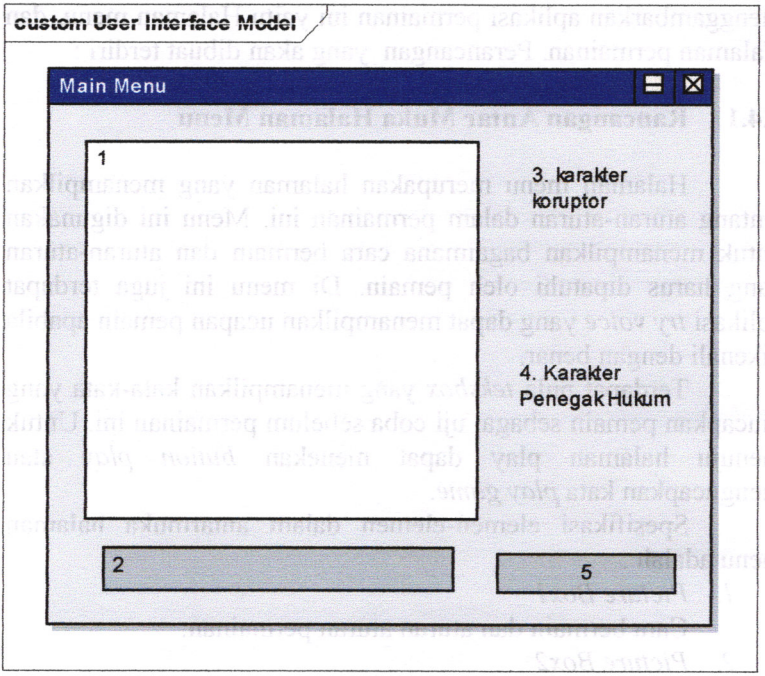
Terdapat pula *teksbox* yang menampilkan kata-kata yang diucapkan pemain sebagai uji coba sebelum permainan ini. Untuk menuju halaman play dapat menekan *button play* atau mengucapkan kata *play game*.

Spesifikasi elemen-elemen dalam antarmuka halaman menu adalah :

1. *Picture Box1*
   Cara bermain dan aturan aturan permainan.
2. *Picture Box2*
   Menampilkan kata yang diucapkan pemain.
3. *Picture Box3*
   Menampilkan gambar koruptor.
4. *Picture Box4*
   Menampilkan gambar penegak hukum.
5. *Button play1*
   Untuk menuju *screen play*.

Terdapat 2 tokoh yang ditampilkan dalam permainan ini yaitu koruptor dan penegak hukum. Dimana karakter koruptor dilambangkan dengan gambar foto "karakter gayus" dan karakter penegak hukum dengan gambar foto "karakter Busro Muqqodah".

Dan untuk memulai permainan ini pemain cukup dapat menekan tombol *button play* atau mengucapkan kata "*playgame*". Seperti terlihat pada gambar 3.7 dibawah ini.



Gambar 3.7 Rancangan Antar Muka Halaman Menu.

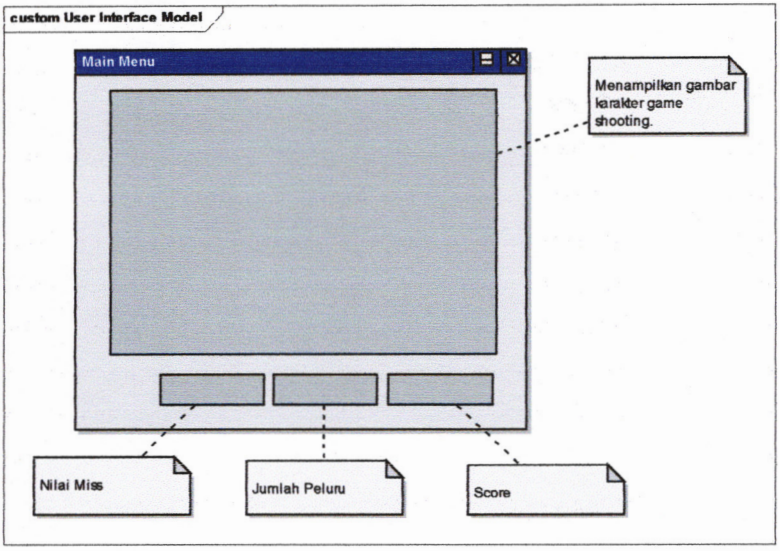
3.4.2 Rancangan Antar Muka Halaman Permainan

Rancangan sebuah halaman permainan adalah halaman dimana permainan akan dimainkan. Didalamnya terdapat beberapa pintu yang digunakan untuk memunculkan gambar koruptor yang akan menjadi sasaran tembak dalam permainan ini. Terdapat *textbox* yang berisi nilai yang didapatkan dalam bermain permainan ini.

Spesifikasi elemen-elemen dalam antarmuka halaman permainan adalah :

1. *Picture Box1*
   Menampilkan backgrond permainan.
2. *Picture Box2*
   Menampilkan tokoh yang harus ditembak.
3. *Text Box1*
   Menampilkan jumlah peluru yang tersedia.
4. *Text Box2*
   Menampilkan jumlah nilai yang didapat.

Tugas pemain adalah mengarahkan pointer ke pintu yang muncul gambar salah satu tokoh dan mengucapkan kata yang sudah ditentukan. Seperti terlihat pada gambar 3.8 dibawah ini.



Gambar 3.8 Rancangan Antar Muka Halaman Permainan.

(halaman ini sengaja dikosongkan)

BAB 4
PEMBUATAN DAN APLIKASI

BAB 4
PEMBUATAN DAN IMPLEMENTASI

Pada bagian ini akan dibahas mengenai pembuatan perangkat lunak yang meliputi lingkungan pembangunan aplikasi pengenalan obyek, aplikasi pengenalan suara dan implementasi permainan.

4.1 Lingkungan Pembangunan Aplikasi

Aplikasi ini dibangun dengan menggunakan perangkat keras *notebook* dengan spesifikasi prosesor Intel(R) Pentium(R) Dual CPU T2390 1,86GHz, memori 1,5GB dan harddisk sebesar 80 GB. Sistem operasi dan beberapa *tools* yang digunakan dalam membangun aplikasi ini adalah :

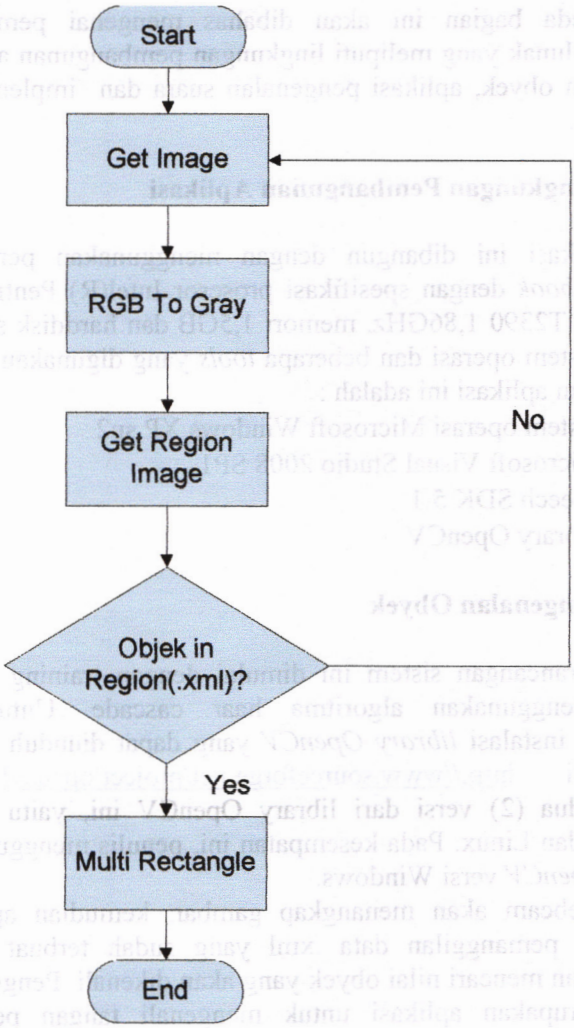
- Sistem operasi Microsoft Windows XP sp2
- Microsoft Visual Studio 2008 SP1
- Speech SDK 5.1
- Library OpenCV

4.2 Pengenalan Obyek

Perancangan sistem ini dimulai dengan training obyek tangan menggunakan algoritma haar cascade. Untuk itu diperlukan instalasi *library OpenCV* yang dapat diunduh secara gratis di http://www.sourceforge.net/project/opencvlibrary. Terdapat dua (2) versi dari library OpenCV ini, yaitu untuk Windows dan Linux. Pada kesempatan ini, penulis menggunakan *Library OpenCV* versi Windows.

Webcam akan menangkap gambar, kemudian aplikasi melakukan pemanggilan data .xml yang sudah terbuat maka aplikasi akan mencari nilai obyek yang akan dikenali. Pengenalan obyek merupakan aplikasi untuk mengenali tangan pemain. Tangan pemain yang dikenali adalah bentuk tangan kanan yang

menggenggam. Gambar 4.1 dibawah ini merupakan flowchat pengenalan obyek.

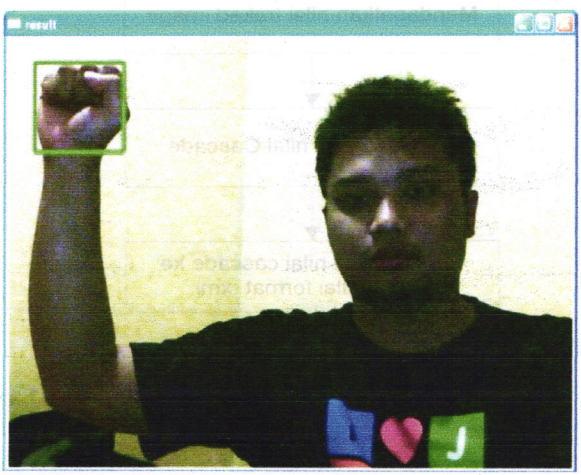


Gambar 4.1 Flowchat Pengenalan Obyek.

Dengan menggunakan fungsi-fungsi dari opencv yang mampu mengenali obyek tangan mengenggam berdasarkan data dari genggam.xml. Berikut ini merupakan pseudocode untuk pengenalan obyek.

```
inisialisasi cascade = "genggam.xml";


get image(); →mendapatkan gambar
gray = cvCreateImage();→membuat suatu gambar gray
get vektor_region();→mendapatkan sebuah region
        if( cascade )→membandingkan dengan nilai cascade
    {
        CvSeq* hand = cvHaarDetectObjects;
        center.x = cvRound((r->x + r->width*0.5)*scale);
        center.y = cvRound((r->y + r->height*0.5)*scale);
        cvRectangle();
        System::Drawing::Point(x,y);→menggambar     area     yg
dikenali
        }
```

Untuk memperoleh data genggam.xml dilakukan proses *haar training* yang akan dijelaskan pada sub bab 4.2.1. Berikut ini merupakan gambar obyek tangan mengenggam dapat dikenali.



Gambar 4.2 Hasil Pengenalan Obyek Tangan Menggenggam.

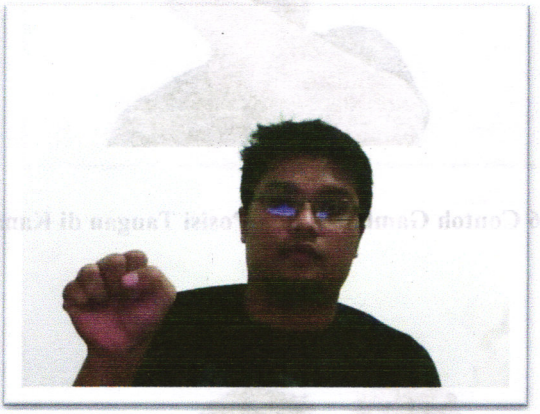
4.2.1 Haar Training

Untuk memperoleh data .xml yang dibutuhkan maka perlu dilakukan proses *haar training* yang dapat dipresentasikan pada gambar blok diagram *haar training* sebagai berikut :

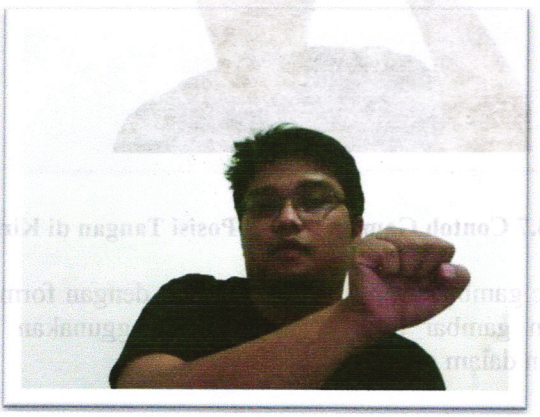
Gambar 4.3 Blok Diagram *Haar Training*.

4.2.1.1 Menyiapkan Data Gambar Positif

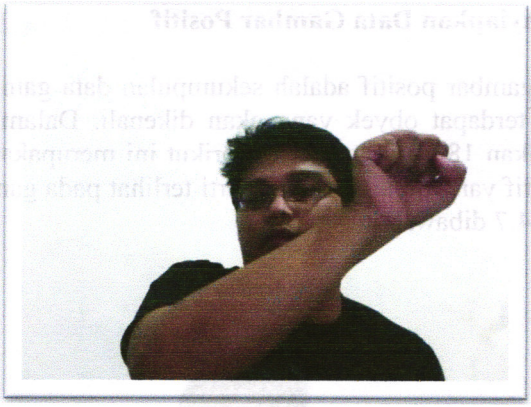
Data gambar positif adalah sekumpulan data gambar yang didalamnya terdapat obyek yang akan dikenali. Dalam training data dibutuhkan 1800 file gambar. Berikut ini merupakan contoh gambar positif yang dibutuhkan. Seperti terlihat pada gambar 4.4, 4.5, 4.6 dan 4.7 dibawah ini.



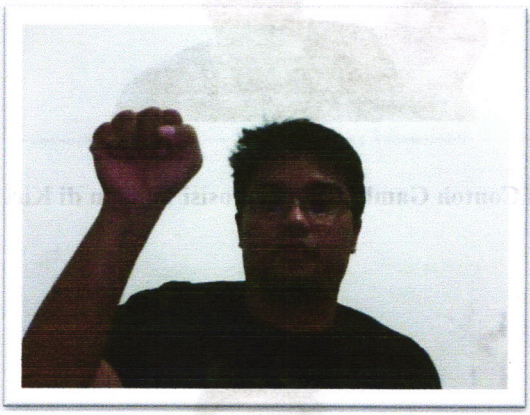
Gambar 4.4 Contoh Gambar Positif Posisi Tangan di Kiri Bawah.



Gambar 4.5 Contoh Gambar Posisi Tangan di Kanan Bawah.



Gambar 4.6 Contoh Gambar Positif Posisi Tangan di Kanan Atas.



Gambar 4.7 Contoh Gambar Positif Posisi Tangan di Kiri Atas.

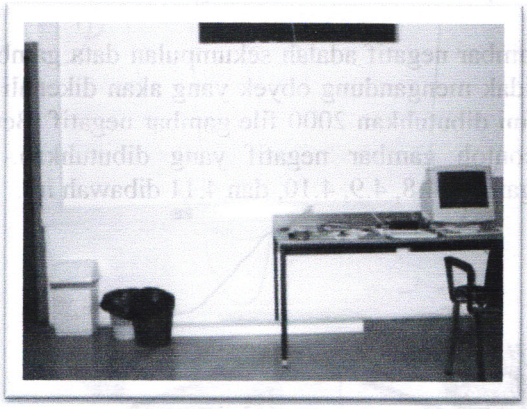
Tipe gambar yang dibutuhkan yaitu dengan format .bmp. Pengambilan gambar positif dilakukan menggunakan webcam dan disimpan dalam satu folder yang sama.

4.2.1.2 Menyiapkan Data Gambar Negatif

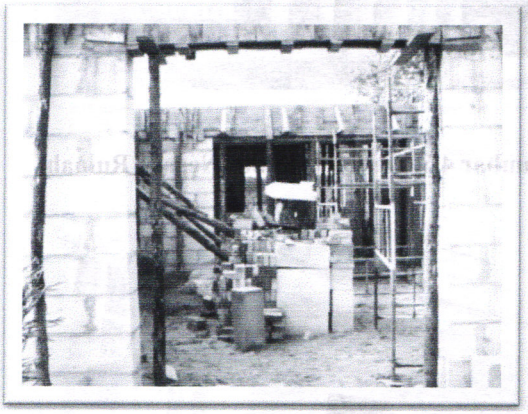
Data gambar negatif adalah sekumpulan data gambar yang didalamnya tidak mengandung obyek yang akan dikenali. Dalam training data ini dibutuhkan 2000 file gambar negatif. Berikut ini merupakan contoh gambar negatif yang dibutuhkan. Seperti terlihat pada gambar 4.8, 4.9, 4.10, dan 4.11 dibawah ini.

Gambar 4.8 Contoh Gambar Negatif Rumah.

Gambar 4.9 Contoh Gambar Negatif Pintu Rumah.



Gambar 4.10 Contoh Gambar Negatif Nuansa Kamar.



Gambar 4.11 Contoh Gambar Negatif Pekarangan Rumah.

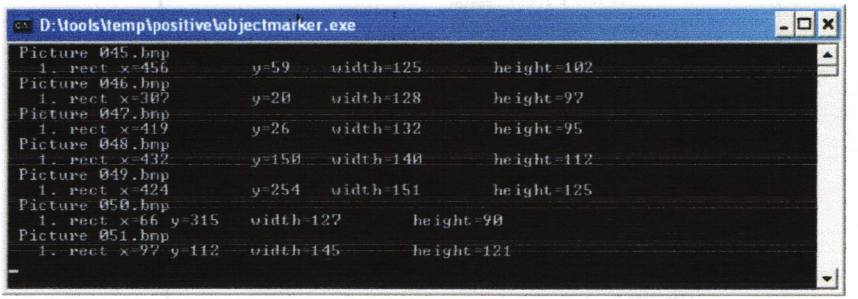
Tipe gambar yang dibutuhkan yaitu dengan format .bmp. Pengambilan gambar negatif kali ini dengan mengunduh dari internet dan disimpan dalam satu folder yang sama.

4.2.1.3 Menyeleksi tangan pada gambar positif

Dalam proses menyeleksi tangan dimaksudkan untuk mengambil data nilai koordinat/posisi obyek yang dikenali dan nilai ukuran obyek pada gambar positif. Seperti terlihat pada gambar 4.12 dibawah ini.

Gambar 4.12 Proses Seleksi Tangan Pada Gambar Positif.

Adapun format penulisan file-nya adalah sebagai berikut : [nama file] [posisi x kiri atas *bounding rectangle*] [posisi y kiri atas *bounding rectangle*] [nilai panjang *bounding rectangle*] [nilai lebar *bounding rectangle*]. Seperti terlihat pada gambar 4.13 dibawah ini.



Gambar 4.13 Nilai Seleksi Obyek.

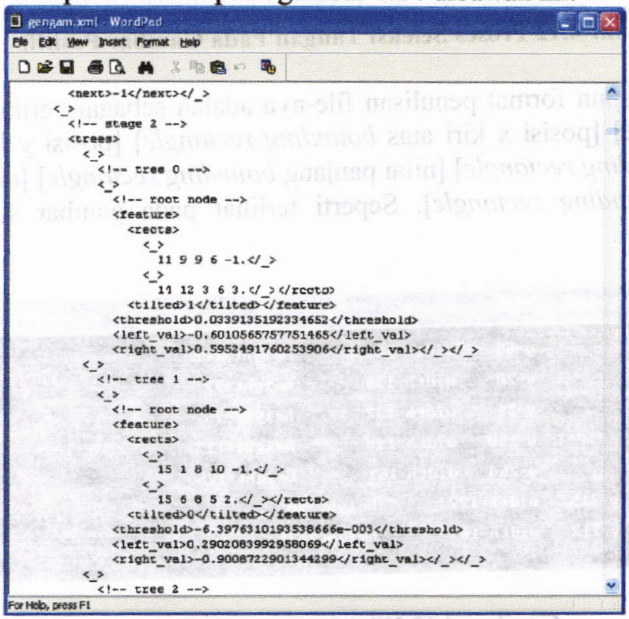
4.2.1.4 Mengkonversi nilai cascade ke dalam format .xml

Proses konversi nilai ini dijalankan dengan perintah *haartraining.exe -data data/cascade -vec data/vector.vec -bg negative/infofile.txt -npos 1800 -nneg 2000 -nstages 20 -mem 1000 -mode ALL -w 30 -h 30 –nonsym.*

Proses menkonversi nilai *cascade* gambar ke dalam nilai .xml membutuhkan variable sebagai berikut :

- Npos = Jumlah gambar positif
- Nneg = Jumlah gambar negatif
- Nstages = Jumlah node
- Mem = memori yang disiapkan untuk proses
- W = ukuran lebar
- H = ukuran tinggi

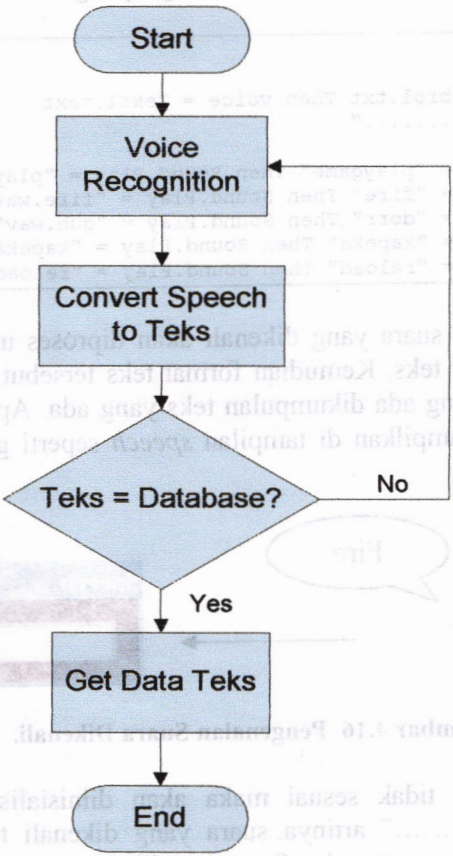
Dan berikut ini merupakan contoh hasil data .xml yang telah diproses. Seperti terlihat pada gambar 4.14 dibawah ini.



Gambar 4.14 Genggam.xml.

4.3 Pengenalan Suara

Dalam proses pengenalan suara dalam proyek akhir ini akan memanfaatkan library *SpeechAPI(SAPI) SDK 5.1* yang mengubah suara menjadi teks. Proses pengenalan suara dapat ditunjukkan pada gambar 4.15 di bawah ini.

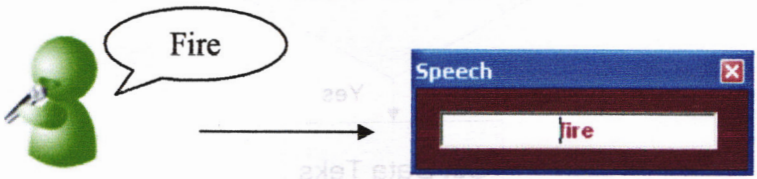


Gambar 4.15 Algoritma Pengenalan Suara.

Definisi *Speech API* adalah suatu library *computer vision* yang dibuat oleh para developer Windows. *Speech SDK 5.1* dapat di unduh secara gratis di http://www.microsoft.com/speech/developers.aspx. Suara yang diucapkan akan ditangkap oleh mikropon dan akan diproses untuk mencari mana kata yang paling sesuai diantara kumpulan kata. Berikut ini merupakan pseudocode fungsi pengenalan suara.

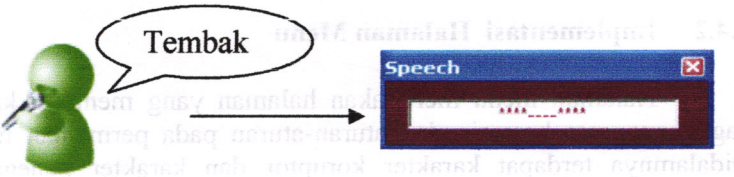
```
get voice
write()
if voice = ngobrol.txt Then voice = Text1.text
else voice = "......."

if Text1.Text = "playgame" Then Sound.Play = "playgame.wav"
if Text1.Text = "fire" Then Sound.Play = "fire.wav"
if Text1.Text = "dorr" Then Sound.Play = "gun.wav"
if Text1.Text = "kapeka" Then Sound.Play = "kapeka.wav"
if Text1.Text = "reload" Then Sound.Play = "reload.wav"
```

Format suara yang dikenali akan diproses untuk diubah menjadi format teks. Kemudian format teks tersebut dicocokkan dengan data yang ada dikumpulan teks yang ada. Apabila sesuai maka akan ditampilkan di tampilan *speech* seperti gambar 4.16 dibawah ini.



Gambar 4.16 Pengenalan Suara Dikenali.

Apabila tidak sesuai maka akan diinisialisasi dengan format teks"....." artinya suara yang dikenali tidak cocok dengan database yang ada. Seperti terlihat pada gambar 4.17 dibawah ini.



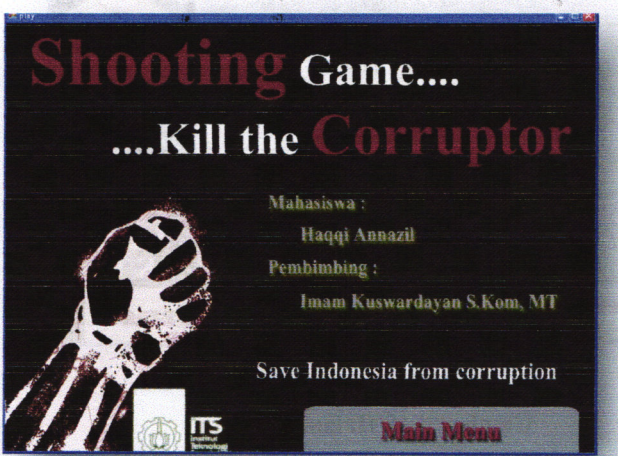
Gambar 4.17 Suara Tidak Dikenali.

4.4 Implementasi Sistem Permainan

Pada permainan menembak ini terdapat beberapa halaman yang berbeda untuk menampilkan permainan.

4.4.1 Implementasi Halaman Intro

Halaman intro merupakan awal pembuka pada permainan ini. Fungsi utama dari halaman adalah untuk menampilkan judul permainan, nama mahasiswa dan pembimbing. Didalam tampilan intro ini hanya akan dilakukan proses menampilkan gambar dan sebuah button untuk menuju ke halaman menu.

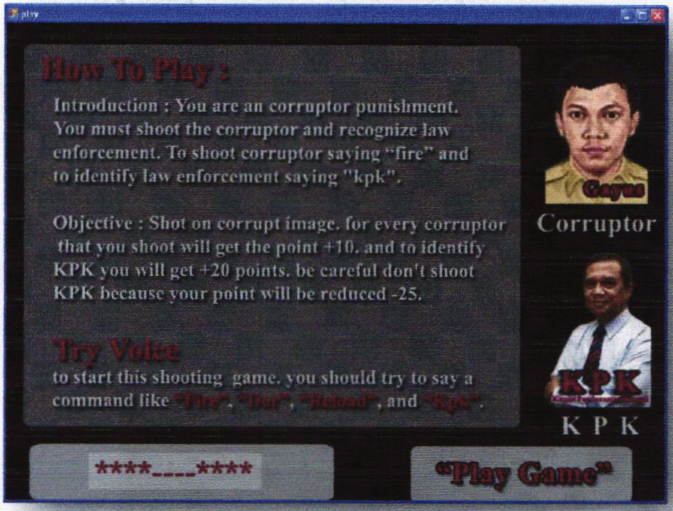


Gambar 4.18 Halaman Intro.

4.4.2 Implementasi Halaman Menu

Halaman menu merupakan halaman yang menunjukkan bagaimana cara bermain dan aturan-aturan pada permainan ini. Didalamnya terdapat karakter koruptor dan karakter penegak hukum. Terdapat pula menu untuk mencoba kata yang yang telah disediakan yaitu *fire*, *dor*, *reload* serta kpk. Proses pengenalan suara pemain dilakukan dengan memanggil fungsi pengenalan suara yang telah dibahas sebelumnya.

Dan apabila suara yang dikenal telah cocok dengan database pada permainan, maka akan dimunculkan pada label *speech*. Untuk memulai permainan dapat dilakukan dengan mengucapkan *"play game"* maka halaman menu akan memanggil halaman permainan.



Gambar 4.18 Halaman Menu.

4.4.3 Implementasi Halaman Permainan

Sebuah halaman permainan merupakan tampilan dimana permainan ini akan dijalankan. Didalam tampilan halaman permainan ini fungsi pengenalan suara dan pengenalan obyek akan dipanggil. Karakter koruptor dan karakter penegak hukum akan dimunculkan bergantian secara random setelah tertembak.

Gambar 4.18 Halaman Permainan.

Aturan perolehan nilai dibutuhkan agar pemain mampu menyelesaikan permainan ini. Dan aturan perolehan nilai dalam permainan ini akan ditunjukkan pada tabel 4.1 berikut ini.

Tabel 4.1 Aturan Perolehan Nilai Permainan.

Tujuan			Menguji Aturan Permainan
	Perintah	Obyek	Score
1	*Fire*	Koruptor	+10
		Koruptor Lebam	-
		KPK	-25
2	*Dor*	Koruptor	-
		Koruptor Lebam	10
		KPK	-25
3	*Kpk*	Koruptor	-25
		Koruptor Lebam	-
		KPK	+20

BAB 5
UJI COBA DAN EVALUASI

BAB 5
UJI COBA DAN EVALUASI

Bab ini membahas pengujian dari fitur-fitur permainan yang sudah dibuat. Pengujian meliputi pengenalan obyek, pengenalan suara, dan skenario permainan.

5.1 Lingkungan Pelaksanaan Uji Coba

Uji coba permainan ini diuji dengan menggunakan perangkat keras notebook dengan spesifikasi prosesor Intel Pentium(R) Dual CPU T2390 1,86 GHz, memori 1,5 GB dan harddisk sebesar 80 GB.

5.2 Uji Coba Aplikasi Permainan

Pada uji coba yang akan dilakukan ini akan diberikan skenario untuk mengetahui kinerja dari aplikasi. Uji coba dilakukan mulai dari masuknya input dan diproses sampai akhirnya menghasilkan output. Dilakukan juga uji coba terhadap aplikasi permainan yang dibuat.

5.2.1 Uji coba Pengenalan Obyek

Untuk uji coba pengenalan obyek ini dilakukan dengan 2 tahap yaitu pengenalan bentuk tangan kanan yang menggenggam dengan bentuk tangan lain dan pengenalan letak koordinat obyek tangan yang dikenal.

5.2.1.1 Uji coba bentuk tangan kanan yang menggenggam dengan bentuk tangan lain.

Sesuai dengan implementasi sistem yang telah dibuat sebelumnya bahwa fitur ini mampu mengenali tangan kanan

menggenggam sekaligus mampu membedakannya dengan bentuk tangan yang lain.

Tabel 5.1 Uji Coba Bentuk Tangan.

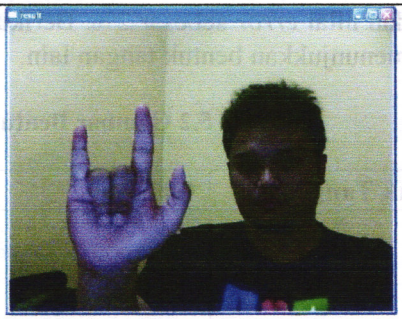
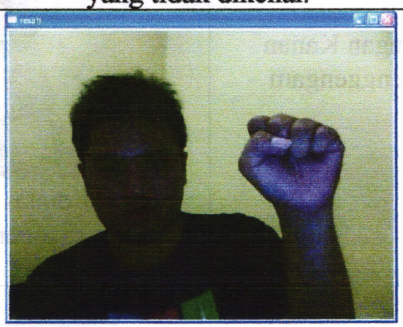
Tujuan		Membedakan bentuk tangan kanan yang menggenggam dengan bentuk tangan lain.		
No	Aksi Aktor	Jumlah	Dikenal	Tidak Dikenal
1.	Tangan Kanan Mengenggam	100	98	2
2.	Tangan Terbuka	100	1	99
3.	Tangan Membentuk Metal	100	13	87
4.	Tangan Kiri Menggengam	100	21	79
5.	Tangan Kanan menggengam posisi miring	100	1	99

Dari uji coba diatas dapat diketahui bahwa fungsi pengenalan obyek mampu mengenali bentuk tangan kanan yang menggenggam dan membedakan dengan bentuk yang lain sebesar

98% dan nilai *error* sebesar 2%. Berikut ini merupakan tabel 5.2 yang menunjukkan bentuk tangan lain.

Tabel 5.2 Gambar Bentuk Tangan.

Bentuk Tangan	Gambar
Tangan Kanan Menggengam	 Gambar diatas adalah bentuk tangan kanan mengenggam yang dikenal.
Tangan Terbuka	 Gambar diatas adalah bentuk tengan terbuka yang tidak dikenal.

Tangan Membentuk Metal	 Gambar diatas adalah bentuk tangan metal yang tidak dikenal.
Tangan Kiri Menggengam	 Gambar diatas adalah bentuk tangan kiri menggengam yang tidak dikenal.
Tangan Kanan menggengam posisi miring	 Gambar diatas adalah bentuk tangan kanan menggengam posisi miring yang tidak dikenal.

5.2.1.2 Uji coba koordinat obyek tangan yang dikenal.

Sesuai dengan implementasi sistem yang telah dibuat sebelumnya bahwa fitur ini mampu mengikuti gerakan obyek dan dapat diimplementasikan kedalam permainan.

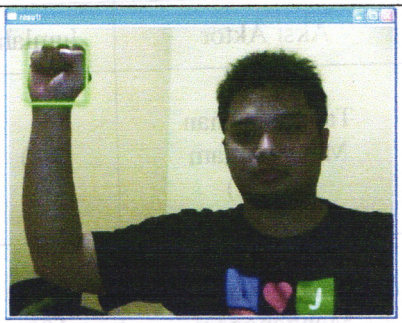
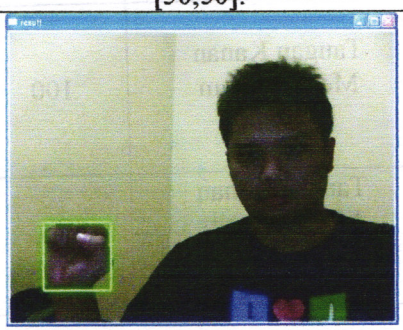
Tabel 5.3 Uji Coba Letak Tangan.

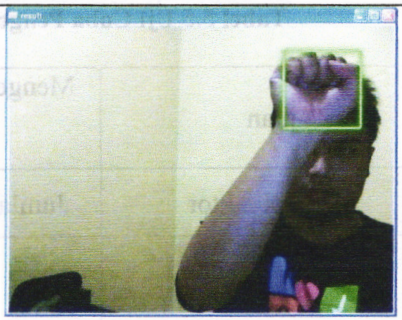
Tujuan		Mengenali bentuk tangan kanan yang menggenggam dikenal pada koordinat tertentu		
No	Aksi Aktor	Jumlah	Dikenal	Tidak Dikenal
1	Tangan Kanan Mengenggam (50,50)	100	98	2
2	Tangan Kanan Mengenggam (400,60)	100	98	2
3	Tangan Kanan Mengenggam (60,600)	100	90	10
4	Tangan Kanan Mengenggam (400,580)	100	85	15

Dari uji coba diatas dapat diketahui bahwa fungsi *tracking* ini mampu mengenali tangan kanan yang menggenggam

dan mengetahui dimana posisi tangan pemain sebesar 93% dan nilai *error* sebesar 7%. Besarnya nilai *error* ini karena *gesture* pemain yang sulit dan warna pakaian pemain yang gelap pada koordinat [400,600] dari nilai resolusi sebesar [640,480]. Berikut ini merupakan tabel 5.4 yang menunjukkan posisi obyek tangan menggenggam yang dikenal.

Tabel 5.4 Gambar Posisi Tangan.

Bentuk Tangan dan koordinat tangan	**Gambar**
Tangan Kanan Mengenggam (50,50)	 Gambar diatas adalah tangan kanan menggengam yang dikenal pada koordinat [50,50].
Tangan Kanan Mengenggam (400,60)	 Gambar diatas adalah tangan kanan menggengam yang terkenali pada koordinat [400,60].

Tangan Kanan Mengenggam (60,600)	 Gambar diatas adalah tangan kanan menggengam yang terkenali pada koordinat [60,600].
Tangan Kanan Mengenggam (400,600)	 Gambar diatas adalah tangan kanan menggengam yang terkenali pada koordinat [400,600].

5.2.2 Uji Coba Pengenalan Suara

Pada uji coba pengenalan suara penulis mencoba kata-kata yang ada dalam basis data permainan. Didalam tabel dapat ditunjukkan tingkat kinerja penggunaan *library SpeechAPI* 5.1 yang mampu mengenali suara pemain dengan baik. Kata yang digunakan yaitu *fire, reload, play game, dor*, dan kpk.

Tabel 5.5 Uji Coba Pengenalan Suara.

Tujuan		Mengenali kata yang diucapkan oleh pemain.		
No	Aksi Aktor	Jumlah	Dikenal	Tidak Dikenal
1	*Fire*	50	48	2
2	*Reload*	50	47	3
3	*Play Game*	50	45	5
4	*Dor*	50	48	2
5	Kpk	50	40	10

Dari uji coba diatas dapat diketahui bahwa fungsi pengenalan suara ini mampu mengenali ucapan pemain dengan tingkat keberhasilan sebesar 91% dan nilai *error* sebesar 9%. Berikut ini merupakan tabel 5.6 yang menunjukkan suara yang dikenal oleh fungsi pengenalan suara.

Tabel 5.6 Gambar Tampilan Pengenalan Suara.

Kata yang Diucapkan	Gambar
Fire	
Reload	
Playgame	
Dor	
Kpk	

5.3 Uji Coba Skenario Permainan

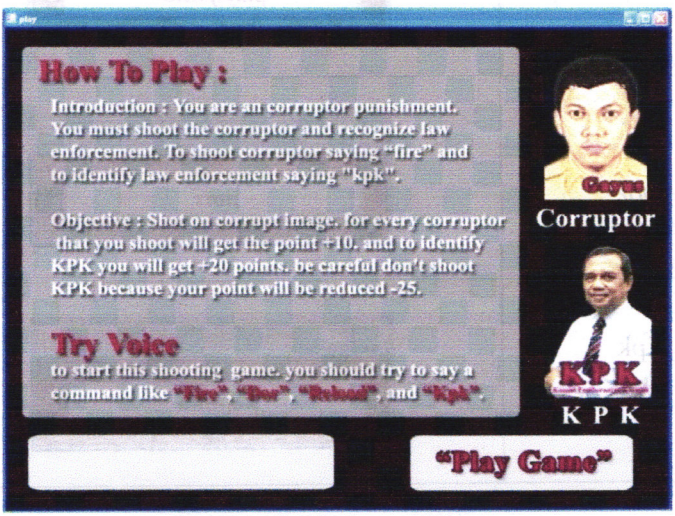
Proses uji coba permainan ini bertujuan untuk mengetahui apakah fungsi-fungsi dan aturan dalam permainan ini dapat dijalankan dengan baik.

5.3.1 Uji Coba Fungsi Permainan

Pada tahap pertama akan diuji coba apakah fungsi-fungsi didalam permainan ini dapat dijalankan dengan baik.

5.3.1.1 Fungsi Tampilan Halaman Menu

Dan gambar berikut ini yang menunjukkan tampilan menu. Dimana gambar karakter koruptor dan penegak hukum dapat ditampilkan dengan baik. Ketika pemain mengucapkan "*play game*" maka menu ini mampu memulai permainan dengan memunculkan halaman permainan. Seperti terlihat pada gambar 5.2 dibawah ini.



Gambar 5.1 Halaman Menu.

5.3.1.2 Fungsi Tampilan Play

Permainan ini mempunyai tingkat kesulitan yang berbeda pada setiap levelnya. Tingkat kesulitan di dalam tiap level dapat ditunjukkan dalam tabel 5.7 berikut ini.

Tabel 5.7 Tingkat Kesulitan Dalam Permainan

Tujuan		Menguji tingkat kesulitan permainan		
No	Tingkatan Level	Gambar yang ditampilkan	Perintah	Status
1.	Level 1	Koruptor	Fire	Berhasil
2.	Level 2	Koruptor KPK	Fire Kpk	Berhasil
3.	Level 3	Koruptor Koruptor Lebam KPK	Fire Dor Kpk	Berhasil

Pada level 1 permainan ini memiliki tingkat kesulitan yang rendah. Aturan permainan yang terlihat pada tabel 5.7. Dimana gambar yang ditampilkan hanya gambar koruptor dan perintah menembaknya dengan mengucapkan kata "fire".

Apabila telah berhasil mengenai obyek maka pemain akan mendapatkan nilai sebesar 10 point. Dan untuk melanjutkan ke level 2 maka pemain harus mengumpulkan nilai sebanyak 100 point.

Tampilan level 1 pada halaman permainan sesuai dengan rancangan dan tampak pada gambar 5.3 dibawah ini.

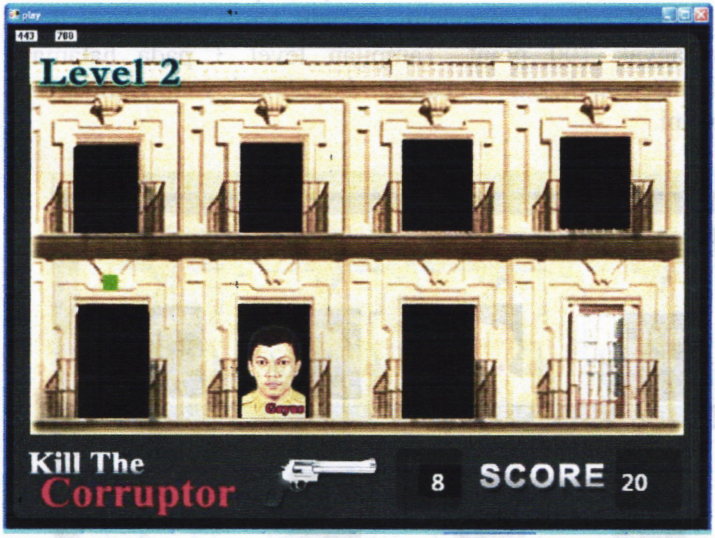
Gambar 5.3 Halaman Permainan Level 1

Pada level 2 permainan ini memiliki tingkat kesulitan sedang. Aturan permainan yang terlihat pada tabel 5.7 dimana gambar yang ditampilkan adalah gambar koruptor dan gambar KPK. Perintah menembaknya dengan mengucapkan kata "*fire*" untuk gambar koruptor dan mengucapkan kata "kpk" untuk gambar KPK.

Apabila telah berhasil mengenai obyek koruptor maka pemain akan mendapatkan nilai sebesar 10 point dan apabila berhasil mengenai gambar KPK akan mendapatkan nilai sebesar 20 point. Hati-hati dalam memberi perintah karena apabila

gambar KPK ditembak maka nilai pemain akan dikurangi sebesar -25 point. Di level 2 ini pemain hanya diberi 10 peluru. Untuk dapat mengisi peluru lagi pemain cukup memberi perintah dengan mengucapkan "*reload*".

Untuk melanjutkan ke level 2 maka pemain harus mengumpulkan nilai sebanyak 100 point. Tampilan level 2 pada halaman permainan sesuai dengan rancangan dan tampak pada gambar 5.4 dibawah ini.



Gambar 5.4 Halaman Permainan Level 2

Pada level 3 permainan ini memiliki tingkat kesulitan yang tinggi. Aturan permainan seperti terlihat pada tabel 5.7 dimana gambar yang ditampilkan adalah gambar koruptor, gambar koruptor Lebam dan gambar KPK. Perintah menembaknya dengan mengucapkan kata "*fire*" untuk gambar koruptor, mengucapkan kata "*dor*" untuk gambar koruptor Lebam dan mengucapkan kata "kpk" untuk gambar KPK.

Apabila telah berhasil mengenai obyek koruptor maka pemain akan mendapatkan nilai sebesar 10 point dan apabila berhasil mengenai gambar KPK akan mendapatkan nilai sebesar 20 point. Hati-hati dalam memberi perintah karena apabila gambar KPK ditembak maka nilai pemain akan dikurangi sebesar -25 point. Di level 3 ini pemain hanya diberi 10 peluru. Untuk dapat mengisi peluru lagi pemain cukup memberi perintah dengan mengucapkan "*reload*".

Level 3 adalah level terakhir pada permainan ini maka untuk menyelesaikan permainan pemain harus mengumpulkan nilai sebanyak 100 point. Tampilan level 3 pada halaman permainan sesuai dengan rancangan dan tampak pada gambar 5.5 dibawah ini.

Gambar 5.5 Halaman Permainan Level 3

BAB 6
PENUTUP

BAB 6
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil pengamatan selama proses perancangan, implementasi, dan uji coba perangkat lunak didalam tugas akhir ini, dapat diambil kesimpulan sebagai berikut :

1. Proses pengenalan obyek dengan menggunakan metode haar training dengan *webcam* sebagai sarana untuk mengambil gambar obyek tangan menggenggam pemain. Sesuai dengan uji coba koordinat obyek tangan yang terkenali yang memiliki nilai keberhasilan sebesar 93% maka proses pengenalan obyek dianggap telah mampu mengenali gerakan tangan pemain dengan baik.
2. Proses pengenalan obyek dilakukan di area yang terang dengan *background* putih dengan posisi tangan dalam keadaan tegak dan tidak boleh miring. Fungsi pengenalan obyek mampu mengenali bentuk tangan kanan yang menggenggam dan membedakan dengan bentuk yang lain sebesar 98%.
3. Berdasarkan hasil uji coba performa yang dilakukan, fungsi pengenalan suara ini mampu mengenali ucapan pemain dengan tingkat keberhasilan sebesar 91%. Maka proses pengenalan suara menunjukkan performa yang cukup baik.

6.2 Saran

Saran yang dapat menunjang untuk pengembangan kedepan agar aplikasi ini berjalan dengan baik pada tugas akhir ini adalah :

1. Untuk menghasilkan sistem pengenalan obyek dengan akurasi yang tinggi, maka dibutuhkan proses training gambar positf dan negatif yang lebih banyak lagi.

2. Aplikasi pengenalan obyek dan pengenalan suara dapat digabungkan dengan *game engine* yang lebih menarik secara grafis.
3. Kecepatan prosessor dan kapasitas memori sangat mempengaruhi kecepatan dalam proses pengenalan obyek maka semakin tinggi spesifikasi komputer yang digunakan maka akan semakin baik.
4. Area disekitar permainan diusahakan sepi atau tidak ada *noise* suara disekitarnya.

DAFTAR PUSTAKA

[1] "Nintendo WII" 2011. Tentang produk Nintendo, http://www.nintendo.com/wii/console, diakses pada 5/01/11.

[2] "*Cam Game*" 2011. Game Menggunakan webcam, http://www.gamecamportal.com/, diakses pada 5/01/2011.

[3] "*Game*", 2011. Wikipedia, http://en.wikipedia.org/wiki/Game_design, diakses pada 5/01/2011.

[4] Crawford, C (2003). Chris Crawford on Game Design. New Riders.

[5] "*Action Game*", 2011. Wikipedia, http://en.wikipedia.org/wiki/Action_game, diakses pada 5/01/2011.

[6] "*Game Design*", 2011. Wikipedia, http://en.wikipedia.org/wiki/Game_design, diakses pada 5/01/2011.

[7] Neo, Naotoshi. "*OpenCV Haar Training*". 2007.

[8] Gary Bradski, Adrian Kaehler. "*Learning OpenCV*". Oreilly team's Intel. 2005.

[9] "Microsoft Speech API 5.1" 2011. Introduction, http://www.microsoft.com/speech/developers.aspx. diakses pada 5/01/2011.

[10] Brenda Brathwaite; Ian Schreiber .(2009). "*Challanger for Game Designers*". Charles River Media.

(halaman ini sengaja dikosongkan)

LAMPIRAN

Genggam.xml

```
<?xml version="1.0"?>
<!--
   24x24 gesture Tangan genggam

-->
<opencv_storage>
<A_gest type_id="opencv-haar-classifier">
  <size>
    24 24</size>
  <stages>
    <_>
      <!-- stage 0 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  3 3 9 16 -1.</_>
                <_>
                  3 7 9 8 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-
0.0223442204296589</threshold>

<left_val>0.7737345099449158</left_val>
            <right_val>-
0.9436557292938232</right_val></_></_>
        <_>
          <!-- tree 1 -->
          <_>
            <!   root node   >
            <feature>
```

```
              <rects>
                <_>
                  0 9 12 5 -1.</_>
                <_>
                  6 9 6 5 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-9.3714958056807518e-
003</threshold>

<left_val>0.5525149106979370</left_val>
            <right_val>-
0.9004204869270325</right_val></_></_></trees>
      <stage_threshold>-
0.3911409080028534</stage_threshold>
      <parent>-1</parent>
      <next>-1</next></_>
    <_>
      <!-- stage 1 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  12 14 12 10 -1.</_>
                <_>
                  12 14 6 5 2.</_>
                <_ >
                  18 19 6 5 2.</_></rects>
              <tilted>0</tilted></feature>

<threshold>0.0127444602549076</threshold>
            <left_val>-
0.7241874933242798</left_val>

<right_val>0.5557708144187927</right_val></_></_
>
```

```
        <_>
          <!-- tree 1 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  2 4 16 8 -1.</_>
                <_>
                  2 8 16 4 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-
0.0203973893076181</threshold>

<left_val>0.3255875110626221</left_val>
            <right_val>-
0.9134256243705750</right_val></_></_>
        <_>
          <!-- tree 2 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  9 6 15 14 -1.</_>
                <_>
                  9 13 15 7 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>1.5015050303190947e-
003</threshold>
            <left_val>-
0.8422530293464661</left_val>

<right_val>0.2950277030467987</right_val></_></_
>
        <_>
          <!-- tree 3 -->
          <_>
            <!-- root node -->
```

```
            <feature>
              <rects>
                <_>
                  0 10 10 5 -1.</_>
                <_>
                  5 10 5 5 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-9.5540005713701248e-
003</threshold>

<left_val>0.2949278056621552</left_val>
            <right_val>-
0.8186870813369751</right_val></_></_>
        <_>
          <!-- tree 4 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  8 0 16 6 -1.</_>
                <_>
                  8 0 16 3 2.</_></rects>
              <tilted>1</tilted></feature>
            <threshold>-9.0454015880823135e-
003</threshold>
            <left_val>-
0.9253956079483032</left_val>

<right_val>0.2449316978454590</right_val></ ></
></trees>
      <stage_threshold>-
0.8027257919311523</stage_threshold>
      <parent>0</parent>
      <next>-1</next></_>
    <_>
</opencv_storage>
  <!-- stage 2 -->
      <trees>
```

```
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  11 9 9 6 -1.</_>
                <_>
                  14 12 3 6 3.</_></rects>
              <tilted>1</tilted></feature>

<threshold>0.0339135192334652</threshold>
            <left_val>-
0.6010565757751465</left_val>

<right_val>0.5952491760253906</right_val></_></_
>
        <_>
          <!-- tree 1 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  15 1 8 10 -1.</_>
                <_>
                  15 6 8 5 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-6.3976310193538666e-
003</threshold>

<left_val>0.2902083992958069</left_val>
            <right_val>-
0.9008722901344299</right_val></_></_>
        <_>
          <!-- tree 2 -->
          <_>
            <!-- root node -->
```

```
            <feature>
              <rects>
                <_>
                  12 23 12 1 -1.</_>
                <_>
                  18 23 6 1 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>3.5964029375463724e-
003</threshold>
            <left_val>-
0.6108912825584412</left_val>

<right_val>0.3585815131664276</right_val></_></_
>
        <_>
          <!-- tree 3 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  0 8 16 11 -1.</_>
                <_>
                  8 8 8 11 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>3.1002631294541061e-
004</threshold>

<left_val>0.2521544992923737</left_val>
            <right_val>-
0.9231098890304565</right_val></_></_></trees>
      <stage_threshold>-
0.6695849895477295</stage_threshold>
      <parent>1</parent>
      <next>-1</next></_>
    <_>
<!-- stage 3 -->
      <trees>
        <_>
```

```
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  12 22 12 2 -1.</_>
                <_>
                  18 22 6 2 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>8.9982077479362488e-
003</threshold>
            <left_val>-
0.6216139197349548</left_val>

<right_val>0.5311666131019592</right_val></_></_
>
        <_>
          <!-- tree 1 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  6 7 10 5 -1.</_>
                <_>
                  6 7 5 5 2.</_></rects>
              <tilted>1</tilted></feature>
            <threshold>5.8961678296327591e-
003</threshold>

<left_val>0.3589088022708893</left_val>
            <right_val>-
0.8741096854209900</right_val></_></_>
        <_>
          <!-- tree 2 -->
          <_>
            <!-- root node -->
            <feature>
```

```
              <rects>
                <_>
                  10 8 3 2 -1.</_>
                <_>
                  10 9 3 1 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-7.3489747592248023e-
005</threshold>

<left_val>0.2021690011024475</left_val>
            <right_val>-
0.8340616226196289</right_val></_></_>
          <_>
            <!-- tree 3 -->
            <_>
              <!-- root node -->
              <feature>
                <rects>
                  <_>
                    15 15 3 4 -1.</_>
                  <_>
                    15 15 3 2 2.</_></rects>
                <tilted>1</tilted></feature>
              <threshold>-1.3183970004320145e-
003</threshold>
              <left_val>-
0.8218436241149902</left_val>

<right_val>0.2309758067131043</right_val></_></_
></trees>
        <stage_threshold>-
0.9460288882255554</stage_threshold>
        <parent>2</parent>
        <next>-1</next></_>
      <_>
        <!-- stage 4 -->
        <trees>
          <_>
            <!-- tree 0 -->
```

```
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  4 18 20 6 -1.</_>
                <_>
                  4 18 10 3 2.</_>
                <_>
                  14 21 10 3 2.</_></rects>
                <tilted>0</tilted></feature>
              <threshold>5.8955969288945198e-
003</threshold>
              <left_val>-
0.7554979920387268</left_val>

<right_val>0.3239434063434601</right_val></_></_
>
        <_>
          <!-- tree 1 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  3 1 20 14 -1.</_>
                <_>
                  3 1 10 7 2.</_>
                <_>
                  13 8 10 7 2.</_></rects>
                <tilted>0</tilted></feature>
              <threshold>8.6170788854360580e-
003</threshold>
              <left_val>-
0.7028874754905701</left_val>

<right_val>0.2782224118709564</right_val></_></_
>
        <_>
```

```
          <!-- tree 2 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  2 11 3 9 -1.</_>
                <_>
                  3 14 1 3 9.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-1.5837070532143116e-
003</threshold>
            <left_val>-
0.7751926779747009</left_val>

<right_val>0.2773326933383942</right_val></_></_
>
        <_>
          <!-- tree 3 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  0 4 12 20 -1.</_>
                <_>
                  0 4 6 10 2.</_>
                <_>
                  6 14 6 10 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>7.9292394220829010e-
003</threshold>
            <left_val>-
0.7723438143730164</left_val>

<right_val>0.2167312055826187</right_val></_></_
>
        <_>
          <!-- tree 4 -->
```

```
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  16 15 6 2 -1.</_>
                <_>
                  16 15 6 1 2.</_></rects>
              <tilted>1</tilted></feature>
            <threshold>-1.4443190302699804e-
003</threshold>
            <left_val>-
0.8843228220939636</left_val>

<right_val>0.2078661024570465</right_val></_></_
>
        <_>
          <!-- tree 5 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  11 8 7 2 -1.</_>
                <_>
                  11 9 7 1 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-4.8251380212605000e-
004</threshold>

<left_val>0.2337501049041748</left_val>
            <right_val>-
0.6776664853096008</right_val></_></_>
        <_>
          <!-- tree 6 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
```

```
                <_>
                  20 15 4 6 -1.</_>
                <_>
                  22 15 2 6 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>8.0077340826392174e-
003</threshold>
            <left_val>-
0.3731102049350739</left_val>

<right_val>0.5163818001747131</right_val></_></_
></trees>
      <stage_threshold>-
1.0588489770889282</stage_threshold>
      <parent>3</parent>
      <next>-1</next></_>
    <_>
      <!-- stage 5 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  14 19 1 2 -1.</_>
                <_>
                  14 20 1 1 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-5.8145709772361442e-
005</threshold>

<left_val>0.3404448032379150</left_val>
            <right_val>-
0.6792302131652832</right_val></_></_>
        <_>
          <!-- tree 1 -->
          <_>
```

```
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  0 6 2 7 -1.</_>
                <_>
                  1 6 1 7 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-1.1419489746913314e-
003</threshold>

<left_val>0.3598371148109436</left_val>
            <right_val>-
0.5890597105026245</right_val></_></_>
          <_>
            <!-- tree 2 -->
            <_>
              <!-- root node -->
              <feature>
                <rects>
                  <_>
                    8 0 10 2 -1.</_>
                  <_>
                    8 0 5 2 2.</_></rects>
                <tilted>1</tilted></feature>
              <threshold>5.8654937893152237e-
003</threshold>
              <left_val>-
0.9622359871864319</left_val>

<right_val>0.1721540987491608</right_val></_></_
>
          <_>
            <!-- tree 3 -->
            <_>
              <!-- root node -->
              <feature>
                <rects>
                  <_>
```

```
                5 8 16 7 -1.</_>
              <_>
                13 8 8 7 2.</_></rects>
            <tilted>0</tilted></feature>
          <threshold>1.1028599692508578e-
004</threshold>
          <left_val>-
0.7706093192100525</left_val>

<right_val>0.2389315962791443</right_val></_></_
>
        <_>
          <!-- tree 4 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  2 9 14 12 -1.</_>
                <_>
                  9 9 7 12 2.</_></rects>
              <tilted>0</tilted></feature>

<threshold>0.0145609602332115</threshold>

<left_val>0.1552716046571732</left_val>
          <right_val>-
0.8984915018081665</right_val></_></_></trees>
      <stage_threshold>-
0.7966647148132324</stage threshold>
      <parent>4</parent>
      <next>-1</next></_>
    <_>
```

Fungsi Pengenalan Obyek

```
void detect_and_draw( IplImage* img )
{

        if(lv==1){
                rl=99;
                reload->Visible=false;
                rs=2;
                deal=1;
        }
        if(lv==2){
                deal=1;
        }

        //=============***********==================
CvPoint pos_gerak;

double scale = 1;


//=============***********==================
        CvPoint ttk1,ttk2;

    CvPoint scala_gerak;
        IplImage *gray, *small_img;
    int i;

    gray = cvCreateImage( cvSize(img->width,img->height), 8,
1 );
    small_img = cvCreateImage( cvSize( cvRound (img-
>width/scale),
                        cvRound (img->height/scale)), 8, 1
);

        cvCvtColor( img, gray, CV_BGR2GRAY );
    cvResize( gray,small_img, CV_INTER_LINEAR );
    cvEqualizeHist( small_img, small_img );
    cvClearMemStorage( storage );

        if( cascade )
    {
         double t = (double)cvGetTickCount();
         CvSeq* hand = cvHaarDetectObjects( small_img,
cascade, storage,
                                                1.1, 2, 0

//|CV_HAAR_FIND_BIGGEST_OBJECT
```

```
//|CV_HAAR_DO_ROUGH_SEARCH

|CV_HAAR_DO_CANNY_PRUNING

//|CV_HAAR_SCALE_IMAGE
                                                ,
                                                cvSize(60, 10)
);
        t = (double)cvGetTickCount() - t;
        //printf( "detection time = %gms\n",
t/((double)cvGetTickFrequency()*1000.) );
        for( i = 0; i < (hand ? hand->total : 0); i++ )
        {
            CvRect* r = (CvRect*)cvGetSeqElem( hand, i );
            //CvMat small_img_roi;
            //CvSeq* nested_objects;
            CvPoint center;
                        CvScalar color = {155,205,143};

            int radius;

            center.x = cvRound((r->x + r->width*0.5)*scale);
            center.y = cvRound((r->y + r-
>height*0.5)*scale);
            radius = cvRound((r->width + r-
>height)^0.25^scale);
                        ttk1.x = center.x - 10;
                        ttk1.y = center.y - 10;
                        ttk2.x = center.x + 10;
                        ttk2.y = center.y + 10;
                        pos_gerak.x=center.x;
                        pos_gerak.y=center.y;

        scala_gerak.x=(int)(pos_gerak.x*1024)/640;

        scala_gerak.y=(int)(pos_gerak.y*768)/400;
                        label3-
>Text=String::Concat("",scala_gerak.x);
                        label4-
>Text=String::Concat("",scala_gerak.y);
                        label5-
>Text=String::Concat("",radius);
                        //this->pointer->Location =
System::Drawing::Point(scala_gerak.x,scala_gerak.y);
                        if((radius <=80)&&( radius >= 50)){

                                cvRectangle( img, ttk1, ttk2,
CV_RGB(0,0,255), 2, 8, 0);
```

```
                                this->pointer->Location =
System::Drawing::Point(scala_gerak.x,scala_gerak.y);

                        }

********* Void Maen**********

void maen()
{
        storage = cvCreateMemStorage(0);
      cascade = (CvHaarClassifierCascade*)cvLoad(
cascade_name, 0, 0, 0 );
    //=====Load Camera==============
    if( !input_name || (isdigit(input_name[0]) &&
input_name[1] == '\0') )
        capture = cvCaptureFromCAM( !input_name ? 0 :
input_name[1] - '0' );


    cvNamedWindow( "result", 1 );
//================================================
    if( capture )
    {
        for(;;)
        {
            if( !cvGrabFrame( capture ))
                break;
            frame = cvRetrieveFrame( capture );
            if( !frame )
                break;
            if( !frame_copy )

                frame_copy = cvCreateImage( cvSize(frame-
>width,frame->height),
                                            IPL_DEPTH_8U,
frame->nChannels );
            if( frame->origin == IPL_ORIGIN_TL )
                cvCopy( frame, frame_copy, 0 );
            else
                cvFlip( frame, frame_copy, 0 );

            detect_and_draw( frame_copy );

            if( cvWaitKey( 10 ) >= 0 )
                goto _cleanup_;
        }

        cvWaitKey(0);
_cleanup_:
```

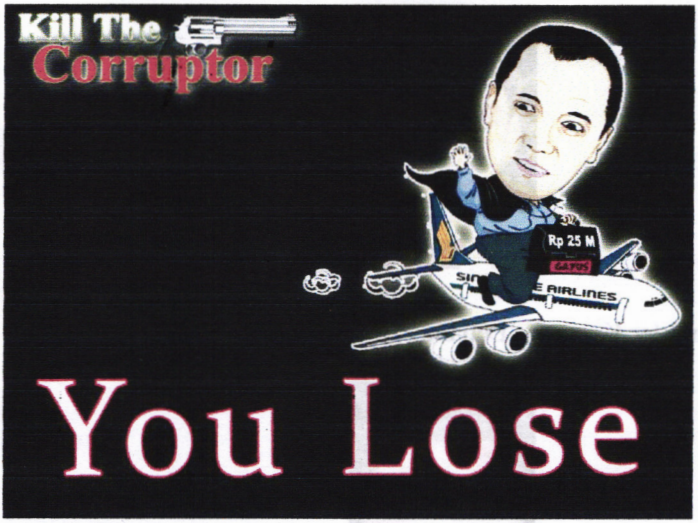
```
        cvReleaseImage( &frame_copy );
        cvReleaseCapture( &capture );
    }
    else
    {
        if( image )
        {
            detect_and_draw( image );
            cvWaitKey(0);
            cvReleaseImage( &image );
        }

    }

    cvDestroyWindow("result");
```

Fungsi Pengenalan Suara

```
Option Strict Off
Option Explicit On
'Imports System.IO
Friend Class Speech_recognition
    Inherits System.Windows.Forms.Form
    Dim WithEvents recognizer As
SpeechLib.SpSharedRecoContext
    Dim grammar As SpeechLib.ISpeechRecoGrammar
    Dim tts As New SpeechLib.SpVoice
    Dim oFile As System.IO.File
    Dim oWrite As System.IO.StreamWriter
    Dim i, b, j, ko, jol, tamp, sip As Integer
    Dim ngobrol(100), fg, nt, kata(100) As String
    Dim mystream As New
System.IO.StreamReader("c:/bahan/ngobrol.txt")

    Sub initReco()
        recognizer = New SpeechLib.SpSharedRecoContext
        grammar = recognizer.CreateGrammar(1)
        Dim topRule As SpeechLib.ISpeechGrammarRule
        topRule = grammar.Rules.Add("myDict",
SpeechLib.SpeechRuleAttributes.SRATopLevel, 1)
        While Not (mystream.EndOfStream)
            i = i + 1
            ngobrol(i) = mystream.ReadLine
            Label1.Text = ngobrol(i)
            topRule.InitialState.AddWordTransition(Nothing,
ngobrol(i), " ", , ngobrol(i), 1, 1)
            grammar.Rules.Commit()
            grammar.CmdSetRuleIdState(1,
SpeechLib.SpeechRuleState.SGDSActive)
        End While

    End Sub

    Private Sub recognizer_FalseRecognition(ByVal
StreamNumber As Integer, ByVal StreamPosition As Object,
ByVal Result As SpeechLib.ISpeechRecoResult) Handles
recognizer.FalseRecognition
        Text1.Text = "....."

    End Sub

    Private Sub recognizer_Recognition(ByVal StreamNumber As
Integer, ByVal StreamPosition As Object, ByVal
RecognitionType As SpeechLib.SpeechRecognitionType, ByVal
```

```
Result As SpeechLib.ISpeechRecoResult) Handles
recognizer.Recognition
            End Sub

    Private Sub Form1_Load(ByVal sender As Object, ByVal e
As System.EventArgs) Handles Me.Load
        tts.Voice = tts.GetVoices().Item(1)
        initReco()
   End Sub
    Sub tulis()
        Timer1.Enabled = True
        oWrite = oFile.CreateText("c:\bahan\sample.txt")
        fg = Text1.Text
        oWrite.WriteLine(fg)
        oWrite.Close()
    End Sub
    Private Sub Timer1_Tick(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Timer1.Tick
        ko = ko + 1
        If (ko > 2) Then
            Text1.Text = "****----****"
            oWrite = oFile.CreateText("c:\bahan\sample.txt")
            fg = Text1.Text
            oWrite.WriteLine(fg)
            oWrite.Close()
            Timer1.Enabled = False
        End If

    End Sub

End Class
```

Tampilan Ketika Permainan Selesai



Tampilan Ketika Misi Berhasil

Gambar Karakter Koruptor

Gambar Karakter Koruptor Lebam

Gambar Karakter Penegak Hukum

BIODATA PENULIS

BIODATA PENULIS

*Penulis, **Haqqi Annazil**, lahir di Surabaya, 18 Maret 1986 dan merupakan anak pertama dari dua bersaudara. Penulis telah menempuh pendidikan dasar di SDN Menur Pumpungan II Surabaya pada tahun 1998, kemudian melanjutkan ke pendidikan menengah di SMP Negeri 6 Surabaya serta pendidikan menengah atas yaitu di SMA Negeri 6 Surabaya. Tahun 2005, penulis melanjutkan jenjang studi D3-Information Teknology di Politeknik Elektro Negeri Surabaya(PENS-ITS). Penulis telah menyelesaikan pendidikan serta melanjutkan jenjang studi S1 di Teknik Informatika ITS Surabaya melalui jalur alih jenjang S1 pada tahun 2009.*

Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Rekayasa Perangkat Lunak (Software Enginering). Penulis memiliki ketertarikan dalam membangun Enterprise Software Development serta Game Development. Penulis dapat dihubungi di email i_am_haqqi_annazil@yahoo.co.id

(halaman ini sengaja dikosongkan)