IMPLEMENTASI OBYEK 3D *VIRTUAL REALITY* PADA APLIKASI BERSEPEDA DI UI BERBASIS 3D GAMESTUDIO

TUGAS AKHIR

Oleh

ANNA GIANTY R.A
06 06 04 2273

DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS INDONESIA
DEPOK
GENAP 2007/2008

IMPLEMENTASI OBYEK 3D *VIRTUAL REALITY* PADA APLIKASI BERSEPEDA DI UI BERBASIS 3D GAMESTUDIO

TUGAS AKHIR

Oleh

ANNA GIANTY R.A
06 06 04 2273

TUGAS AKHIR INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN PERSYARATAN MENJADI SARJANA TEKNIK

DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS INDONESIA
DEPOK
GENAP 2007/2008

PERNYATAAN KEASLIAN SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul :

IMPLEMENTASI OBYEK 3D *VIRTUAL REALITY* PADA APLIKASI BERSEPEDA DI UI BERBASIS 3D GAMESTUDIO

yang dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Ekstensi Teknik Elektro Fakultas Teknik Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari skripsi yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, 10 Juli 2008

(Anna Gianty R.A)

NPM: 06 06 04 2273

PENGESAHAN

Skripsi dengan judul :

IMPLEMENTASI OBYEK 3D *VIRTUAL REALITY* PADA APLIKASI BERSEPEDA DI UI BERBASIS 3DGAMESTUDIO

Dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Ekstensi Teknik Elektro Fakultas Teknik Universitas Indonesia. Skripsi ini telah diujikan pada sidang ujian skripsi pada tanggal 3 Juli 2008 dan dinyatakan memenuhi syarat/sah sebagai skripsi pada Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia.

Depok, 10 Juli 2008

Dosen Pembimbing,

(Dr.Ir.Riri Fitri Sari MM, Msc.)

NIP : 132 127 785

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada:

Dr.Ir.Riri Fitri Sari MM, M.Sc

selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberi pengarahan, diskusi, dan bimbingan serta persetujuan sehingga skripsi ini dapat selesai dengan baik.

Anna Gianty RA NPM : 0606042273 Departemen Teknik Elektro	Dosen Pembimbing: Dr.Ir.Riri Fitri Sari MM,M.Sc

IMPLEMENTASI OBYEK 3D *VIRTUAL REALITY* PADA APLIKASI BERSEPEDA DI UI BERBASIS 3D GAMESTUDIO

ABSTRAK

Saat ini teknologi komputer 3D berkembang begitu pesat sehingga memungkinkan pecinta aplikasi 3D dapat memilih berbagai *tool software* yang ada untuk mewujudkan kreasi yang diinginkannya. Unsur *real* dalam suatu simulasi akan diwujudkan dengan adanya grafik yang tampak seperti dunia nyata misalnya pergerakan pada suatu obyek, perspektif kamera, kemampuan *tool* yang digunakan yang mendukung fakta *real* lain seperti adanya batasan objek yang bergerak menembus sebuah obyek yang lain misalnya pada saat sebuah obyek bergerak bertabrakan dengan dinding, pohon, bangunan dan sebagainya. Selain itu, hal penting lain yang juga perlu dilengkapi adalah kemampuan *user* untuk berinteraksi dengan aplikasi dan mengontrol obyek dalam dunia maya tersebut dengan mudah.

Tool pembantu yang digunakan dalam tugas akhir ini adalah 3D Gamestudio, yang merupakan *authoring system* dan digunakan untuk membuat aplikasi 3D seperti *game* 3D dan dapat juga digunakan untuk membuat aplikasi *virtual reality*. Tujuan utama dari tugas akhir ini adalah membuat obyek 3D sepeda yang memiliki perilaku tertentu yang nantinya akan ditempatkan di jalur sepeda baru di lingkungan UI yang dibuat terpisah dengan tugas akhir ini. Secara terpisah juga nantinya akan digunakan kacamata 3D nirkabel E-Dimensional untuk PC sebagai alat streoskopik untuk membantu melihat aplikasinya secara lebih nyata 3D. Dalam proses ini, sepeda akan bergerak dan berinteraksi dengan lingkungan sekitarnya seperti pada jalur jalannya, pohon-pohon disekelilingnya, dan sebagainya.

Aplikasi ini kemudian diuji coba oleh pemakai yang menjadi penguji penelitian. Hal-hal yang menjadi topik pengujian adalah pengetahuan penguji tentang bahasa pemograman yang digunakan dan didapatkan nilai rata-rata 2,07 dari skala 1-4 yang artinya level pengenalan terhadap aplikasi yang digunakan masih rendah. Hasil kondisi obyek secara umum bernilai rata-rata 3,49 dari skala 1-4, yang diartikan kondisi obyek sudah baik. Tanggapan penguji bahwa *virtual reality* dapat meniru dunia nyata juga baik dengan rata-rata 3,2 dari skala 1-4. Pengembangan proyek ini di masa mendatang mendapat dukungan positif dari semua penguji dengan nilai rata-rata 4 dari skala 1-4.

Kata kunci : *Virtual Reality*, 3D Gamestudio, Obyek 3D

Anna Gianty RA NPM : 0606042273 Electrical Engineering Department	Counsellor : Dr.Ir.Riri Fitri Sari MM,M.Sc

3D OBJECT IMPLEMENTATION OF UI BICYCLING VIRTUAL REALITY APPLICATION BASED ON 3D GAMESTUDIO

ABSTRACT

Nowadays, 3D computer technology grows rapidly and provides alternatives for 3D application lovers with various 3D software to support their imaginations come true. The real elements in a simulation will be actualized with the graphics that imitate the real world such as the object movement, camera perceptions, and object collision handling to other things like wall, tree, or buildings. In addition, it support user to interact with the application and easily control object in this virtual world.

The software used in this final project is 3D GameStudio, one of the authoring systems that will not only support the creation of 3D game but also to make a virtual reality. The main purpose in this project is to create 3D bicycle object that have some behaviors. This object will be placed in the new UI bicycle virtual track which has been prepared separately out of this project. Finally, as a separated process too, the application will be viewed using E-Dimensional 3D wireless glasses for PC, in order to see it more real 3D. In this process, the bike will move and interact with the environment.

Subsequently, this application will be tested by the project testers. The focus evaluation consist of the software used in this project, general object condition, user's respond to virtual reality and the development for the future. The average result is 2.07 from the range 1-4 for the first category, it means the level of language knowledge is still low. The general object condition is having good response and the average is about 3.49 from the range 1-4. The virtual reality category is about 3.2 and it means the testers agree that virtual reality can simulate the real world. The last point about future works is about 4 from the range 1-4, it means all testers support the future development for this project.

Keywords : *Virtual Reality,* 3D Gamestudio, 3D object

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR SINGKATAN

VR	*Virtual Reality*
3D	*Three Dimension (tiga dimensi)*
3DGS	*3D Gamestudio*
PC	*Personal Computer*
WED	*World Editor*
SED	*Script Editor*
MED	*Model Editor*
UML	*Unified Modeling Language*
ABT	*Adaptive Binary Tree*
LOD	*Level Of Detail*
BSP Tree	*Binary Space Rartitioning Tree*

BAB I

PENDAHULUAN

1.1 LATAR BELAKANG

Dunia sekarang ini dengan teknologi yang semakin pesat berkembang memungkinkan manusia untuk melakukan apa saja, yang mungkin dahulu kala hanya ada dalam imajinasi saja. Termasuk diantaranya adalah simulasi 3D di komputer, yang saat sekarang ini memungkinkan semua bidang dapat disimulasikan, seperti simulasi operasi atau bedah dalam bidang medis, simulasi penggunaan pesawat dalam bidang militer, simulasi proses fisika, kimia, ataupun biologi dalam bidang pendidikan, game komputer untuk bidang *entertainment*, dan sebagainya. Aplikasi teknologi 3D seperti *virtual reality* memungkinkan manusia melihat ke sebuah dunia yang tidak nyata yang dibuat untuk memenuhi suatu kebutuhan tertentu. *Virtual reality* ini tidak hanya memungkinkan penggunanya melihat dunia yang telah dibuat tetapi juga membuat penggunanya seakan-akan berada di dunia buatan tersebut. Salah satu contoh aplikasinya adalah simulasi bersepeda di UI khususnya di sebagian jalur sepeda yang baru saja dibangun di UI. Yang menjadi kunci penting dalam simulasi ini adalah obyek sepeda yang akan bergerak dalam jalur sepeda yang disediakan. Obyek ini merupakan sepeda yang nantinya dapat bergerak maju, mundur, kiri, kanan. Selain itu, untuk melengkapi aspek nyata, perspektif kamera juga merupakan unsur yang penting, yaitu dengan perspektif *first person* yang memungkinkan penglihatan langsung dari obyek dan nantinya ditambahkan perspektif tambahan yaitu *third person* yang memungkinkan kita dapat melihat obyek dengan kamera yang mengikuti obyek ataupun yang mengorbit.

Salah satu *tool* pembantu untuk membuat *virtual reality* adalah 3D Gamestudio atau disingkat 3DGS. 3DGS yang digunakan pada tugas akhir ini adalah edisi Gamestudio A7 Extra. 3DGS ini adalah salah satu *tool* 3D yang memiliki 3 buah *editor* yaitu *World Editor* (WED), S*cript Editor* (SED), dan

Model Editor (MED). WED adalah *editor* utama dalam aplikasi ini yang merupakan *editor level* dan merupakan tempat untuk menggabungkan semua bagian dari aplikasi yang akan dibuat (baik itu program, grafik 3D, dan *level* itu sendiri). SED adalah *editor* untuk *script* dengan bahasa pemogramannya adalah Lite-C. Dan terakhir adalah MED yang merupakan *editor* untuk membuat model dan *terrain*.

Satu hal yang dapat membantu manusia untuk masuk ke dalam dunia buatan atau dapat melihat dengan jelas efek 3D dari sebuah simulasi di komputer adalah dengan menggunakan alat bantu seperti kacamata khusus. Kacamata khusus 3D ini dapat membantu melihat sebuah simulasi tampak lebih nyata dibandingkan melihat simulasinya dengan mata telanjang. Lebih jauh, selain menggunakan kacamata, teknologi *virtual reality* sekarang ini dilengkapi dengan informasi penginderaan tambahan yang memungkinkan visualisasinya tampak lebih nyata, seperti penambahan *speaker* atau *headphone*, namun hal ini tidak akan dibahas secara mendalam dalam penulisan tugas akhir ini.

1.2 TUJUAN PENULISAN

Tujuan dari tugas akhir ini adalah :

- membuat simulasi 3D berupa obyek sepeda yang mempunyai perilaku pergerakan sepeda disertai perspektif kamera yang langsung dari obyek sepeda ataupun tidak langsung
- simulasi 3D dibuat dengan menggunakan bahasa pemograman 3D Gamestudio
- sebagai tambahan aplikasi akan dilihat menggunakan kacamata nirkabel 3D E-Dimensional untuk melihat efek 3D yang lebih nyata, dan hal ini akan dibahas lebih mendalam pada penulisan tugas akhir lain yang terpisah.

1.3 BATASAN MASALAH

Pada tugas akhir ini, pengerjaan dibatasi pada pembuatan obyek sepeda 3D dengan menggunakan aplikasi 3D Gamestudio dengan perilaku pergerakan dan perspektif kamera tertentu.

1.4 METODOLOGI PENELITIAN

Penelitian ini dimulai dengan studi literatur, perancangan program obyek pada aplikasi 3D Gamestudio, implementasi dari perancangan yang dibuat, dan terakhir evaluasi hasil kerja.

1.5 SISTEMATIKA PENELITIAN

Hasil kerja pada penelitian ini akan dievaluasi dalam hal pengetahuan tentang bahasa pemograman, keadaan obyek yang dibuat secara umum, tanggapan terhadap *virtual reality* dan pengembangan sistem di masa mendatang.

1.6 SISTEMATIKA PENULISAN

Pada penulisan tugas akhir ini dibagi menjadi lima bagian, yaitu :

BAB I Pendahuluan

Pendahuluan yang berisi latar belakang masalah, tujuan, batasan masalah, metodologi penelitian, sistematika penelitian, dan sistematika penulisan.

BAB II Landasan Teori

Bagian ini berisi penjelasan tentang konsep dan prinsip dasar yang mendasari pembuatan tugas akhir ini.

BAB III Perancangan

Bagian ini berisi langkah kerja, mulai dari analisa *requirement*, hingga hasil perancangan sistem.

BAB IV Pengujian Dan Analisa

Bagian ini berisi analisa program dan evaluasi responden terhadap sistem yang dibuat.

BAB IV Kesimpulan

Bagian ini berisi kesimpulan yang merupakan pernyataan singkat yang menggambarkan isi dari tugas akhir yang dibuat.

BAB II

LANDASAN TEORI

2.1 VIRTUAL REALITY

Virtual reality (VR) atau realitas maya adalah teknologi yang membuat pengguna dapat berinteraksi dengan suatu lingkungan yang disimulasikan oleh komputer (*computer-simulated environment*), yang merupakan suatu lingkungan nyata yang ditiru atau merupakan suatu lingkungan yang benar-benar hanya ada dalam imajinasi. Lingkungan *virtual reality* terkini umumnya menyajikan pengalaman *visual*, yang ditampilkan pada sebuah layar komputer atau melalui sebuah tampilan stereoskopik khusus. Selain itu, beberapa simulasi juga mengikutsertakan tambahan informasi penginderaan, seperti suara melalui *speaker* atau *headphone* [1]. Pengguna dapat berinteraksi dengan dunia virtual ini menggunakan *input devices* standar seperti *keyboard* atau *mouse,* atau dengan *multimodal devices* seperti *wired glove*, *polhemus boom arm*, dan *omnidirectional treadmill*. Lingkungan nyata yang ditiru misalnya simulasi operasi atau bedah dalam bidang medis, simulasi penggunaan pesawat dalam bidang militer, simulasi proses fisika, kimia, ataupun biologi dalam bidang pendidikan, sedangkan yang sangat berbeda dengan kenyataan misalnya *game* 3D. Dalam praktek sekarang ini sangat sukar untuk menciptakan pengalaman realitas maya dengan kejernihan tinggi, karena keterbatasan teknis daya proses, resolusi citra dan lebar pita komunikasi. Tapi bagaimanapun, keterbatasan itu diharapkan dapat diatasi dengan berkembangnya pengolahan citra dan teknologi komunikasi data yang menjadi lebih hemat biaya dan lebih kuat dari waktu ke waktu.

Kata "*virtual reality*" pertama kali terdapat dalam sebuah novel karangan Damien Broderick. Dimasa mendatang *virtual reality* ini akan membawa banyak perubahan dalam kehidupan manusia. *Virtual reality* ini akan diintegrasikan dalam kehidupan sehari-hari dan dapat digunakan untuk kegiatan manusia. Teknologi ini akan berkembang dan mempengaruhi kehidupan manusia, komunikasi antar individu, dan *cognition* (virtual genetik). Karena akan semakin banyak orang yang menghabiskan waktu di ruang virtual maka akan terjadi

”migrasi” ke *virtual space*, dan mengakibatkan perubahan di banyak bidang seperti ekonomi, sosial, dan budaya.

2.2 STEREOSKOPIK 3D

Stereoskopik atau *binocular vision* sudah ditemukan di alam jutaan tahun yang lalu. Manusia dan berbagai jenis hewan memiliki dua mata yang melihat pada arah yang sama. Bahkan beberapa hewan memiliki dua mata pada sisi kepalanya untuk mendapatkan area pandang yang lebih luas, khususnya hewan pemburu. Dua mata yang paralel sejajar memberikan dua perspektif yang berbeda pada pemandangan yang sama. Dari perbedaan antara dua gambar otak dapat mengkalkulasi jarak dari masing-masing obyek yang ada. Penglihatan 3D adalah segala sesuatu tentang informasi yang merupakan konsumsi otak [2]. Stereoskopik pandangan 3D adalah sebuah *sense* yang penting, sebanding dengan kemampuan untuk melihat warna. Tiga dimensi bukan hanya pembicaraan semata, tapi hal ini adalah sesuatu yang penting dalam kehidupan. Penting disini dimisalkan saja seekor binatang peloncat bisa saja jatuh ketika sedang lompat dari satu pohon ke pohon yang lain bila tidak mempunyai kemampuan ini atau seekor predator yang tidak dapat melihat penglihatan 3D akan melewatkan mangsa buruannya. Tentunya bagi manusia *stereo-vision* ini juga merupakan hal yang vital, misalnya dalam profesi manusia, bisa saja sulit untuk mendapatkan lisensi menjadi pilot atau lisensi lain karena tidak memiliki kemampuan ini [2].

Salah satu contoh alat strreoskopik ini adalah kacamata elektronik 3D yang biasa dipakai oleh ilmuwan, *engineer*, arsitek, dokter bedah, atau orang-orang yang menyetir mobil kecil di permukaan Mars, karena tanpanya mereka tidak dapat mengerjakan pekerjaan mereka dengan sebaik mungkin.

Terdapat 3 alasan kenapa stereo-3D belum menjadi standar di film, televisi, fotografi, dan juga belum meluas di perangkat lunak komputer karena :

- masalah teknik termasuk isu ergonomi
- harga
- standarisasi perangkat keras dan perangkat lunak yang masih kurang

Stereo vision tidak hanya membicarakan mengenai dua mata. Peran yang sangat penting sebenarnya dimainkan oleh otak. Banyak orang memiliki mata yang sehat, namun otaknya tidak dapat menggabungkan dua perspektif dalam satu penglihatan dengan *sense* 3D yang benar. Jadi jika mata belum dapat melihat perbedaan ketika menonton material-3D dibandingkan 2D, hal tersebut perlu diperiksakan, dan dicek dengan sebuah tes *stereo-vision*. Kemampuan untuk melihat 3D dikembangkan pada masa kanak-kanak. Jika fase kritikal tersebut terhilang, karena masalah mata pada saat itu, maka anak bisa saja tidak akan dapat melihat 3D yang nyata dalam hidupnya, walaupun matanya sudah sukses dirawat di kemudian hari. Satu hal yang perlu diperhatikan adalah anak-anak sebaiknya tidak diperbolehkan untuk melihat menggunakan alat-alat elektronik yang saat ini banyak dipakai karena belum tentu memberi pengaruh yang baik, jadi sebaiknya pertama kali belajar melihat stereoskopik melalui dunia nyata dulu [2].

Untuk stereoskopik 3D dibutuhkan *software Stereo-3D* dalam berbagai penggunaan baik itu untuk fotografi, film, tv, video, atau perangkat lunak komputer. Sebuah pengalaman 3D yang nyata tidak dapat dilihat diluar material yang standar karena manusia tidak akan mendapat efek 3D diluar udara tipis. Pada dasarnya ada dua tipe dari perangkat elektronik stereo-3D yaitu :

- *VR-Helmets/Head Mounted Devices* yaitu perangkat yang memproduksi sebuah gambar sendiri. Sebenarnya ini adalah monitor yang memperbolehkan penglihatan stereoskopik dengan monitor LCD atau CRT yang kecil untuk masing-masing mata. Sebagai tambahan, helm ini memiliki *headtracker* (yang mengganti atau melengkapi *keyboard, mouse,* atau *joystick input* dengan gerakan kepala) dan dilengkapi *stereo-headphone* [2].
- *Liquid Crystal-Shutterglasses/Polarization Glasses* yaitu kacamata yang mempengaruhi anda untuk melihat gambar pada monitor yang standar. *Shutterglasses* dan *shutter-screen* digunakan dalam hubungan dengan monitor tabung cahaya katode atau proyektor. Dua gambar untuk penglihatan stereoskopik ditampilkan pada monitor yang standar. Untuk pembagian waktunya, gambar untuk mata kiri ditampilkan, setelah itu baru gambar untuk mata kanan muncul untuk durasi yang sama dan seterusnya.

Tugas dari kacamata ini adalah untuk mencegah mata kiri melihat gambar yang didedikasikan untuk mata kanan, demikian sebaliknya. Untuk melakukan ini cahaya diblok oleh *"LCD-shutter*". Ada dua cara untuk melakukan ini, salah satunya dengan menempatkan *shutter* pada monitor dan melihatnya melalui kacamata polarisasi non-elektrik dan pasif. Cara kedua adalah menempelkan *shutter* pada kacamata. Inilah yang disebut *shutterglasses* yang biasanya lebih mudah daripada cara polarisasi yang ditambahkan pada monitor. Shutterglasses ini tidak terbatas pada masalah resolusi dan warna [2]. Ada beberapa cara untuk mengkoneksikan *shutterglasses* ke sistem komputer yaitu :

1.) Melalui ISA-card

2.) Melalui parallel atau serial port

3.) 3D-port yang memang ada pada *graphics-board VGA* (solusi professional) yang dilewatkan melalui kabel (dengan atau tanpa fitur yang khusus).

2.2.1 Kacamata Wireless 3D Edimensional Untuk PC

Pada tugas akhir ini kacamata yang dipakai untuk melihat efek 3D yang leih nyata adalah kacamata 3D tanpa kabel untuk PC keluaran E-Dimensional. Perangkat tersebut akan dibahas sekilas pada bagian ini, namun untuk pembahasan teori yang lebih mendalam terpisah pada pembuatan tugas akhir yang lain. Kacamata ini membantu kita melihat efek 3D yang lebih baik lagi pada sebuah tampilan 3D di layar monitor. Kacamata tanpa kabel ini memiliki komponen tambahan yang khas dibanding kacamata dengan kabel yaitu pemancar inframerah. Kacamata ini menggunakan 2 baterei model CR-1620 dan letaknya menghadap pada arah yang sama dengan kedua lensa dengan kutub (+) di sebelah luar atau berhadapan dengan kening saat kita menggunakannya [3].

Kacamata ini memiliki tombol *on* yang kecil diatas lensa sebelah kiri. Pastikan untuk menekannya ketika akan menerima sinyal. Kacamata ini akan otomatis *turn off* bila berada diluar jangkauan pemancar atau tidak ada gambar 3D yang ditampilkan selama 1 menit, jadi menekan tombol *on* beberapa kali tidak

akan membuatnya *off*. Tombol *on* seharusnya ditekan ketika ada gambar 3D pada layar dan sinyal dikirimkan ke kacamata (gambar yang ditampilkan itu adalah 3D jika monitor terlihat lebih buram dilihat tanpa memakai kacamata).

Komponen lain untuk menggunakan alat ini adalah *dongle* yang merupakan konektor pada komputer untuk mengkontrol perangkat yang lain. *Dongle* E-Dimensional adalah kotak segitiga dengan kabel VGA yang keluar dari satu sisi. *Dongle* ini akan mengontrol kacamata sebaik mensinkronisasikannya dengan komputer. *Dongle* ini memiliki 3 koneksi di bagian belakang, satu ditengah untuk monitor, yang paling besar untuk pemancar tanpa kabel, dan yang satunya berwarna perak untuk koneksi kacamata dengan kabel yang dikoneksikan secara langsung. Saat ini *dongle* hanya mendukung koneksi VGA bukan DVI. Pemancar kacamata tanpa kabel dikoneksikan langsung pada *dongle*, dan disarankan meletakkan pemancar pada bagian atas monitor. Pastikan pemancar itu menghadap langsung pada wajah sehingga didapatkan posisi berhadapan langsung dan tanpa halangan dari kacamata ke *pemancar*. Untuk diingat pemancar ini tidak menyala untuk menyatakannya *on*, karena dua bola lampu yang ada menggunakan inframerah yang tidak terlihat dan tidak merusak mata [3].

Sistem yang dapat dipakai menggunakan kacamata ini adalah monitor eksternal CRT atau LCD dengan koneksi VGA (bukan layar laptop) dan disarankan untuk memakai monitor memiliki refresh 70Hz.

Untuk perangkat lunak atau bagi E-Dimensional disebut sebagai *driver*, karena bersifat men-*drive* perangkat kacamatanya, perlu dicek lebih dahulu apakah PC memiliki *graphics card* dari Nvidia atau *video card* tipe lain misalnya dari ATI, Intel, ataupun lainnya. Jika dari Nvidia maka perlu digunakan driver Nvidia dan bila memiliki tipe lainnya bisa menggunakan *driver* E-Dimensional yang bersifat universal. Untuk media *player* animasi atau video bergerak yang dibuat bisa dijalankan dengan VLC media *player portable* [3].

Gambar 2.1 Kacamata 3D E-Dimensional

2.3 APLIKASI 3D

Dalam pemograman 3D seringkali kita mendengar istilah *3D engine, 3D language,* dan *3D authoring system*. Sebelum kita lebih jauh mempelajarinya maka perlu dulu untuk mengenal lebih jauh perbedaannya. Perbedaan antara ketiga istilah tersebut dapat dilihat pada penjelasan dibawah ini.

3D Engine adalah sebuah *library* untuk fungsi grafik 3D. *Engine* ini banyak tersedia di Internet, mulai dari yang bersifat *free* sampai *nonfree*. 3D engine ini meminta *programming* dengan sebuah sistem pengembang yang bersifat eksternal, biasanya Microsoft Visual C++. Pemograman game di *3D engine* menawarkan fleksibilitas yang maksimum, khususnya saat anda membutuhkan akses ke *source code engine*. Tapi bagaimanapun, untuk mempelajari *engine* ini dibutuhkan usaha yang maksimal dan investasi waktu yang banyak sebelum membuat *game* atau aplikasi lainnya [4].

Pendekatan yang lebih mudah dari pembuatan aplikasi 3D adalah *3D Language*. Dapat dilakukan pemograman dengan bahasa *script* yang secara khusus didesain untuk game 3D. Bahasa seperti ini sebenarnya tidak menawarkan fleksibilitas, tetapi menghindarkan banyak masalah yang biasa ditemui pada pemograman yang sesungguhnya. Banyak bahasa 3D menggunakan BASIC, yang masih terus dikembangkan sampai sekarang, tetapi bahasa ini bukan pilihan yang terbaik untuk proyek yang kompleks, karena strukturnya yang masih tidak rapi. Pilihan yang lebih baik adalah C atau Java [4].

Cara yang paling mudah dari ketiga sistem ini untuk membuat suatu *game* atau aplikasi adalah a*uthoring system*, yang memiliki *3D engine* sendiri dan *visual editor* untuk membuat prototype *game* atau aplikasi 3D secara cepat. Tentunya, hanya *game* atau aplikasi sederhana yang dapat dibuat tanpa pemograman. Karena itu, *authoring system* normalnya juga menyediakan bahasa *scripting* untuk pemograman. Dengan, sistem ini, sebuah aplikasi dapat diselesaikan dalam pembagian waktu agar s*ource code* atau bahasanya termasuk fungsi *library* dari *3D engine*-nya dapat dimengerti[4].

Aplikasi yang dipakai untuk membuat 3D sudah banyak tersedia, untuk memilihnya tergantung kebutuhan kita saja. Pada sub-bab berikutnya akan dijelaskan mengenai bahasa pemograman 3D yang akan digunakan yaitu 3D GameStudio yang merupakan *game authoring system*. Dari namanya, kita langsung dapat mengetahui tujuan software ini adalah membuat *game*, tapi walaupun demikian *tool* ini tetap dapat membantu aplikasi yang akan kita buat nanti yaitu simulasi *virtual reality*.

Berikut adalah tabel perbandingan ketiga hal yang telah dijelaskan diatas:

Tabel 2.1 Perbandingan *3D Engine, 3D Language, Authoring System* untuk Beberapa Aplikasi [4]

	FPS	Authoring Systems					3D Languages				3D Engines							
Fitur	Game Maker	Quest 3D	CC	Vir tools	A7 Extra	A7 Com	Dark Basic	DB Pro	Blitz	Lite C	Tv3D	Irrlicht	Torque	Nebula	Crystal	Ogre	PR	Irrklang
DirectX version	DX6	DX8	DX8	DX9	DX9	DX9	DX7	DX9	DX7	DX9	DX8	DX9	DX7	DX8	OG	DX8	DX9	DX8
Scene manager	-	-	Octr	Portal	ABT	ABT	-	BSP	-	ABT	BSP	Octr	—	Octr	Portal	BSP	BSP	BSP
LOD system	-	ya	ya	ya	ya	ya	-	ya	ya	ya	-	-	ya	Ya	ya	ya	ya	ya
Terrain (tanah)	ya	-	ya	ya	ya	ya	ya	ya	ya	ya	ya	ya	ya	ya	ya	ya	ya	ya
Shadow mapping	-	ya	-	ya	ya	ya	-	**	ya	**	ya	**	ya	-	ya	ya	ya	ya
Dynamic shadows	-	-	ya	ya	ya	ya	-	ya	-	ya	ya	ya	ya	ya	ya	ya	ya	ya
Shaders	-	ya	ya	-	-	ya	-	ya	-	-	ya	-	-	ya	ya	ya	ya	ya
Particle generator	-	ya	ya	ya	ya	ya	-	ya	-	ya	ya	ya	ya	ya	ya	ya	ya	ya
Beam generator	-	-	-	-	-	ya	-	-	-	-	-	-	-	ya	ya	-	-	-
Template system	ya	-	-	ya	ya	ya	-	-	-	-	-	-	-	-	-	-	-	-
Bones animation	-	ya	ya	ya	ya	ya	-	ya	ya	ya	ya	ya	ya	ya	ya	ya	ya	ya
Save/Load system	ya	-	—	ya	ya	ya	-	-	-	ya	-	-	-	-	-	-	-	ya
Plugin support	-	ya	ya	ya	ya	ya	-	ya	ya	ya	-	-	-	ya	ya	-	-	ya
Network system	-	—	ya	ya	-	ya	ya	ya	ya	-	-	ya	ya	ya	ya	ya	ya	ya
Physics engine	-	ya	ya	$5000	ya	ya	-	-	-	ya	-	-	-	-	-	-	-	ya
Level editor	-	ya	ya	ya	ya	ya	-	-	ya	-	-	-	ya	-	-	-	ya	ya
Model editor	-	ya	-	ya	ya	ya	-	-	-	ya	-	-	-	-	-	-	-	-
Script editor	-	ya	ya	ya	ya	ya	Ya	ya	ya	ya	-	-	-	-	-	-	-	-
Script compiler	-	-	-	-	ya	ya	-	ya	ya	ya	-	-	-	-	-	-	-	-
Script debugger	-	-	-	-	ya	ya	-	ya	-	ya	-	-	ya	ya	-	-	-	-
Script syntax	-	LUA	Lisp	Chart	C	C	Basic	Basic	Basic	C	-	-	C	TCL	Python	-	-	C

2.4 3D GAMESTUDIO

3D Gamestudio atau yang sering dikenal sebagai Gamestudio dan disingkat 3DGS, adalah sistem pengembangan komputer *game* 3D yang mengijinkan pengguna membentuk *game* 3D ataupun aplikasi *virtual reality* lainnya, dan mempublikasikannya secara *free* (tanpa royalti). 3DGS ini hadir dengan sebuah *editor* model, *editor level*, dan *editor script* dilengkapi *debugger*. 3DGS ini juga berisi kumpulan tekstur dan koleksi *artwork*, dan sistem *template game* yang memungkinkan pembuatan *game* sederhana seperti *game* tembak-tembakan yang sederhana tanpa pemograman. Untuk *game* yang lebih kompleks, perlu diintegrasikan *scripting* bahasa Lite-C, ataupun sistem pengembangan eksternal lain seperti Visual C++ atau Delphi. Edisi komersil tetap dibutuhkan untuk mendapatkan fitur *networking* ataupun efek *shaders*, namun beberapa orang mungkin lebih memilih *engine open source* seperti Irrlich yang memang tidak begitu mudah ditangani namun dapat dimodifikasi oleh penggunanya dan bersifat *free* [5].

Perusahaan 3DGS ini adalah pengembang *tool game* yang pertama menerima sertifikat ISO 9001 untuk sistem kontrol kualitas. Sebelum sebuah *update* Gamestudio baru diluncurkan, diuji dulu beberapa bulan oleh tim internal perusahaan dengan pengujian sampai 100 beta. Setelah itu, di-*upload* ke forum Gamestudio untuk *test beta*, yang normalnya mencakup 1000 penguji. Dengan cara itulah maka *upgrade* terbaru di *downloadpage* baru boleh ditampilkan [4].

3DGS ini dapat digunakan untuk membuat *game* 2D dan 3D, simulasi, ataupun aplikasi multimedia lainnya. Untuk *game* 3D normalnya terdiri dari satu atau beberapa lingkungan *virtual* atau ”level”. Sebuah level dibangun dari blok-blok geometri, *terrain* yang *irregular*, variabel entiti, sebaik *item* spesial lain seperti cahaya, suara dan jalur aktor. Pola gambar, yang disebut tekstur, diletakkan pada setiap permukaan dari blok. Blok ini dapat bermacam-macam bentuk dan materialnya memiliki properti tertentu seperti cair atau padat. Entiti sendiri dapat berupa *sprite* sederhana, model animasi, atau sub-level. Segala sesuatu yang bergerak dalam level seperti pintu animasi atau monster berada dalam kontrol sebuah program atau *script*. *Script* juga bertanggung jawab

terhadap *user interface* dan terhadap spesial efek seperti cahaya yang dinamis, nyala api, partikel, kabut, atau bayangan [6].

2.4.1 Sejarah

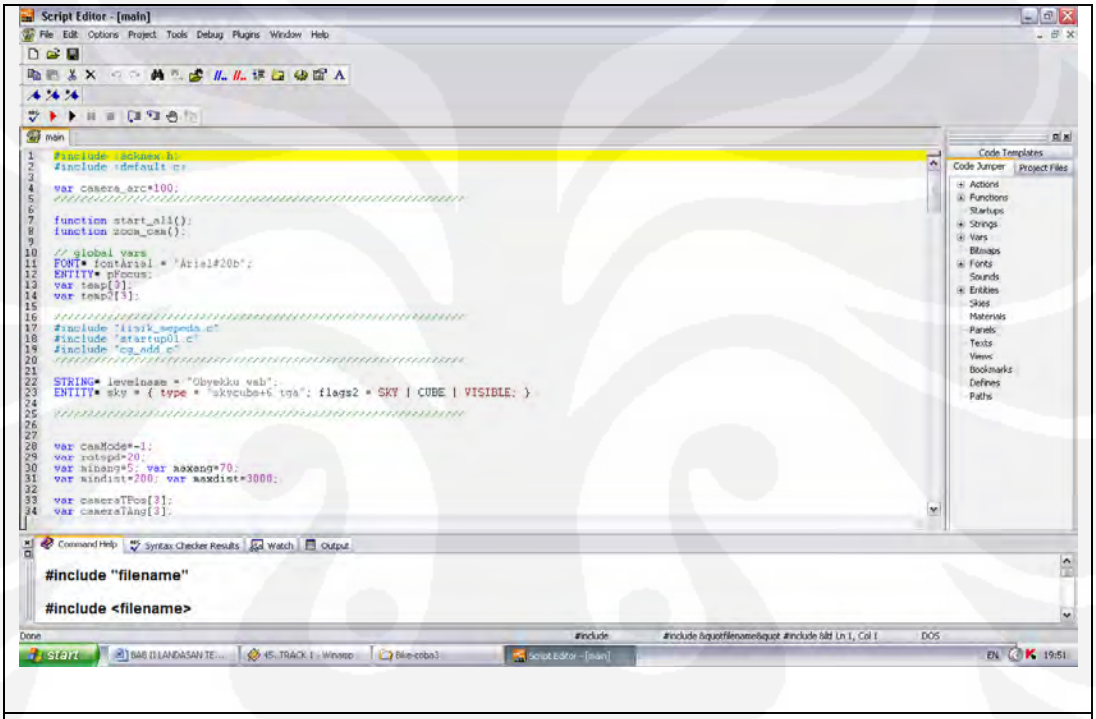
3D GameStudio ini pertama kali dibuat tahun 1993 oleh Lary Myers dengan perusahaannya bernama ACK (Animation Construction Kit) 3D.Awalnya produk ini bernama Wolfenstein. Kemudian tahun 1994 dikembangkan versi lanjutan dari ACK disebut ACK NEXT GENERATION dan penggagasnya adalah Johann Christian Lotter dari OP Group. Kemudian 1995 ACKNEX-2 ditulis untuk pertunjukan X-Base di TV Jerman. ACKNEX-2 kemudian dimiliki oleh Conitec dan dirilis dengan nama "3D GameStudio". Tahun 1997 ACKNEX-3 dirilis, kemudian tahun 1999 A4 dirilis, menyusul A5 dirilis pada tahun 2000, A6 pada tahun 2003, dan A7 pada tahun 2007. Per oktober 2007 versinya adalah 7.06.1, dan per Januari 2008 versinya adalah 7.07. GameStudio A7 memiliki 2 makna yang berbeda. Kata Gamestudio mengacu pada editor dan sistem *template game,* tetapi bagian A7 mengacu pada *game engine* [5].

2.4.2 Script Editor (SED)

Gamestudio ini adalah the *game authoring system* dan menggunakan Lite-C sebagai bahasa pemogramannya.Penjelasan lebih detail tentang Lite-C ini akan dijelaskan pada bagian berikutnya. Banyak dari sistem 3D menggunakan bahasa *scripting* untuk mengontrol obyek atau aktor. Semakin banyak sesuatu yang bergerak dalam sebuah *game*, semakin banyak instruksi *script* yang harus dieksekusi setiap detik. Kebanyakan bahasa *script* terinterpretasi yang artinya bahwa instruksi awalnya diterjemahkan dulu ke kode byte yang intermedier. Prosesor menginterpretasi kode byte per byte pada saat *run*, yang menyebabkan eksekusi yang lambat dan berpengaruh pada *frame rate*. Sebuah *script compiler* menerjemahkan bahasa tidak ke dalam kode byte tapi ke dalam kode atau bahasa mesin yang nyata, yaitu bahasa asli dari prosesor. *Script* yang dicompile lebih cepat 10 kali dari *script* yang dinterpretasi dan tidak mempengaruhi *frame rate* walaupun dalam *game* besar dengan ratusan pergerakan objek yang simultan atau bergerak terus menerus. Yang perlu diperhatikan saat memilih aplikasi, beberapa

sistem memiliki daftar "*compiler*" pada daftar fiturnya, tapi kata *tersebut* hanya berarti kompilasi ke sebuah kode *byte* intermedier, yang artinya sistem ini menggunakan bahasa interpretasi [4].

Editor script ini digunakan untuk membuat, mengedit, dan men-*debug script*, atau untuk mengedit file teks yang lain. Untuk membantu, SED ini menyediakan *syntax* yang di-*highlight*, penyelesaian kode, dan fitur-fitur lain. Berikut adalah gambar dari SED :



Gambar 2.2 Tampilan SED

Sebelah kanan, kita dapat melihat daftar variabel, elemen, dan kode *template*. Tentunya, window ini dapat dipindahkan ke sebelah kiri jika memang diinginkan. Dengan adanya *Code Jumper* kita dapat langsung melompat diantara *function*, bitmap, entiti, dan lain-lain. *Code template* berguna untuk meminimalkan penulisan kode; dan dengannya dapat dibuat *template* sendiri untuk digunakan di SED. Pada bagian bawah, terdapat *Command Help*, yang menyediakan semua bantuan saat sebuah kode dituliskan. Di bagian bawah juga terdapat *Syntax Checker Result* yang merupakan tempat dimana hasil dari *Syntax Check* berada. Jika *project* akan di-*debug* maka variabel, entiti , dan sebagainya

dapat dilihat di *Watch Tab*. Dan pada bagian atasnya, ada *button* untuk *commenting, indenting, test run,* dan sebagainya. Jika tidak yakin dengan fungsi *button* tersebut anda bisa mendekatkan *cursor* sehingga petunjuknya akan muncul [6].

2.4.2.1 Lite-C

Lite-C adalah bahasa pemograman yang didedikasikan untuk membuat aplikasi multimedia dan game komputer, dengan sebuah syntax yang hampir sama dengan bahasa pemograman C. Perbedaan utama dengan C adalah adanya *native support* untuk multimedia seperti suara, gambar, film, elemen *user interface*, model 2D dan 3D, *terrain* (hamparan tanah untuk membuat suatu lingkungan), *level game*, *collision detection* (algoritma deteksi tabrakan), dan *rigid body physics* (untuk membuat bentuk fisik tubuh). Bahasa ini dibuat untuk mencapai hasil yang cepat dengan hanya menggunakan beberapa baris bahasa Lite-C saja [7].

Sebelumnya dikenal istilah C-Script, C-Script ini adalah bahasa *script* standar dari Gamestudio sampai tahun 2007. Lite-C adalah bahasa pemograman baru yang ada dari tahun 2007 sampai sekarang. Lite-C ini sekilas hampir sama dengan C-Script, tapi kalau lebih dicermati bahasa ini ternyata dapat memberikan kelebihan lain. Sebagai pengetahuan, Lite-C adalah satu-satunya bahasa *script* di dunia ini yang memperbolehkan akses penuh ke fungsi API Windows, DirectX, dan fungsi OpenGl. Untuk tujuan ini, maka Lite-C tidak hanya mensupport syntax C tetapi juga beberapa fitur C++ seperti metode-metode dan *overloading function* [6]. Selain itu, karena Lite-C menawarkan pencapaian yang cepat untuk pemula, maka sebenarnya tujuan lain bahasa ini adalah mengajarkan tentang dasar pemograman. Untuk hal ini, Lite-C berisi tutorial yang praktis untuk orang yang bukan *programmer* game atau *programmer* aplikasi multimedia yang biasa berhubungan dengan bahasa berorientasi obyek [7].

Untuk berbagai alasan, bahasa C adalah yang terbaik untuk programming game. Bahasa C itu jelas, pendek, dan mudah untuk dipahami dibanding bahasa BASIC, dan dapat di-*run* lebih cepat dari bahasa *interpreted scripting* lain seperti

LUA atau PYTHON. Sehubungan dengan support C++ , seperti disebutkan diatas karena Lite-C memiliki akses langsung ke fungsi DirextX dan Windows API sehingga Lite-C memungkinkan efek-efek programming yang susah dilakukan bahasa sripting yang lain. Sekarang ini C/C++ digunakan untuk hampir semua *game* komersil. Walaupun ada *game* memakai skripting LUA atau PYTHON, tetapi tetap menggunakan C/C++ untuk semua bagian yang kompleks atau *time critical*, seperti rendering, fisik, dan efek. Kelebihan lain, *library* grafik dan windows memiliki interface berbasis C/C++, selain itu C juga adalah bahasa yang digunakan untuk efek *shader*. Sehingga tentunya dengan mengetahui C maka sangat esensial untuk kita menangani semua pekerjaan pemograman seperti di industri game [4].

Seperti telah disebutkan sebelumnya, Lite-C mendukung windows API dan *Component Object Model* (COM), karena itu program OpenGL dan Direxť X dapat secara langsung ditulis di Lite-C. DirectX *renderer* adalah salah satu metode untuk mengakses hardware 3D, renderer disini maksudnya adalah bagian inti dari 3D engine yang secara aktual menggambar objek 3D pada layar. DirextX renderer menggunakan Microsoft's DirectX *library*, yang terintegrasi pada Windows. Lainnya halnya dengan metode lain yaitu OpenGL *renderer* yang menggunakan perangkat keras 3D melalui OpenGl graphics *library*, yang tersedia untuk sistem Linux and Mac. Pada *3D card* yang lama, OpenGl often merender sedikit lebih cepat, sementara pada 3D *card* modern DirectX menawarkan fitur dan performansi yang lebih baik [4].

Lebih jelasnya, Lite-C ini memiliki fitur-fitur sebagai berikut [8] :

- Merupakan pengembangan lanjut *syntax* C yang cukup mudah dan memiliki *multitasking* yang transparan
- *Compile* langsung ke bahasa mesin
- Integrasi yang mudah antara API eksternal dengan interface DLL dan COM
- Memiliki *engine rendering* ABT (Adaptive Binary Tree) yang *powerful*
- 2D sprite dan model 3D, disertai animasi *bones* dan *vertex*

- Dapat diprogram untuk membuat efek partikel
- Memiliki *rigid body physics* dan *engine* untuk tabrakan
- Fungsi vektor, matriks, fisik, dan tabrakan sudah terintegrasi didalamnya
- Graphical User Interface dengan *button, slider, elemen, bitmap* untuk *font*
- Mendukung semua fungsi DiretX 9
- Memiliki fungsi untuk suara, musik, dan file film dan *track* CD
- *Editor script* dengan *syntax highlight* dilengkapi dengan *debugger single-step*
- *Tool* ini dapat mendukung format file seperti : FBX, 3DS, X, OBJ, ASE, MAP, MDL, MD2, FX, BMP, PCX, TGA, JPG, DDS, WAD, MID, WAV, OGG, MP3, MPG, AVI. Selain itu format yang dibuat melalui filter import dari 3rd party .

Saat ini versi Gamestudio yang ada dibedakan menjadi 4 yaitu Lite-C, Extra, Commercial, Pro.

Berikut adalah beberapa perbandingan antara keempatnya :

Tabel 2.2 Perbandingan 4 Edisi 3D GameStudio [9]

Edisi	**Lite-C free**	**Lite C full**	**Extra**	**Commercial**	**Pro**
Editor level	tidak	tidak	ya	Ya	distributable
Physics engine	*rigid*	*rigid*	*rigid*	*Rigid*	*Rigid+fluid*
Terrain	Ya	ya	ya	Sedikit *(chunked)*	Sedikit *(chunked)+LOD*
Efek *Shader*	tidak	tidak	tidak	Ya	ya
Render-ke-Tekstur	tidak	tidak	tidak	Ya	ya
Compile-ke-EXE	tidak	ya	ya	Ya	ya
Animasi Rangka	Ya	ya	ya	Ya	Ya+lebih berbobot

Template Game	tidak	tidak	ya	Ya	ya
Kumpulan Artwork	tidak	ya	ya	Ya	ya
Akses Forum Beta	tidak	tidak	tidak	Tidak	ya
Kompetebilitas C-Script	tidak	tidak	ya	Ya	ya

Untuk perbandingan lite-c dengan fitur-fitur A7 dapat dilihat pada tabel di bawah ini :

Tabel 2.3 Perbandingan Fitur-fitur A7 [9]

Fitur	**lite-C free**	A7 Extra	A7 Commercial	A7 Pro
Bahasa	lite-C	lite-C+C-Script	lite-C+C-Script	lite-C+C-Script
*Physics engine**	*full*	*full*	*full*	*full*
Improved shadows	ya	ya	Ya	Ya
New scene manager **	ABT	ABT	ABT	ABT+BSP
Improved terrain	ya	Ya	Sedikit (*chunked*)	Sedikit (*chunked*)+LOD
Resolusi Layar	Tak terbatas	Tak terbatas	Tak terbatas	Tak terbatas
Compile-ke-EXE	tidak	Ya	Ya	Ya
Render-ke-tekstur	tidak	Tidak	Ya	Ya
WED/MED FBX import	ya	Ya	Ya	Ya
*Shader library***	tidak	Tidak	Ya	Ya

Keterangan :

* *Physics Engine* mengkalkulasi pergerakan, rotasi, dan respon kolisi dari objek yang kaku (*rigid body object*) dengan mengaplikasikan fisik yang real terhadapnya. Sebenarnya, penggunaan *physics engine* tidak diperlukan secara absolut terhadap semua aplikasi atau game, fisik sederhana seperti "Newtonian", untuk akselerasi dan deselerasi, dapat juga diprogram atau dituliskan dalam *script*. *Physis engine* menggunakan properti objek seperti momentum, *torque* atau elastisitas untuk mensimulasikan tingkah laku atau *behavior*. Hal ini tidak saja memberi hasil yang lebih realisitis, tapi lebih mudah ditangani daripada menuliskan tingkah laku atau *behaviour* ke dalam script. *Physics engine* membolehkan konstruksi kompleks dari alat-alat mekanik dan beberapa juga mendukung bentuk fisik yang *rigid*, seperti air [9].

** Sebuah *scene manajer* memperbaiki *frame rate* dengan merendering hanya bagian yang terlihat pada sebuah level *game*. Ada beberapa metode pada *scene manajer* ini, diantaranya :

a. *Octree* yaitu sistem yang membagi level ke dalam 3 daerah kubik dan hanya merender isi dari bagian yang berada dalam *view* kamera, terkait kepraktisan ini Octress masih digunakan oleh beberapa 3D *engine* saat ini.
b. *ABT (Adaptive binary tree)* adalah sistem yang membagi level ke dalam area segiempat tergantung pada ukuran level dan isinya.
c. *BSP tree* adalah sistem yang membagi level ke dalam area yang tidak beraturan (*irregular*) dan hanya merender bagian yang benar-benar terlihat. Ini adalah system yang cepat dan lebih efektif terutama untuk level *indoor*. Dengan BSP, kecepatan rendering *indoor* tidak tergantung pada ukuran level. Sehingga hal ini membentuk *frame rate* yang pas walaupun pada PC tua Tetapi kekurangannya sistem ini harus di pre-kalkulasi dulu oleh *map compiler* pada level editor.

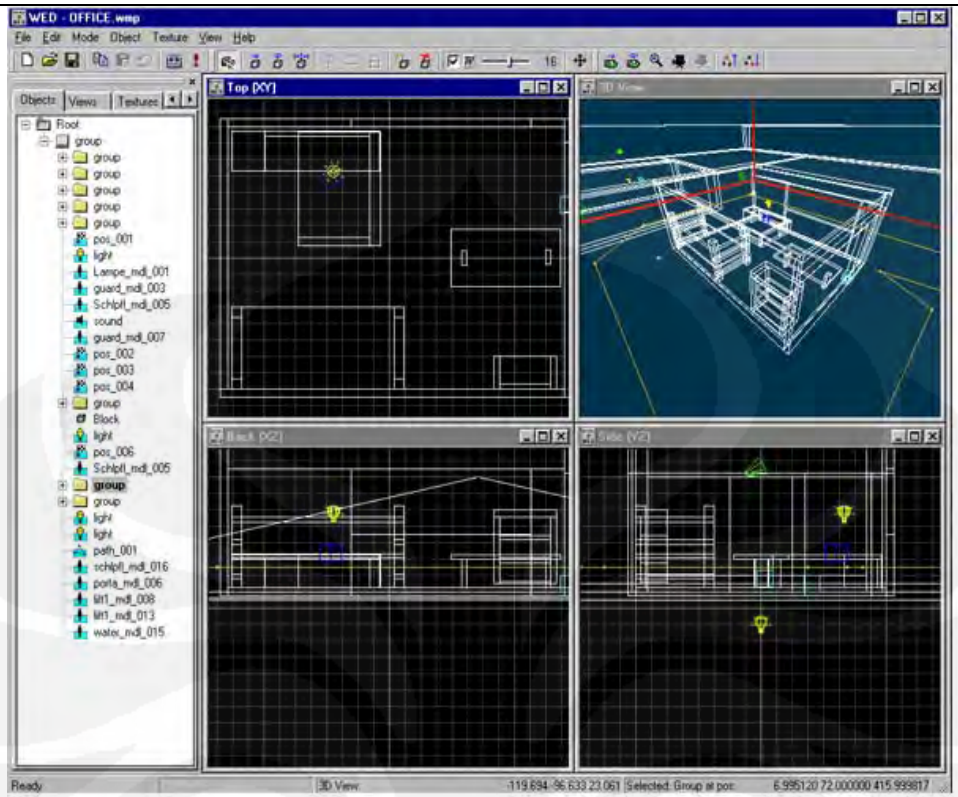
Engine komersil yang *high end* saat ini normalnya menggunakan *BSP tree*, sementara *engine* yang sederhana kebanyakan menggunakan ABT, Octree

atau sistem yang lain. Untuk level *outdoor*, dimana BSP tree kurang bagus dibanding ABT dan Octree, maka digunakan sistem LOD (Level Of Detail) untuk menjaga frame rate yang tinggi. Sistem ini secara otomatis berpindah ke bentuk objek yang lebih sederhana ketika objek jauh dari kamera, yang membuat pengurangan jumlah keseluruhan polygon per frame [9].

*** *Shader* menambahkan dimensi baru pada rendering grafik. Hal ini memperbolehkan fungsi transformasi, pencahayaan, dan rendering untuk dimodifikasi pada saat *run* berbasis pada sebuah vertex dan piksel. Sebuah *shader* adalah *script* kecil yang *run* dalam *hardware* grafik untuk setiap verteks atau piksel yang direnderpada layar / *screen* [9].

2.4.3 World Editor (WED)

World editor atau WED adalah *editor* untuk membentuk dunia virtual atau yang disebut level dalam sebuah aplikasi *game*[6]. Dengan WED, dapat diposisikan berbagai macam obyek, men-*assign action* ke dalam model (yang juga dikenal sebagai entiti) yang mana didefinisikan melalui *script*, men-*assign* tekstur ke level geometri, dan membangun level anda menggunakan teknik ABT atau BSP tree (khususnya untuk A7 pro). Karena WED adalah *editor* utama dari Gamestudio, maka WED adalah tempat untuk menggabungkan semua bagian dari *game* atau aplikasi yang akan dibuat (program, grafik 3d, dan *level*), sehingga dengan WED sebuah game dapat dijalankan [5]. Berikut adalah gambar dari tampilan WED :



Gambar 2.3 Tampilan WED

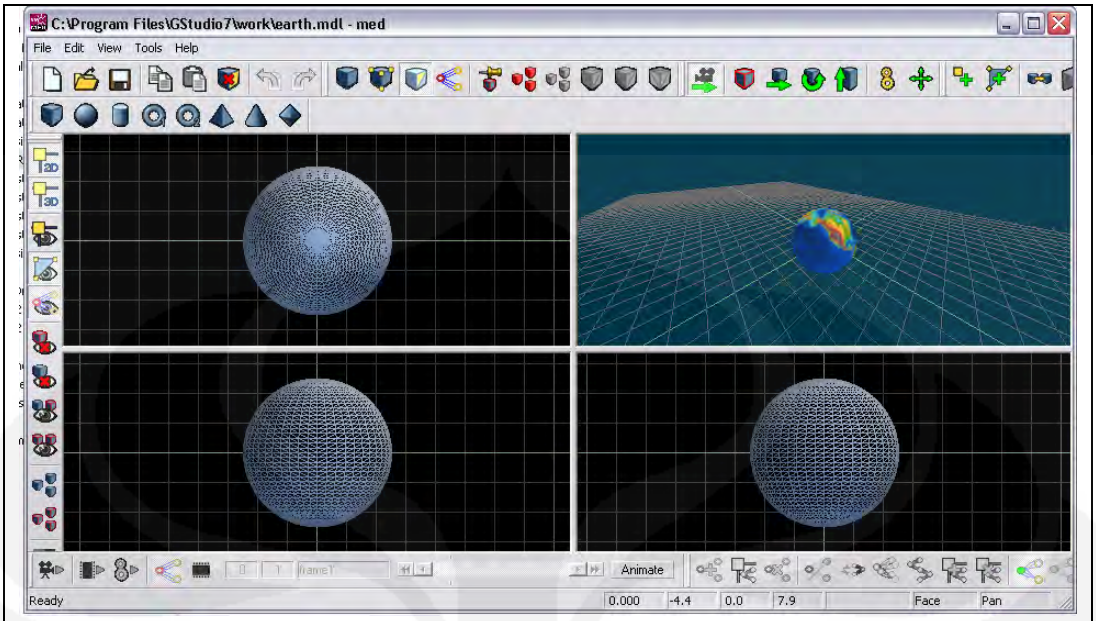
Layout atau tata letak dari WED cukup sederhana. Bagian utama, yaitu bagian pusat di kanan , adalah tempat pengeditan banyak dilakukan. *Editor* ini terdiri dari 3 grafik dan sebuah view 3D. Window kiri atas adalah *top view*, yang memiliki koordinat X dan Y. Window kanan bawah adalah *side view* atau koordinat Y dan Z. Window kiri bawah adalah *back view* atau koordinat X dan Z. Window kanan atas adalah *3D view*, yang memberikan view awal dari level anda sebelum di-*build* dan di-*run*. Grafik dibagi dalam ukuran dengan perkalian 128 dan lebih jauh dibagi dalam 16 untuk membantu dengan adanya *spacing*. Grafik akan otomatis me-*resize* ketika sebuah *bit* di-*zoom-out* dan berada dalam ukuran perkalian 8 (1024 dan 128 kemudian 8192 and 1024). Bagian di sebelah kiri mendaftar obyek dalam level anda, tekstur, dan beberapa hal lain. Untuk efek yang baik sebuah tekstur harus perkalian ukuran genap yang berpangkat dua (seperti 256x128, 1024x256, atau 64x64). Sesuatu yang ganjil misalnya 394x213 atau 723x1280 akan terlihat kurang baik dan lambat pada rendering [5].

Bagian atas dari WED adalah tool bar untuk memanipulasi objek, menambahkan objek baru (seperti entiti, suara, dan cahaya), mem-*build* level, me-*run* level, dan beberapa pilihan lain. Ketika kita mengklik kanan sesuatu atau memilih properti, kita dapat secara manual memasukkan posisi, men-*assign* sebuah *action* terhadap entiti, atau menyesuaikan tekstur pada setiap sisi dari blok.

Kekurangan WED adalah ketidakmampuannya untuk melakukan segala sesuatu tanpa block atau terrain. Sebagai contoh susah untuk mem-*build* sebuah jalan , karena hampir tidak mungkin untuk memindahkan blok ke dalam posisi yang sempurna, jadi tidak ada ”langkah” atau celah/pemisah pada jalan (tidak mungkin untuk memperoleh batas atau edge dari blok secara bersamaan dan sempurna, fitur *snap to grid* yang ada membantu hanya ketika menggunakan blok non-rotasi yang berdekatan). Segitiga atau fitur *snap-block-edge-to-block-edge* akan memperbaiki hal itu dan akan membuat kemungkinan untuk membuat lantai yang lebih kompleks lebih dari satu blok tanpa celah atau step.

2.4.4 Model Editor (MED)

Model editor atau MED menyediakan kapabilitas untuk mendesain model dan kadang kala digunakan untuk membuat level. Model dapat dibuat dari bentuk mulai dari kotak sederhana sampai ke model manusia atau lingkungan yang kompleks seperti sebuah kota. Model dibuat dari *meshes*, yaitu grup dari segitiga yang sering disebut *polygon*, yang diletakkan bersama-sama untuk membentuk sebuah figur; tulang untuk animasi; satu atau beberapa tekstur untuk kulit; dan file efek (.fx) untuk *shader* [5]. Berikut adalah tampilan MED :



Gambar 2.4 Tampilan MED

Seperti WED, layout MED juga ada 4 window. Tapi , untuk defaultnya, MED tidak memiliki *grid*, namun tetap bisa dikonfigurasikan untuk *grid*. Selain MED hampir sama dengan WED dalam hal layout, MED memiliki tambahan lain yaitu editor skin, yang memperbolehkan model untuk ditekstur. Editor skin memiliki layout yang komplit. Tekstur ditunjukkan pada sisi kiri dan model disebelah kanan dengan dikelilingi toolbar. Tekstur biasanya dibuat dalam editor grafik eksternal dan diimpor dari file gambar bmp, tga, atau pcx [5].

Bentuk model dapat dibuat baik melalui bentuk primitif yang sudah disediakan seperti kubus atau piramid atau juga dengan membentuk kumpulan verteks dan membentuk sisi dari model tersebut. Model tidak harus mengikuti batasan-batasan yang dimiliki BSP, sepanjang model dapat direndering dengan cepat, maka akan membuat model tersebut menjadi pilihan terbaik untuk desain level.

2.4.4.1 Obyek 3D

Obyek3-D adalah sekumpulan titik-titik 3D(x,y,z) yang membentuk luasan-luasan (*face)* yang digabungkan menjadi satu kesatuan. *Face* ini adalah gabungan titik-titik yang membentuk luasan tertentu atau sering dinamakan dengan sisi [10].

Sebuah *game engine* dapat merender beberapa tipe dari obyek 3D yang terpisah, yang disebut entiti, pada layar. Entiti ini dapat dibuat menggunakan *script* atau ditempatkan dalam level oleh WED. Entiti-entiti ini tidak disimpan dalam *file level,* tetapi merupakan *file* eksternal yang dibaca dari *folder* kerja. Berikut adalah beberapa tipe dari entiti :

a. Model

Sebuah model adalah adalah sebuah obyek 3D animasi, yang disimpan pada *file* eksternal MDL. Model ini terdiri dari sebuah *mesh* 3D dengan tulang atau sendinya dan sebuah kulit pelapis dari model tersebut. Entiti model dapat memiliki bayangan dan normalnya digunakan untuk obyek bergerak atau obyek animasi, misalnya model aktor, monster, ataupun yang setipe itu. Model dibentuk menggunakan program *editor* model, seperti MED, atau dapat diimpor dari bentuk format file 3D lain seperti 3DS, X, OBJ, ASE, atau MD2. Eksternal *editor* seperti Truespace®, gameSpace®, Maya® atau 3D Studio MAX® dapat secara langsung menyimpan modelnya dalam format Gamestudio. Bagian yang berbeda dari *mesh* model dapat memiliki tekstur yang berbeda, properti materi, dan *shader* yang berbeda [6].

b. *Sprite*

Sebuah *sprite* yang juga disebut *billboard* adalah sebuah obyek 2D yang datar yang dapat digunakan untuk beberapa tujuan. Obyek ini bisa ditempatkan pada dinding atau lantai sebagai sebuah dekorasi, dapat juga berdiri tegak dalam sebuah daerah sebagai sebuah *billboard* atau papan spanduk, atau dapat berperilaku sebagai sebagai obyek semi 3D dengan

selalu menghadap ke kamera. *Sprite* disimpan dalam *file* eksternal seperti PCX, BMP, TGA, atau DDS dan dapat dibuat menggunakan program gambar standar seperti Gimp, PaintShop Pro® or Adobe Photoshop®. *File* TGA atau DDS dapat berisi sebuah *channel alpha* yang memberikan sebuah nilai transparan untuk setiap piksel. *File* PCX, BMP, atau TGA dapat dianimasi. *File* DDS dapat berisi beberapa *mipmaps* untuk mendapatkan kualitas yang baik dan rendering yang cepat. Entiti *sprite* direnderlebih cepat daripada entiti model atau *map*, dan obyek ini bisa digunakan untuk ledakan, lampu, pohon, rumput atau semacamnya.

c. *Sublevel (Map Entities)*

Sebuah entiti *map* adalah hasil *compile level* yang sederhana yang disimpan dalam sebuah *file* eksternal WMB. Entiti ini dapat digunakan untuk bagian *level* yang bergerak keseluruhan seperti pintu, *platform*, atau kendaraan. Karena ini merupakan *file* terkompilasi, maka entiti ini dapat dibuat di WED. Tekstur dan bayangan dari level dan dari *map* entiti dialokasikan awal dalam memori video pada saat *loading*, untuk menjaga kemulusan grafik. Tekstur dari semua tipe entiti hanya dialokasikan ketika entiti terlihat [6].

d. *Terrain*

Terrain terdiri dari satu atau beberapa tekstur yang dimap pada *grid* segiempat. Entity ini disimpan dalam sebuah *file* HMP eksternal. Entiti *terrain* dapat digunakan sebagai bagian dari level. Entiti ini dapat dibuat di MED atau diimpor dari *map* RAW atau BMP atau PCX yang dibuat dengan program pembangun *terrain* [6]. Engine di Gamestudio mendukung dua tipe terrain yaitu unchunked atau chunked. Terrain yang unchuncked direnderseperti sebuah model dan tidak boleh lebih dari 128x128 verteks. Algoritma rendering terrain lebih baik untuk terrain yang chunked, pada saat dipanggil tipe ini dibagi-bagi ke dalam *chunked* persegi yang disimpan dalam sebuah *cache*, terpisah ketika berada diluar *view*

frustrum dan direnderer pada langkah resolusi berbeda tergantung pada jaraknya dari kamera dan setting *terrain_lod.*

Perbedaan obyek-obyek ini dapat terlihat dari tabel berikut.

Tabel 2.4 Perbedaan Entiti [6]

Tipe Entiti	**Model**	***Sprite (image)***	***Map***	***Unchunked Terrain****	***Chunked Terrain****
Kebanyakan digunakan untuk	Aktor, kendaraan	Tumbuhan, pohon, dekorasi, efek	Bangunan, *platforms*, pintu, *trains*	Landscape outdoor	Landscape outdoor
Format Import	3DS, X, OBJ, ASE, MDL, MD2	BMP, PCX, TGA, DDS	MAP, WMP	BMP, PCX, RAW, HMP	BMP, PCX, RAW, HMP
Dibuat dengan	MED atau editor model eksternal	Program Paint	*Map editor (WED)*	*Terrain generator, paint program, MED*	*Terrain generator, paint program, MED*
Polygon	~10000	1	~1000	~10000	~1000000
Ukuran	Kecil	Kecil	Menengaj	Besar *(Big)*	Besar (*Huge)*
Animasi	*Vertex, bones, texture, shader*	*Texture, shader*	*Texture, shader*	*Shader, deformation*	*Shader*
Bentuk kolisi	*Bounding box, ellipsoid, polygonal*	*Bounding box, ellipsoid, polygonal*	*Polygonal*	*Polygonal*	*Polygonal*
Pergerakan	Gerak, Rotasi, Skala	Gerak, Rotasi, Skala	Gerak, Rotasi	Tidak ada	Tidak ada
Shading	*PRV, gouraud, dynamic shadows*	*PRV*	*PRV, static shadows*	*Gouraud, shadow map*	*Gouraud, shadow map*

BAB III

PERANCANGAN

3.1 ANALISA REQUIREMENT

Sebelum membuat aplikasi tentunya kita harus mengetahui apa saja yang akan dilakukan oleh sistem kita, caranya adalah menganalisa permintaan di awal proses. Analisa awal ini dilakukan agar kita dapat mengetahui lebih jelas hal-hal yang diminta sehingga nantinya akan memudahkan perancangan dan pembuatan sistem.

3.1.1 Requirement Fungsi

Untuk tugas akhir ini akan dibuat sebuah obyek 3D menggunakan aplikasi 3D Gamestudio. Aplikasi 3D yang akan dibuat merupakan simulasi bersepeda pada jalur sepeda UI yang baru selesai dibangun. Pada tugas akhir ini, akan dibuat sebuah objek sepeda yang dapat bergerak maju, mundur, ke kanan, ke kiri, dan berhenti. Untuk simulasi ini juga akan dibuat dua perspektif kamera yaitu perspektif orang pertama (*first person*) yang merupakan perspektif langsung pada obyek dan perspektif orang ketiga (*third person*) yang merupakan perspektif di luar obyek dengan penempatan kamera yang terorbit dan yang mengikuti obyek. Obyek ini juga dilengkapi dengan respon terhadap halangan.

3.1.2 Requirement Interface

3.1.2.1 Hardware Interface

Hardware yang akan digunakan dalam sistem ini adalah :

- CPU dengan P3-500, 512 MB RAM, CD-ROM, 3D video card (32+ MB)
- *Input device* : keyboard.
- *Output device* : monitor.

3.1.2.2 Software Interface

Software yang akan digunakan dalam sistem ini adalah :

- Windows ME / 2000 / XP / Vista dan DirectX 9.0 keatas
- Gamestudio yang terdiri dari SED, MED, WED untuk membuat aplikasi 3D
- E-D driver sebagai salah satu kelengkapan untuk menggunakan kacamata Edimensional pada PC.

3.2 PERANCANGAN

3.2.1 Batasan Perancangan

Perancangan yang akan dibuat pada tugas akhir ini adalah membuat sebuah objek yang memiliki fungsi-fungsi tertentu yang nantinya dapat disatukan dengan sebuah lingkungan tertentu dalam hal ini sebagian jalur sepeda UI yang baru dibangun (pembuatan lingkungan ini terpisah dan dikerjakan oleh Citra Parameswari).

3.2.2 Perancangan Arsitektur

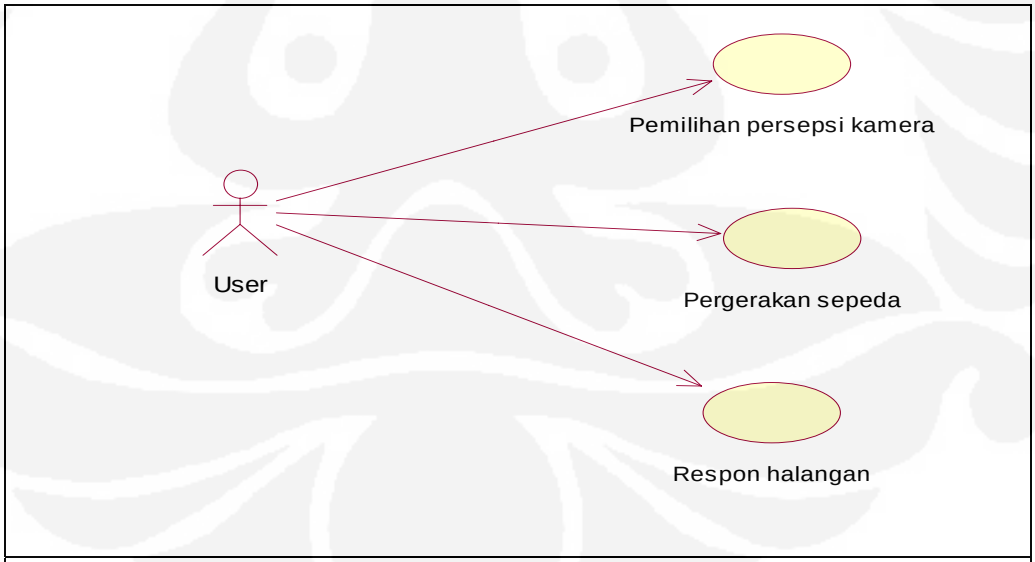
Perancangan arsitektur dalam tugas akhir ini menggunakan *tool* Rational Rose 2000. Arsitektur dari program yang dibuat akan dibuat digambarkan melalui diagram-diagram dalam pemodelan UML. Untuk memahami UML diperlukan tiga konsep dasar mengenai elemen-elemen utama, yaitu: Things, Relationships, dan Diagrams. Things adalah suatu abstraksi yang

merupakan komponen dasar dari UML. Relathionships adalah komponen yang menunjukkan hubungan antara dua buah things. Diagram adalah presentasi secara grafik dari sejumlah elemen, seringkali digambarkan sebagai grafik yang terhubung dari *node* (objek) dan relationship. Diagram digunakan untuk menggambarkan sistem dari perspektif yang berlainan, sehingga diagram dapat dikatakan sebagai proyeksi kedalam sistem. Secara teori sebuah diagram dapat berisi dari beberapa kombinasi things dan relationships. Ketiga elemen utama tersebut merupakan landasan dasar dalam UML yang terdiri dari aturan-aturan atau mekanisme yang dapat digunakan secara bersamaan.

3.2.2.1 Use Case Diagram

Use Case Diagram adalah deskripsi dari fungsionalitas yang dimiliki oleh sistem yang terdiri dari *use case-use case*, aktor dan hubungan interaksinya. *Use Case* digambarkan sebagai suatu cara pandang terhadap sistem dilihat dari perspektif aktor.

Use Case Diagram untuk aplikasi yang akan dibuat dapat terlihat dalam gambar berikut.



Gambar 3.1 *Use Case* Aplikasi

Use case diatas menggambarkan hal-hal yang terkait dengan objek 3D yang nanti akan dibuat dilihat dari perspektif aktor , yang dalam sistem ini adalah pengguna aplikasi (user).

Tabel 3.1 Penjelasan *Use Case* Pemilihan Perspektif Kamera

Nama	Pemilihan perspektif kamera
Deskripsi Singkat	Menyediakan 2 pesepsi kamera untuk melihat objek
Aktor	User
Kondisi awal	▪ Aplikasi sudah dijalankan ▪ Posisi kamera dari perspektif *first person*
Kondisi akhir	Posisi kamera akan tergantung pada pemilihan user yaitu perspektif *first person* atau *third person*

Tabel 3.2 Penjelasan *Use Case* Pergerakan Sepeda

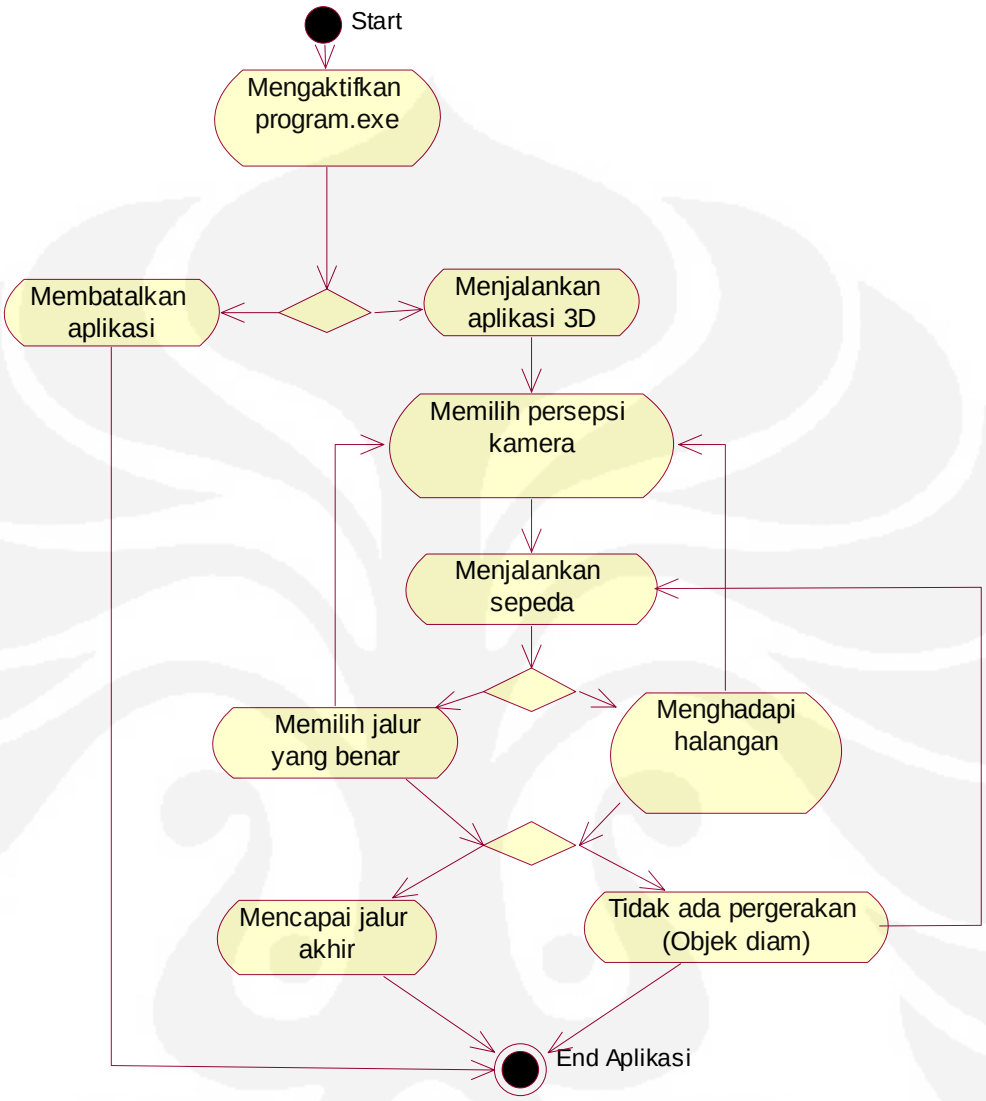
Nama	Pergerakan sepeda
Deskripsi Singkat	Pergerakan sepeda dengan menekan tombol di keyboard untuk bergerak ke kiri, ke kanan, maju, dan mundur
Aktor	User
Kondisi awal	▪ Aplikasi sudah dijalankan ▪ Posisi objek diam
Kondisi akhir	Pergerakan objek sesuai dengan pilihan user (W=maju, S=mundur, D=kanan, A=kiri, *space*=berhenti)

Tabel 3.3 Penjelasan *Use Case* Respon Halangan

Nama	Respon halangan
Deskripsi Singkat	Respon objek terhadap sebuah halangan yang ada di sekitarnya seperti dinding
Aktor	User
Kondisi awal	▪ Aplikasi sudah dijalankan ▪ Posisi objek normal
Kondisi akhir	Objek merespon halangan dengan tidak menembus halangan tersebut

3.2.2.2 Activity Diagram

Activity diagram adalah salah satu diagram dari UML untuk menggambarkan aspek dinamis dari sebuah sistem. Diagram ini sebenarnya merupakan flowchart yang menggambarkan aliran kontrol dari satu aktivitas ke aktivitas lain. *Activity diagram* untuk sistem ini dapat terlihat dalam gambar berikut.



Gambar 3.2 *Activity Diagram* Aplikasi

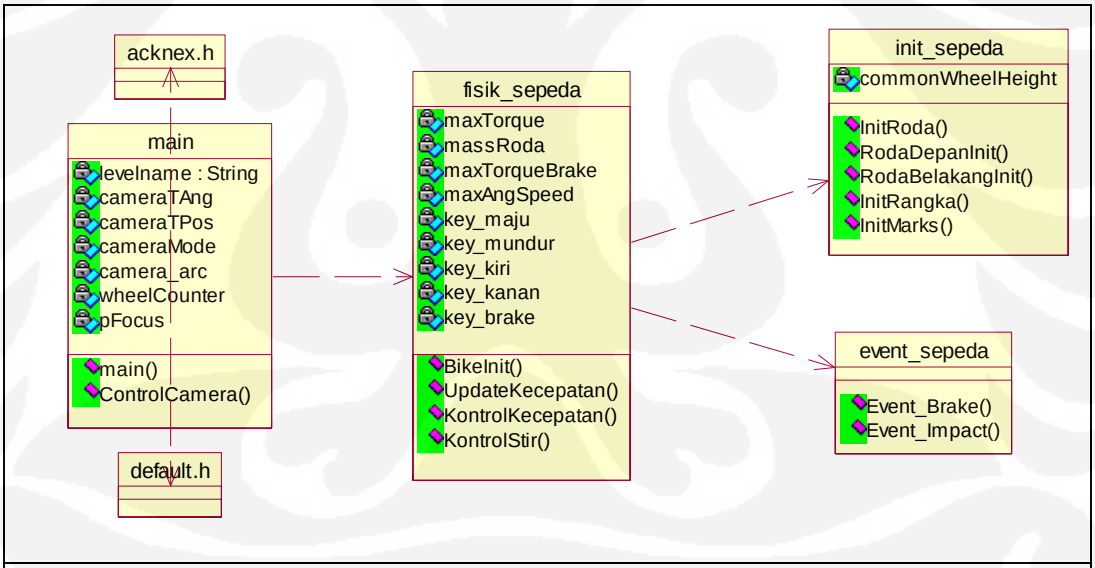
Dari *activity diagram* di atas terlihat jelas bahwa sistem yang dibuat akan diawali dengan menjalankan program yang dibuat. Dengan menjalankan program maka didapat dua pilihan yaitu meneruskan atau keluar dari program. Setelah aplikasi dijalankan dan memilih meneruskan maka objek mulai dijalankan, dengan terlebih dahulu memilih perspektif kamera (aktivitas kamera ini bisa saja dijalankan pada saat objek sudah bergerak).Setelah itu

aktivitas obyek bisa memilih bergerak pada jalur yang tepat ataupun menghadapi rintangan, hal ini dapat dilakukan sampai obyek mendekati jalur akhir dan keluar dari aplikasi. Aktifitas lain objek bisa juga diam saja tanpa melakukan aktifitas apa-apa, sampai user mau menggerakkan lagi seperti langkah sebelumnya ataupun memilih mengakhiri aplikasi.

3.2.2.3 Class Diagram

Class diagram adalah diagram yang paling banyak ditemukan dalam pemodelan sistem berorientasi obyek. Diagram ini menunjukkan keberadaan class-class dan hubungannya masing-masing pada perancangan secara logik dalam sistem dan digunakan untuk menggambarkan desain statis dari sistem..

Class diagram dari sistem yang dibuat akan diperlihatkan pada gambar berikut ini :



Gambar 3.3 *Class Diagram* Aplikasi

Penjelasan Class :

a. Acknex.h

Class ini merupakan library dari virtual engine A7.

b. Default. H

Class ini juga merupakan *library* untuk fungsi-fungsi di Lite-C

c. Main

Class ini merupakan *class* utama program tempat memanggil level (file WMB), tempat menginisialisasi fisik sepeda, tempat untuk mengontrol kamera. *Class* ini bergantung pada class lain yaitu acknex.h, default.h, dan fisik_sepeda.c

d. Fisik_sepeda

Class ini merupakan class untuk membentuk rangka atas sepeda dan mengontrol kecepatan. *Class* ini bergantung pada class init_sepeda.c dan event_sepeda.c

e. Init_sepeda

Class ini merupakan class untuk inisialisasi bagian obyek yang paling penting yaitu bagian ban obyek.

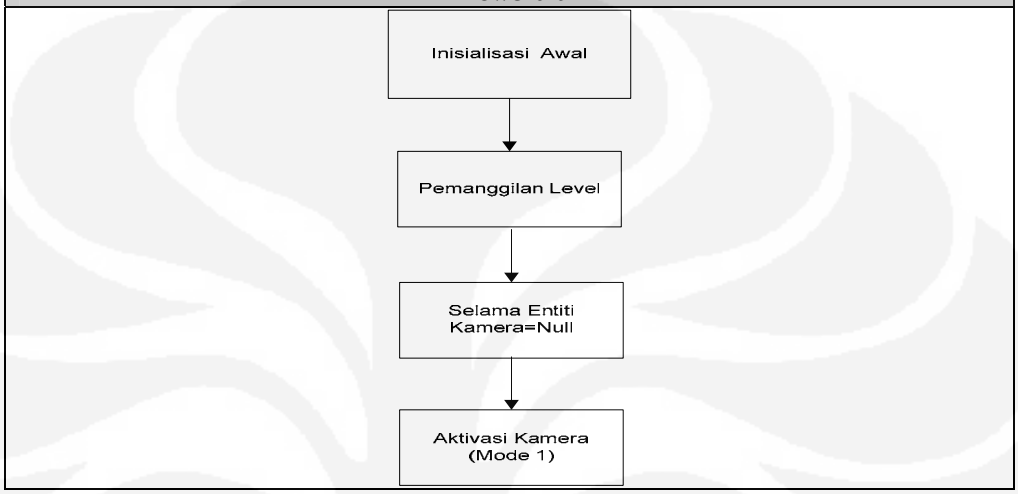
f. Event_sepeda

Class ini merupakan class yang mendeklarasikan peristiwa yang terjadi pada obyek.

Penjelasan *method-method* yang utama adalah sebagai berikut:

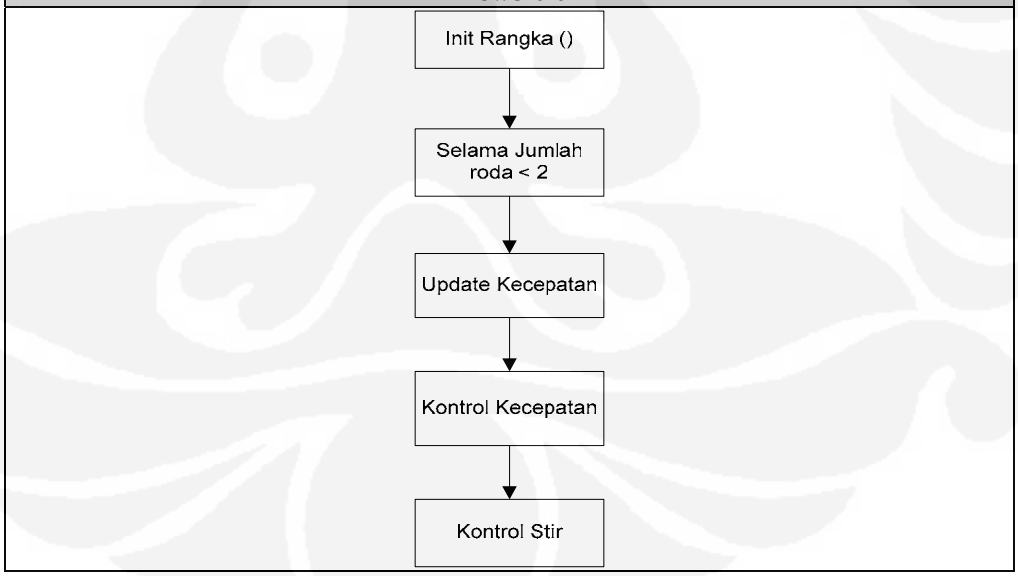
Tabel 3.4 Penjelasan *Method* Main

Nama *Class*	Main
Nama *Method*	Main()
Deskripsi	*Method* utama
Flowchart	



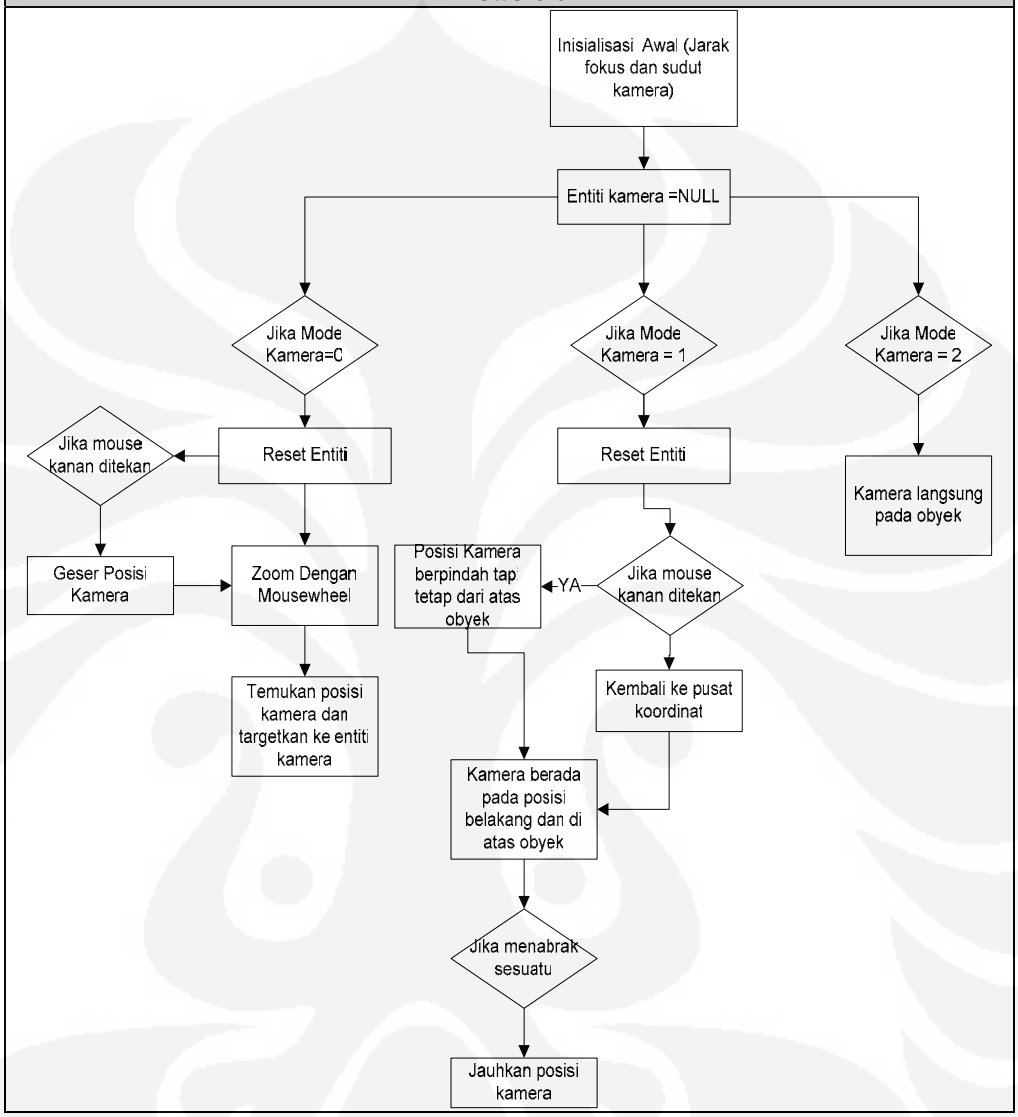
Tabel 3.5 Penjelasan *Action* BikeInit

Nama *Class*	Fisik_sepeda
Nama *Method*	BikeInit
Deskripsi	*Action* untuk rangka atas
Flowchart	



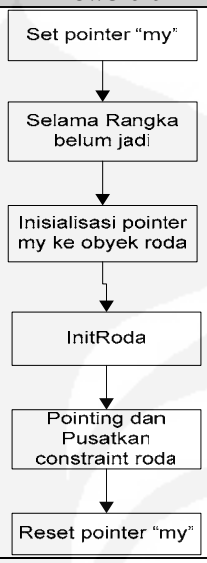
Tabel 3.6 Penjelasan *Method* Control_Camera

Nama *Class*	Main
Nama *Method*	ControCamera
Deskripsi	Fungsi untuk mengontrol kamera
Flowchart	



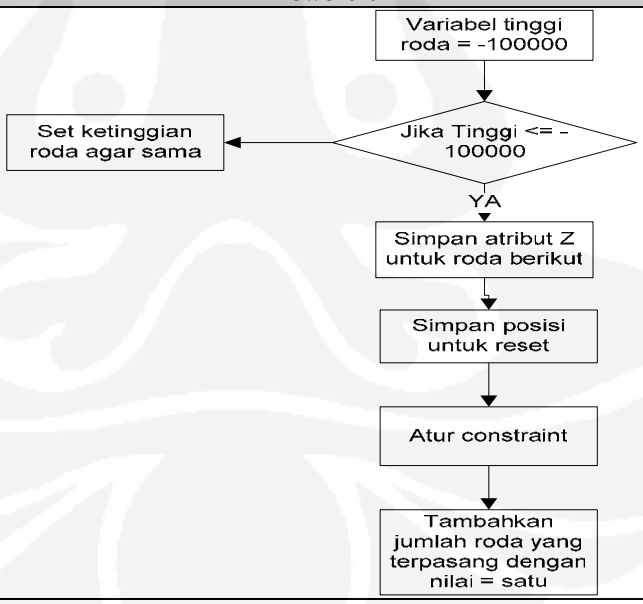
Tabel 3.7 Penjelasan *Action* Roda (RodaBelakangInit dan RodaDepanInit)

Nama *Class*	Init_physics
Nama *Method*	RodaBelakangInit dan RodaDepanInit
Deskripsi	Action untuk roda
Flowchart	



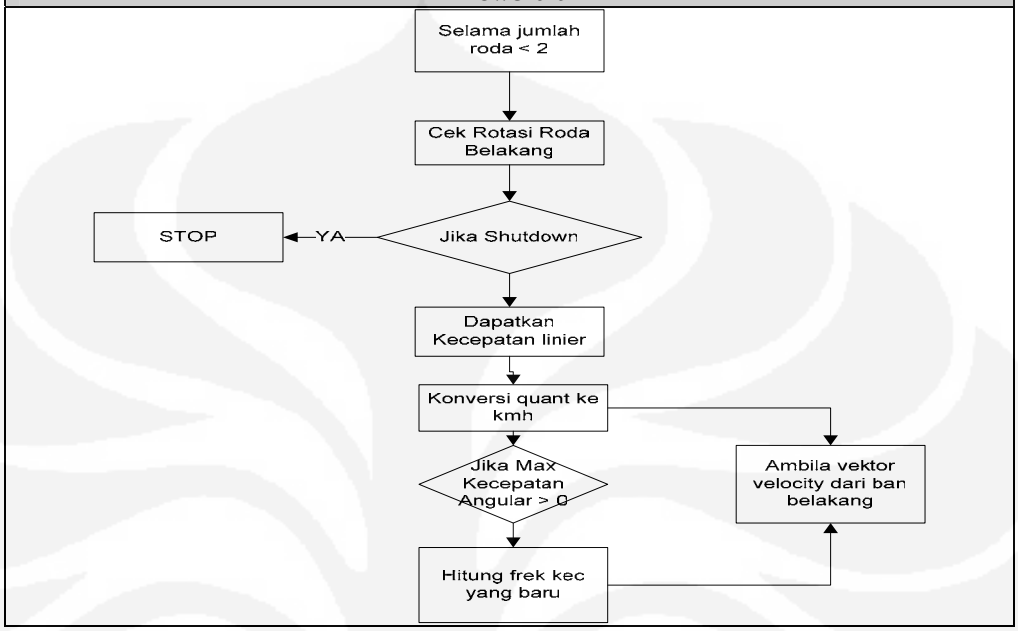
Tabel 3.8 Penjelasan *Method* InitRoda

Nama *Class*	Init_Physics
Nama *Method*	InitRoda
Deskripsi	*Method* untuk inisialisasi roda
Flowchart	



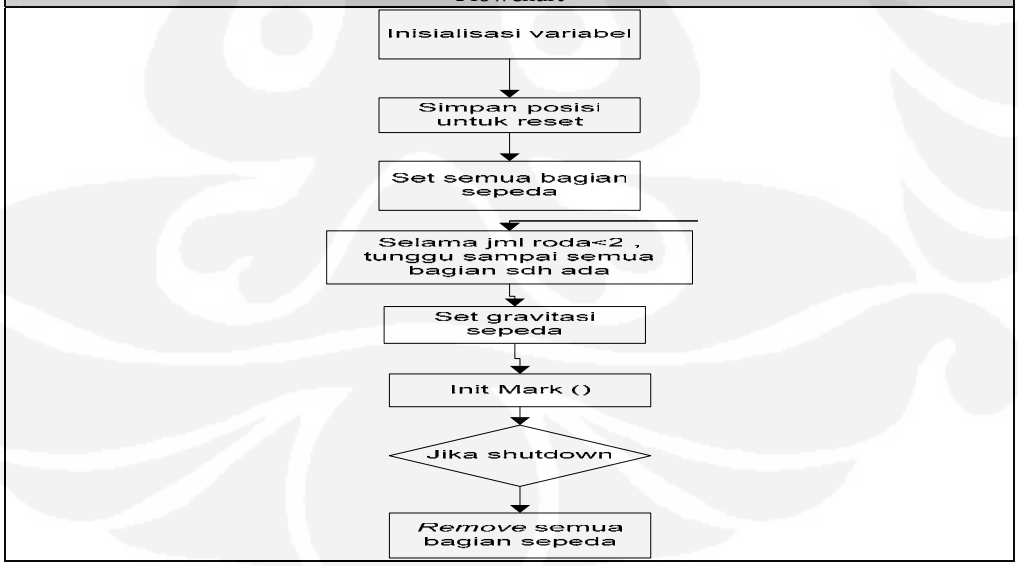
Tabel 3.9 Penjelasan *Method* UpdateKecepatan

Nama *Class*	Fisik_sepeda
Nama *Method*	UpdateKecepatan
Deskripsi	Method untuk mengupdate kecepatan
Flowchart	



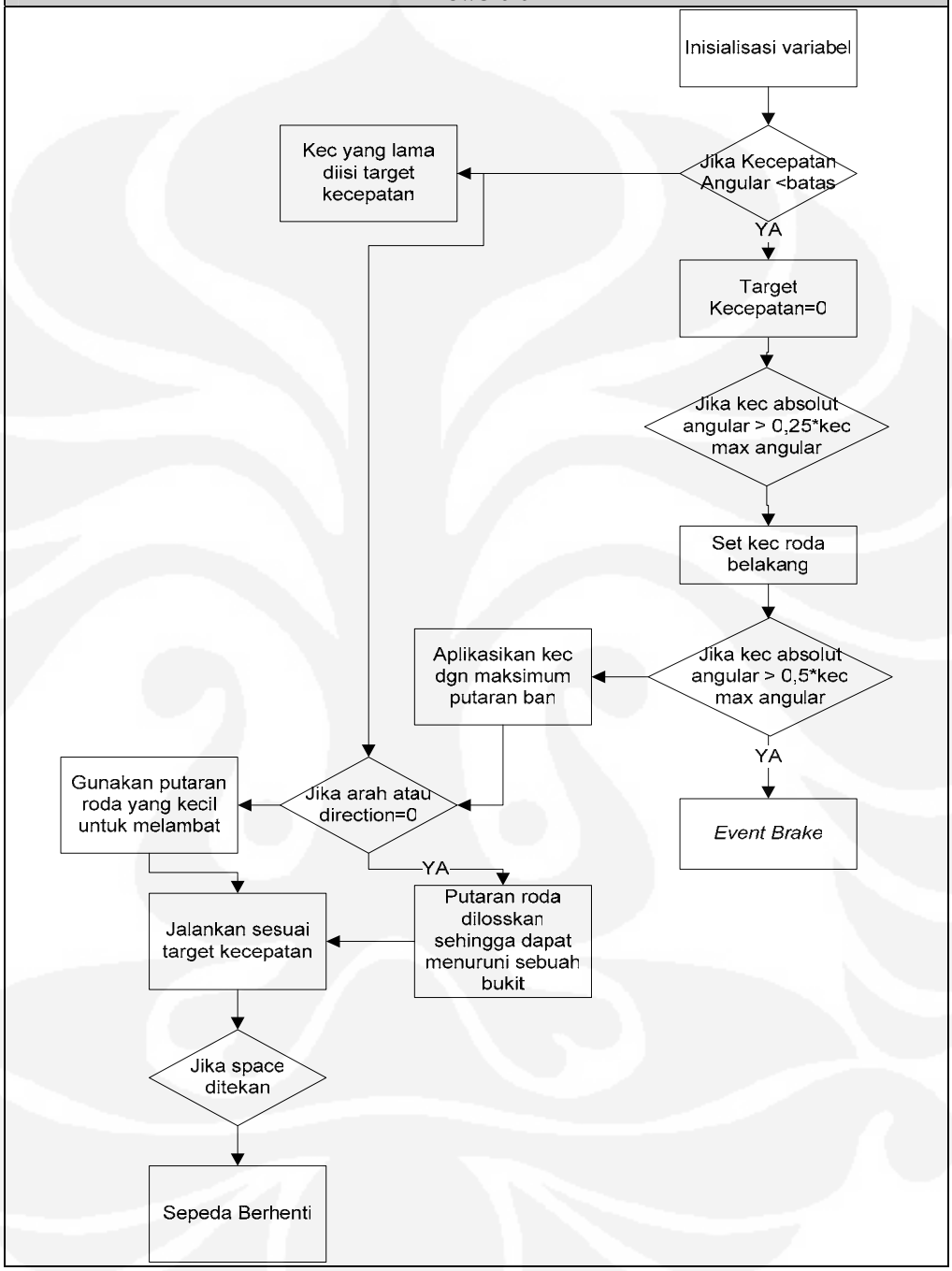
Tabel 3.10 Penjelasan *Method* InitRangka

Nama *Class*	Physics_init
Nama *Method*	InitRangka
Deskripsi	Method untuk konstruksi bagian sepeda
Flowchart	



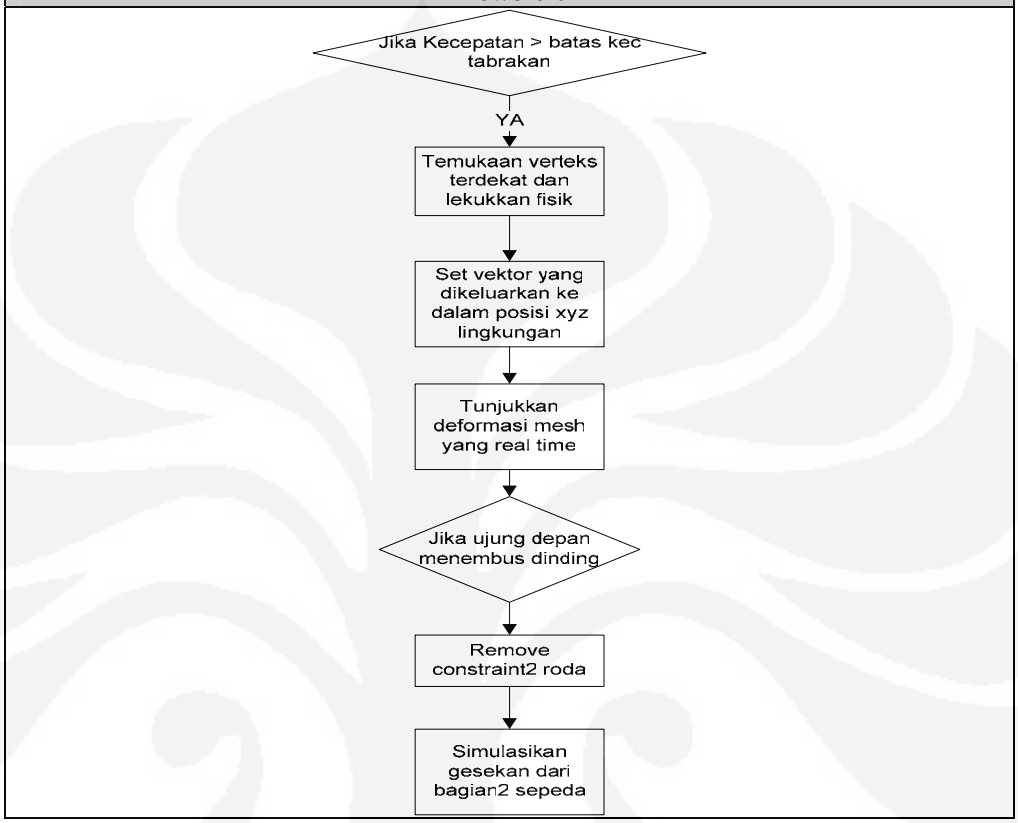
Tabel 3.11 Penjelasan *Method* KontrolKecepatan

Nama *Class*	Fisik_sepeda
Nama *Method*	KontrolKecepatan
Deskripsi	Method untuk mengontrol kecepatan
Flowchart	



Tabel 3.12 Penjelasan *Method* EventImpact

Nama *Class*	Physics_event
Nama *Method*	EventImpact
Deskripsi	Method untuk membongkar obyek
Flowchart	



Untuk melihat *method-method* ini secara detail dapat dilihat pada lampiran pada buku ini (selain *method* utama disertai juga *method* pendukung aplikasi ini).

BAB IV

PENGUJIAN DAN ANALISA

4.1 PENGUJIAN DAN ANALISA PROGRAM

Dalam melakukan pemograman dengan 3D Gamestudio, ditemui banyak hal. Pertama karena aplikasi 3D yang digunakan ini tidak begitu dikenal sehingga perlu dilakukan usaha yang lebih mendalam untuk mengetahuinya. Aplikasi 3DGS yang digunakan dalam tugas akhir ini adalah Gamestudio Extra versi 7.05 dan memiliki 3 *editor* yaitu WED, SED, MED. WED adalah editor yang mengintegrasikan semua *file* yang dibuat di 3DGS, versi yang digunakan dalam tugas akhir ini adalah WED versi 6.835. WED ini memiliki banyak fitur-fitur yang saling terkait untuk membentuk suatu program yang diinginkan. Selain pembelajaran mengenai fungsi fiturnya sendiri, keterkaitan penggunaan fitur-fitur dalam WED ini juga merupakan suatu hal yang harus dikaji lebih jauh. Salah satu ciri pengeksekusian *file* di WED adalah dengan mem-*built* keseluruhan *file* terlebih dahulu dengan tujuan mengubah *file* dengan ekstensi WMP menjadi *file* dengan ekstensi WMB, setelah itu *file* tersebut baru dapat di-*run*, sehingga langkah pengeksekusiannya memiliki urutan proses lebih banyak dibanding aplikasi yang hanya menekan fungsi *run* untuk mengeksekusinya. *File* WMB inilah yang nantinya dapat dipanggil dalam program. Fungsi tombol *run* dalam *editor* ini belum dapat membentuk *file* dengan ekstensi EXE secara langsung, sehingga belum ada *file* yang dapat langsung dibuka untuk melihat aplikasi yang sudah jadi. Cara untuk membentuk file tereksekusi adalah dengan memilih menu *file* kemudian menekan *"publish"* sehingga memungkinkan kita mendapat hasil yang diinginkan. Kelebihan menu ini adalah dapat membentuk suatu *folder* baru yang berisi *file* aplikasi yang sudah jadi disertai *file-file* lain yang dibutuhkan aplikasi dan ini merupakan hasil yang dapat didistribusikan ke pengguna.

Satu hal yang ada pada WED, setiap kali terjadi perubahan pada *file* maka untuk mengeksekusinya harus dilakukan langkah-langkah seperti sebelumnya yaitu kembali di-*built* lagi setelah itu di-*run*. Lama waktu untuk merender sebuah aplikasi tergantung pada besarnya *file* pada obyek yang direnderer ataupun

kekompleksan isi dari *file* tersebut. Sebagai contoh obyek sepeda yang dipakai dalam tugas akhir ini pernah dicoba untuk disatukan sebuah terrain yang kompleks dan berkapasitas sampai ± 5Mb, waktu untuk membuilt-nya sekitar 9200 detik, dibandingkan bila dipasang pada *hollow cube* sederhana hanya memakan waktu 1-15 detik.

Selain WED, editor yang juga perlu dikaji lebih jauh adalah *editor* untuk *script* atau disingkat SED dengan versi 7.05.5 . Bahasa pemograman yang dipakai dalam SED adalah Lite-C yang merupakan *object oriented language* dan merupakan bahasa pengembangan dari bahasa C. Tidak seperti bahasa C yang umum, misalnya Turbo-C, *syntax* di Lite-C ini sudah terarah ke pemograman visual 3D. *Script* yang dituliskan di SED nantinya akan diintegrasikan di WED umumnya berekstension WDL atau C. Hal yang unik dalam Lite-C adalah salah satu *method*-nya yang bernama *action. Action* ini berada dalam sebuah *file* yang dibuat di SED dan bila *file* tersebut sudah diintegrasikan ke WED, maka secara otomatis WED dapat membaca *action* ini sehingga kita dapat dengan mudah menyatukannya pada obyek atau entiti yang diinginkan. *Action* ini merupakan *method* yang mengimplementasikan sebuah proses khusus yang bersesuaian dengan tujuan pembuatannya. *Method* lain dalam bahasa ini yang sudah tidak asing adalah *function,* secara sekilas *function* dan *action* hampir sama, hanya ada tiga perbedaan utama yaitu pada *function* mengeluarkan sebuah nilai (me*return* sebuah *value*) sedangkan pada *action* tidak ada nilai yang dikembalikan. Perbedaan kedua, seperti telah disebutkan diatas, *action* muncul pada list di WED sedang *function* tidak muncul. Perbedaan ketiga *action* dapat langsung jalan sesaat setelah program di-*run* sedang *function* harus dipanggil dulu baru dapat dieksekusi.

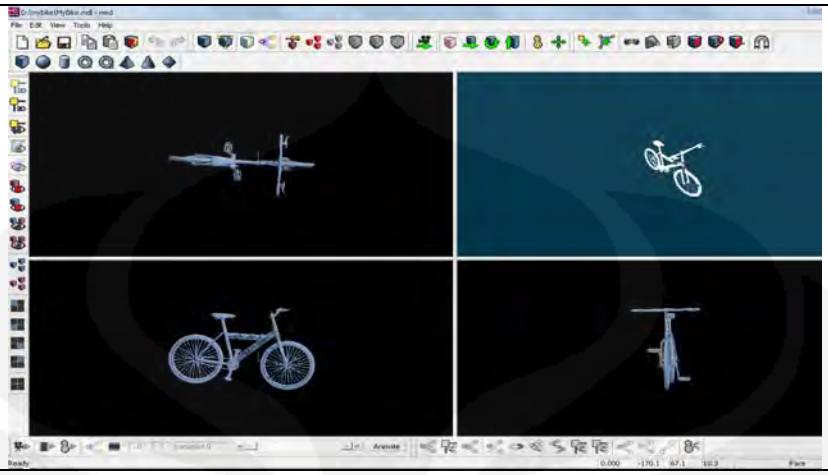
Editor yang lain yaitu MED dengan versi 6.873 adalah salah satu *editor* untuk membentuk model. Selain membentuk model sendiri melalui bentuk-bentuk *default* yang menjadi fiturnya, MED juga dapat mengimport *file* lain seperti yang dijelaskan di bab 2, *file* yang dipakai dalam tugas akhir ini adalah *file* yang diimpor dari *file* 3DS. Satu hal yang perlu diperhatikan dengan sebuah obyek adalah *polygon budget.* Jika setiap obyek yang digunakan memiliki ribuan *polygon* maka kita memerlukan komputer *high end* untuk me-*run* program

tersebut. Sebagai contoh, *video card* yang normal hanya dapat merender *mesh* sampai 65536 verteks atau segitiga. Model yang baik adalah model yang memiliki jumlah verteks dan permukaan atau sisi yang sedikit. Seperti yang dijelaskan di bab sebelumnya, bahwa obyek 3D merupakan kumpulan titik-titik yang saling berhubungan pada sumbu koordinat xyz. Sehingga secara konsep model itu memiliki titik pusat sendiri. Begitu pula dengan lingkungan, tentunya memiliki titik pusat. Sebuah lingkungan akan mendeteksi letak sebuah obyek dengan mengambil salah satu titik pada obyek tersebut yaitu titik pusat obyek itu sendiri. Pemodelan pada 3D memiliki 2 cara yaitu dengan cara *framing* yang artinya setiap gerakan dari sebuah obyek digambarkan satu demi satu sesuai kebutuhannya misalnya pergerakan lari dan disimpan berurutan dalam sebuah *array* yang nantinya dapat dipanggil sesuai fungsi yang diinginkan. Cara kedua adalah menggunakan program, cara ini lebih sulit dibandingkan cara pertama, namun tidak perlu menggambarkan semua pergerakan obyek seperti cara pertama.

Aplikasi yang dibuat pada tugas akhir ini adalah obyek sepeda, aplikasinya masih memiliki kekurangan yaitu pergerakan roda yang belum seimbang dengan stang, sebagaimana sepeda pada umumnya. Pada umumnya sepeda saat bergerak ke kiri atau ke kanan maka seharusnya stang sepeda juga mengikuti setelah itu baru badan lainnya. Stang pada obyek sepeda yang dibuat belum bergerak dikarenakan sepeda dibagi dalam 3 bagian saja, yaitu roda depan, roda belakang, rangka atas, sehingga bagian stang tidak terpisah atau dengan kata lain masih bersatu dengan rangka atas. Hal ini menyebabkan gerakan stang tidak fleksibel. Selain stang bagian lain yang belum bergerak adalah bagian pedal, ini juga disebabkan bagian pedal masih menyatu dengan rangka atas. Tidak adanya pergerakan kedua bagian ini menyebabkan pergerakan sepeda belum sempurna.

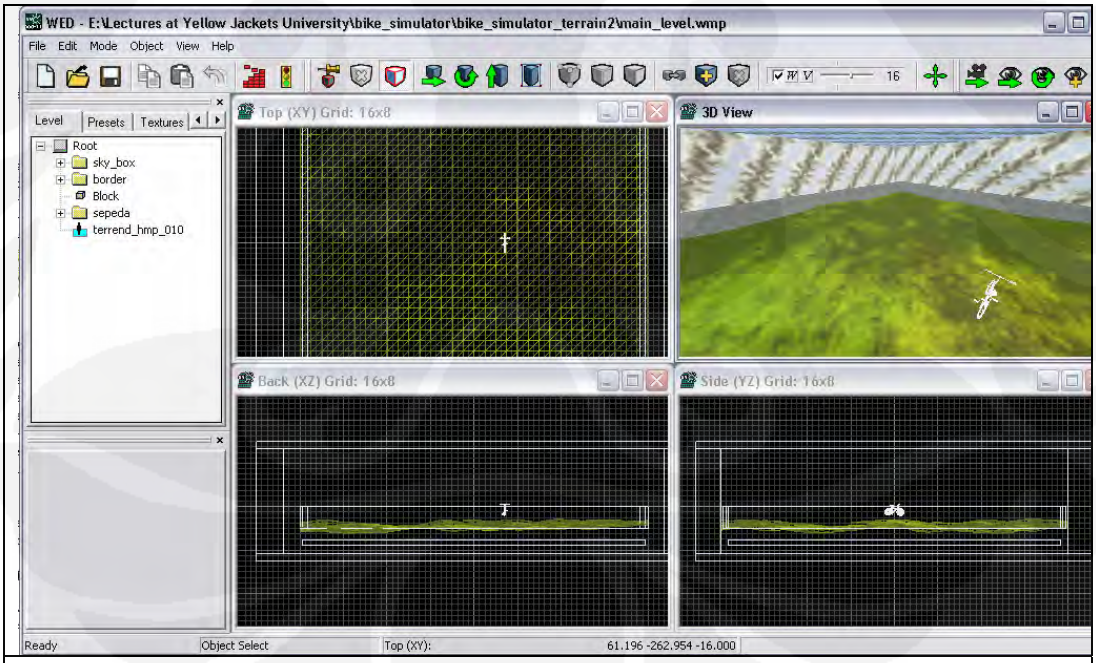
Adapun setelah diuji, aplikasi ini sudah memenuhi kriteria yang dirancang yaitu pergerakan terjadi bilamana pada *keyboard* ditekan beberapa tombol yaitu ”W” untuk maju, ”S” untuk mundur, ”A” untuk kiri, ”D” untuk kanan, dan ”C” untuk memilih perspektif kamera, selain itu tombol *space* juga untuk berhenti.

Berikut adalah gambar obyek yang dibuat :



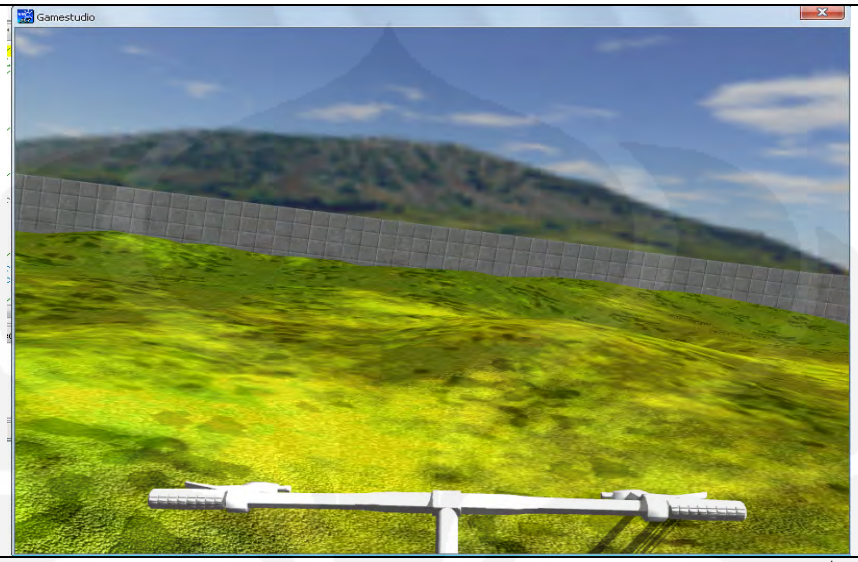
Gambar 4.1 Model Sepeda di MED

Gambar 4.1 menggambarkan model di editor pemodelan MED, sedangkan Gambar 4.2 menggambarkan obyek sepeda diletakkan di editor level WED, dengan sebuah *terrain* yang sederhana.



Gambar 4.2 Model Sepeda di WED

Gambar 4.3 dan 4.4 adalah gambar setelah aplikasi dirun.



Gambar 4.3 Model Sepeda di Lingkungan Berkontur Dengan Perspektif 1st Person

Gambar 4.4 Model Sepeda di Lingkungan Berkontur Dengan Perspektif 3rd Person (kamera yang orbit)

Gambar 4.5 Model Sepeda di tanah datar Dengan Perspektif 3rd Person (kamera pengikut)

4.2 PENGUJIAN OLEH PENGGUNA

Umpan balik dari pengguna didapatkan dengan mempersilahkan 10 penguji untuk mencoba simulasi obyek sepeda yang dibuat. Berikut adalah datanya:

Tabel 4.1 Data Tanggapan Penguji Terhadap Aplikasi

Pertanyaan	Jumlah Tanggapan Terhadap Nilai			
	1	2	3	4
1. Anda familiar dengan 3D Applications.	3	1	2	4
2. Anda familiar dengan Lite-C.	8	1	0	0
3. Anda familiar dengan 3D Gamestudio.	6	3	0	1

1. Sangat tidak familiar (tidak pernah dengar)
2. Tidak familiar (pernah dengar tapi tidak pernah menggunakan)
3. Familiar (pernah dengar, tahu, pernah menggunakan)
4. Sangat tahu (pernah dengar, tahu, dan sering menggunakan)

Tabel 4.2 Data Tanggapan Penguji Terhadap Kondisi Obyek

Pertanyaan	Jumlah Tanggapan Terhadap Nilai			
	1	2	3	4
4. Desain objek 3d (sepeda) sudah baik.	0	1	6	3
5. Pengontrolan pergerakan objek sepeda dapat dilakukan dengan mudah? (pergerakan roda).	0	1	5	4
6. Pergerakan maju sudah baik.	0	0	5	5
7. Pergerakan mundur sudah baik.	0	1	4	5
8. Pergerakan kanan sudah baik.	0	0	3	7
9. Pergerakan kiri sudah baik.	0	0	4	6
10. Pemilihan perspektif kamera *first person* sudah baik. (kamera langsung dari obyek).	0	0	3	7
11.Pemilihan perspektif kamera *third person* sudah baik. (kamera tidak langsung/ untuk melihat obyek)	0	0	2	8
12 Kecepatan obyek sudah baik.	0	2	9	1
13. Deteksi tabrakan obyek sudah baik.	0	0	4	6
1. Sangat tidak setuju 2. Tidak setuju 3. Setuju 4. Sangat Setuju				

Tabel 4.3 Data Tanggapan Penguji Terhadap Pendapat Umum

Pertanyaan	Jumlah Tanggapan Terhadap Nilai			
	1	2	3	4
14. Dunia virtual benar-benar dapat menggambarkan dunia nyata dengan baik.	0	2	4	4
15. Proyek ini bisa dikembangkan di masa depan.	0	0	0	10
1. Sangat tidak setuju 2. Tidak setuju 3. Setuju 4. Sangat Setuju				

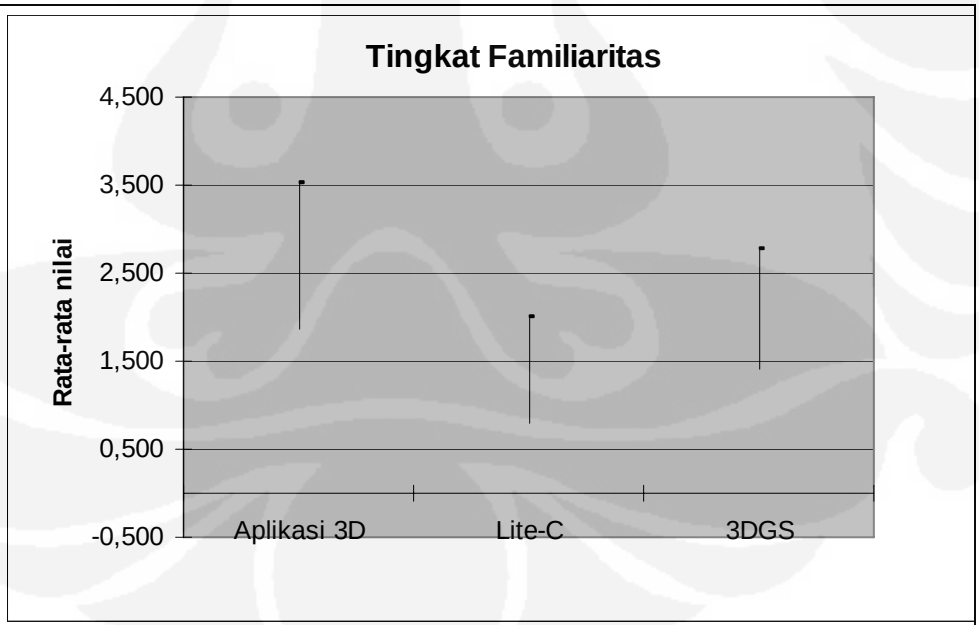
Dari data tersebut dihitung rata-rata dan standar deviasinya yang hasilnya dapat dilihat pada tabel 4.2. Dalam perhitungan ini digunakan tingkat keyakinan 95%, dengan rumus :

$$Rata\text{-}rata \pm ((1{,}96 \times Standar\ Deviasi)/ \sqrt{Populasi})$$

Tabel 4.4 Hasil Pengolahan Data Tanggapan Kuisioner

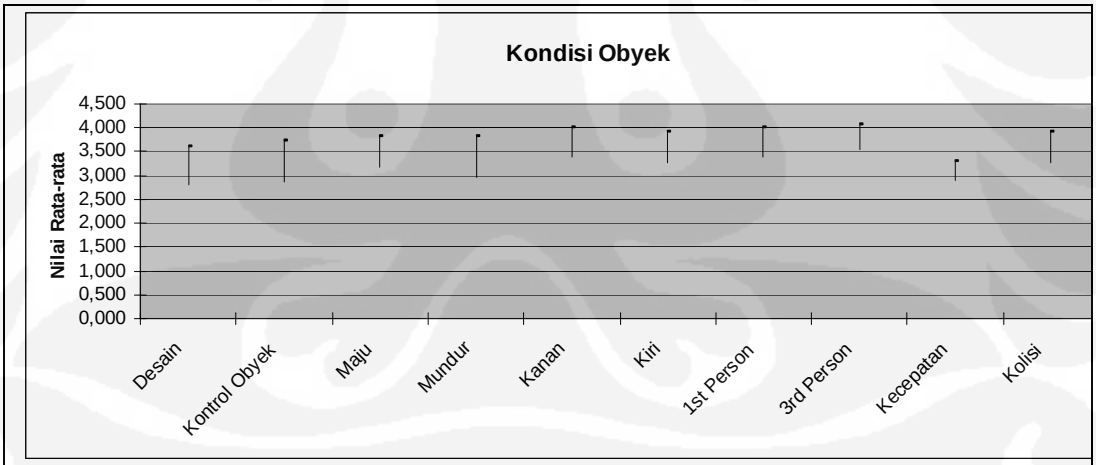
Pertanyaan	Hasil
1. Anda familiar dengan 3D Applications.	2,7 ± 0,829
2. Anda familiar dengan Lite-C.	1,4 ± 0,599
3. Anda familiar dengan 3D Gamestudio.	2,1 ± 0,682
4. Desain objek 3d (sepeda) sudah baik.	3,2 ± 0,392
5. Pengontolan pergerakan objek sepeda dapat dilakukan dengan mudah? (pergerakan roda).	3,3 ± 0,418
6. Pergerakan maju sudah baik.	3,5 ± 0,327
7. Pergerakan mundur sudah baik.	3,4 ± 0,433
8. Pergerakan kanan sudah baik.	3,7 ± 0,299
9. Pergerakan kiri sudah baik.	3,6 ± 0,320
10. Pemilihan perspektif kamera *first person* sudah baik. (kamera langsung dari obyek).	3,7 ± 0,299
11.Pemilihan perspektif kamera *third person* sudah baik. (kamera tidak langsung/ untuk melihat obyek)	3,8 ± 0,261
12 Kecepatan obyek sudah baik.	3,1 ± 0,196
13. Deteksi tabrakan obyek sudah baik.	3,6 ± 0,320
14. Dunia virtual benar-benar dapat menggambarkan dunia nyata dengan baik.	3,2 ± 0,489
15. Proyek ini bisa dikembangkan di masa depan.	4 ± 0,000

Hasil perhitungan di atas digambarkan kedalam tiga buah grafik, yang masing-masing memuat kelompok pertanyaan tertentu.



Gambar 4.6 Grafik Tanggapan Terhadap Tingkat Familiaritas Bahasa Pemograman

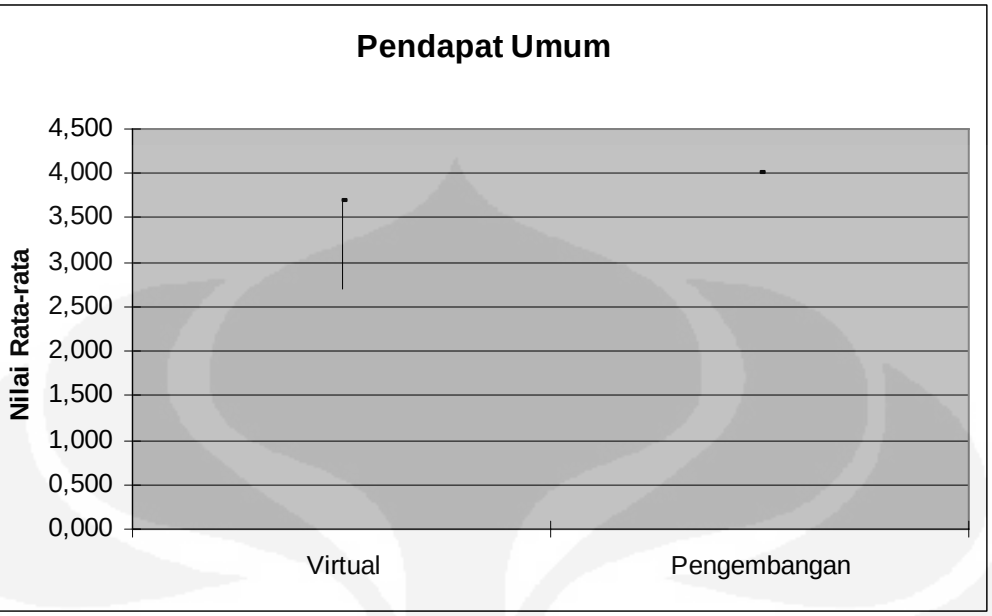
Dari grafik tanggapan terhadap tingkat familiaritas bahasa pemograman dapat dilihat bahwa rata-rata pengguna tidak familiar dengan bahasa pemograman yang digunakan. Angka rata-rata keseluruhan adalah (2,7+1,4+2,1)/3=2,07 dari skala 1-4, yang artinya pengetahuan tentang bahasa pemograman ini memang masih minim, penguji kebanyakan pernah mendengar istilah-istilah ini tapi tidak pernah menggunakan atau mengetahuinya secara mendalam. Untuk aplikasi 3D sebagian penguji hanya merupakan konsumen dari aplikasi-aplikasi 3D yang sudah jadi dan banyak beredar. Kebanyakan user belum pernah tahu tentang Lite-C ataupun 3DGS, setelah dikonfirmasi ternyata beberapa user lebih mengenal bahasa pemograman non-3D seperti bahasa pemograman pengolahan database, bahasa pemograman berbasis web, dan sebagainya. Di Indonesia sendiri, aplikasi 3D Gamestudio memang lebih jarang dipakai dibandingkan aplikasi 3D lainnya seperti 3D Max ataupun Maya, ini juga yang menyebabkan penguji jarang mendengar 3DGS ataupun bahasa yang dipakainya yaitu bahasa Lite-C. Bahasa pemograman yang biasanya lebih dikenal penguji adalah Pascal, Turbo C, Java, PHP, dan sebagainya.



Gambar 4.7 Grafik Tanggapan Terhadap Kondisi Obyek Secara Umum

Dari grafik tanggapan terhadap kondisi obyek didapatkan nilai rata-rata keseluruhan adalah (3,6+3,2+3,3+3,5+3,4+3,7+3,6+3,7+3,8+ 3,6)/ 10= 3,49 dari skala 1-4, nilai ini menunjukkan tanggapan penguji sudah baik karena skala nilai yang ada menunjukkan respon positif dan setuju bahwa kondisi obyek sudah baik. Hal yang memang masih ditanggapi oleh penguji adalah kondisi fisik obyek yang

memang belum begitu baik seperti yang ada di dunia nyata. Dari segi desain, memang masih terlihat sederhana karena masih berwarna putih atau belum di-*skinning*, selain itu roda depan belum menyatu dengan rangka, ini dibuat agar roda depan dapat berputar dengan lebih leluasa ke kiri ataupun ke kanan. Desain fisik keseluruhan (diluar warna obyek) yang lebih baik dapat diperoleh bila teknik pemodelan yang dilakukan adalah *framing*, sehingga model yang ada tetap utuh dan sempurna dan pergerakannya diatur dari sekumpulan bentuk model yang nantinya akan dipanggil sesuai dengan kebutuhan pergerakan. Untuk segi pergerakan tanggapannya sudah baik, obyek sepeda sudah dapat bergerak ke kanan, kiri, maju, mundur, berhenti, namun ada tanggapan dari penguji bahwa obyek belum begitu seimbang. Dari segi perspektif kamera sudah ditanggapi oleh penguji bahwa perspektifnya sudah baik. Untuk deteksi tabrakan obyek sepeda sudah tidak menembus obyek lain misalnya dinding pembatas. Pada saat mendeteksi halangan obyek tidak langsung diam di tempat seperti ada gerakan mundur ke belakang untuk menghindarinya, hal ini normal terkait dengan bentuk obyek roda yang bulat. Untuk kecepatan pergerakan sepeda juga sudah baik, kecepatan dapat terkontrol baik di tanah yang datar, untuk tanah yang berkontur pada saat sepeda akan menanjak kadang kala sepeda harus dimundurkan dulu sedikit kemudian baru dimajukan, sehingga daya yang ada cukup untuk menanjak. Demikian halnya untuk belok ke kanan atau ke kiri pada sebuah tanjakan, pergerakan sepeda dapat dibantu dengan menekan tombol maju sehingga sepeda mempunyai daya untuk melakukan gerakan yang diinginkan.



Gambar 4.8 Grafik Pendapat Umum

Untuk pendapat secara umum, kebanyakan penguji masih berpendapat bahwa dunia virtual dapat meniru dunia nyata dengan baik dengan nilai rata-rata sebesar 3,2. Skala ini menunjukkan bahwa sebagian besar penguji setuju terhadap pernyataan bahwa dunia virtual dapat mewakili dunia nyata. Untuk mencapai hasil yang memang benar-benar mendekati dunia nyata, diperlukan imajinasi tinggi serta ketrampilan yang lebih memadai dalam memanfaatkan aplikasi yang ada untuk mencapai suatu hasil yang lebih baik lagi. Tentunya pengembangan dari sisi aplikasi juga perlu ditingkatkan, sehingga dapat membantu penggunanya membuat dunia virtual yang menyerupai dunia sebenarnya. Sedangkan pendapat untuk pengembangan di masa depan semua responden mendukung pengembangan program ini ataupun pembuatan aplikasi-aplikasi lain yang lebih baik untuk kebutuhan-kebutuhan tertentu, hal ini terlihat dari nilai rata-rata yang didapatkan yaitu 4 dari skala 1-4.

4.3 PENGEMBANGAN MASA DEPAN

Proyek ini sangat baik untuk dikembangkan ke depan khususnya pemakaian aplikasi 3D Gamestudio karena aplikasi ini sebenarnya merupakan aplikasi yang menawarkan fitur-fitur yang cukup lengkap untuk menciptakan aplikasi 3D walaupun tujuan utamanya adalah membuat *game*. Satu hal yang perlu diperhatikan pada saat pemakaian adalah perlunya mengerti dan mendalami aplikasi ini secara mendalam untuk mendapatkan hasil yang benar-benar baik terutama untuk pemula pengguna aplikasi 3D. Pengertian dan pendalaman aplikasi bukan hanya tertuju pada penggunaan 3DGS semata, tetapi konsep mendasar tentang pemograman 3D itu sendiri. Untuk obyek sepedanya sendiri hasil yang sudah didapatkan sudah baik namun pengembangan lain dapat dibuat untuk mendapatkan hasil yang lebih sempurna lagi mengingat sepeda yang sekarang ini dibuat masih memiliki kekurangan seperti dalam hal desain serta pergerakan stang dan pedal. Hal lain yang bisa dikembangkan adalah penyatuan obyek dengan obyek lain seperti pengendara sepeda, penyatuan dengan lingkungan yang lebih real khusunya untuk kebutuhan lingkungan UI yang lebih real, ataupun penyatuan dengan obyek bergerak lain yang muncul bersamaan dalam sebuah tampilan.

Walaupun sudah banyak aplikasi 3D seperti *game* 3D yang beredar luas, ada baiknya hal tersebut dapat menjadi pembelajaran untuk membuat aplikasi yang mempunyai tujuan tertentu. Yang menjadi pemikiran untuk pengembangan aplikasi ini adalah dibuatnya simulasi yang dapat membantu pengajaran di UI ataupun simulasi lain yang dapat dijadikan sebagai aplikasi penunjang pengenalan tentang UI ke masyarakat umum dengan teknologi terkini.

BAB V

KESIMPULAN

Setelah melakukan pemodelan, pemograman, pengujian, maka penulis dapat mengambil kesimpulan :

1. 3D Gamestudio adalah aplikasi 3D yang tujuan utamanya adalah untuk membuat aplikasi *game* selain itu dapat juga digunakan untuk membuat aplikasi *virtual reality*.
2. Aplikasi 3DGS memiliki tiga *editor* yang saling terkait dan perlu pemahaman yang mendalam pada fungsi fiturnya masing-masing serta keterkaitan antara ketiganya sehingga memudahkan pembuatan aplikasi *virtual reality* yang benar-benar dapat menyerupai dunia nyata.
3. Pemograman 3D khususnya obyek yang bergerak pada tugas akhir ini menggunakan konsep pergerakan yang dikontrol melalui program, bukan dengan metode pengembangan pemodelan obyek bergerak yang dilakukan secara *framing* (dimana pergerakan diatur dengan memanggil model tertentu yang diinginkan dari sekumpulan bentuk model pergerakan yang telah dibuat sebelumnya).
4. Pada tugas akhir ini telah berhasil dibuat obyek sepeda dengan perspektif kamera tertentu dan pergerakan kanan, kiri, maju, mundur, berhenti dengan pergerakan roda yang sesuai dengan kondisi tanah serta daya yang diperlukan untuk mencapai hal itu.
5. Hasil pengujian yang dilakukan pada 10 koresponden didapatkan bahwa tingkat pengenalan terhadap bahasa pemograman yang dipakai masih kecil dengan nilai rata-rata sekitar 2,07 dari skala 1-4, untuk kondisi obyek didapatkan tanggapan baik dengan nilai rata-rata 3,49 dari skala 1-4, tanggapan terhadap *virtual reality* juga baik dengan nilai 3,2 dari skala 1-4 dan tanggapan terhadap pengembangannya di masa depan bernilai rata-rata 4, yang artinya semua penguji setuju proyek ini dapat dikembangkan lebih lanjut.

DAFTAR ACUAN

[1] Virtual reality , http://en.wikipedia.org/wiki/Virtual_reality, diakses tanggal 4 Januari 2008

[2] Christoph Bungert (Germany), http://www.stereo3d.com/3dhome.htm, diakses tanggal 4 Maret 2008

[3] E-Dimensional 3D Glasses Guide Stup.pdf, http://www.edimensional.com/ , diakses tanggal 2 Februari 2008

[4] GameStudio-FAQ, http://www.conitec.net/english/gstudio/faq.htm#was1 , diakses tanggal 4 Maret 2008

[5] 3D Game Studio, http://en.wikipedia.org/wiki/3D_Game_Studio, diakses tanggal 3 April 2008

[6] Grup Conitech, Gamestudio Manual, Edisi Extra versi 7,05

[7] Lite-C, http://en.wikipedia.org/wiki/Lite-c, diakses tanggal 3 April 2008

[8] Lite-C, http://www.conitec.net/english/gstudio/litec.htm, diakses tanggal 6 April 2008

[9] Tabel Fitur-Fitur Edisi Gamestudio, http://www.conitec.net/english/gstudio/order7.htm , diakses tanggal 6 April 2008

[10] Basuki A, Ramadijanti N, Grafik 3 Dimensi, http://lecturer.eepis- its.edu/~basuki/lecture/Grafik3D.pdf, diakses tanggal 5 Juni 2008

DAFTAR PUSTAKA

G. Booch, J.Rumbaugh, I.Jacobson, *The Unified Modeling Language User Guide* (Massachusetts: Addison Wesley Longman.Inc, 1998).

Model sepeda, http://www.3dvia.com/search/models/tags/bike, diakses tanggal 23 Mei 2008.

Model sepeda, http://www.turbosquid.com/Search/Index.cfm/FuseAction/ProcessSmartSearch/intCategory/20075, diakses tanggal 9 Juni 2008.

Muliawan, "Membangun Museum Virtual dengan Croquet". Skripsi, Program Sarjana S1 Fakultas Teknik UI, 2003.

Standar Deviasi, http://www.vias.org/tmdatanaleng/cc_standarddev.html, diakses 4 Juli 2008.

Tutorial Movement di 3DGS, http://www.xuduo.cn/Tutorial/Manual_en/ph_constraints.htm, diakses tanggal 18 Juni 2008.

LAMPIRAN

LAMPIRAN 1 LISTING PROGRAM

```
function main()
{
   video_mode=7;   //800x600
      shadow_stencil= OFF;
   d3d_autotransparency = OFF;
   vec_set(d3d_lodfactor,vector(35,60,75)); // default LOD

   detail_size = 16;
   terrain_lod = 0;


   on_h = EventImpact;


   wait(1);
      level_load(levelname);

      while(pFocus==NULL){wait(1);}

      // aktivasi real kamera
      camMode = 1;

}
function ControlCamera_startup()
{
      // view params
      fog_color = 1;
      camera.fog_start=1000;
      camera.fog_end  =1.2*camera.clip_far;
      camera.clip_near=25;

      // inisialisasi level dan tunggu untuk fokus ke obyek (kita
akan orbit disekitarnya)
      while(pFocus==NULL || camMode<0){wait(1);}

      // jarak sekarang dari focus dan sudut kamera
      var cam_dist=200;
      ANGLE cam_ang;
      // reset jika sebelumnya masih bergulung
      cam_ang.pan= pFocus.pan-180;
      cam_ang.tilt=30;
      cam_ang.roll=0;

      while(1)
      {
            // untuk fokus ke obyek (orbit di sekitar entiti)
            while(pFocus==NULL){wait(1);}

            if (camMode==0) // kamera posisi orbit
            {
                  reset(pFocus,INVISIBLE);
                  //orbit
                  if(mouse_right)
                  {
                        cam_ang.pan+=rotspd*mouse_force.x;
                        cam_ang.tilt+=rotspd*mouse_force.y;

      cam_ang.tilt=clamp(cam_ang.tilt,minang,maxang);
                  }
```

```
                  // zoom dengan mouse kanan
                  cam_dist-=integer(mickey.z);
                  cam_dist=clamp(cam_dist,mindist,maxdist);

                  // mencari lokasi kamera dan mengarahkannya ke
entiti pFocus

      camera.x=pFocus.x+cos(cam_ang.pan)*(cam_dist*cos(cam_ang.til
t));

      camera.y=pFocus.y+sin(cam_ang.pan)*(cam_dist*cos(cam_ang.til
t));
                  camera.z=pFocus.z+sin(cam_ang.tilt)*cam_dist;
                  vec_set(temp,pFocus.x);
                  vec_sub(temp,camera.x);
                  vec_to_angle(camera.pan,temp);
            }
            if (camMode==1) // kamera yang mengejar obyek
            {
                  reset(pFocus,INVISIBLE);
                  if(mouse_right)
                  {
                        // mouse untuk memindahkan kamera bagian
depan
                        vecChaseAngOff.pan-=rotspd*mouse_force.x *
time_step;
                        vecChaseAngOff.tilt+=rotspd*mouse_force.y
* time_step;

      vecChaseAngOff.pan=clamp(vecChaseAngOff.pan,-20,20);

      vecChaseAngOff.tilt=clamp(vecChaseAngOff.tilt,-20,20);
                  }
                  else
                  {
                        // kembali ke pusat
                        if(vecChaseAngOff.pan        >        0.25)
{ vecChaseAngOff.pan -= 2* time_step; }
                        if(vecChaseAngOff.pan       <       -0.25)
{ vecChaseAngOff.pan += 2* time_step; }
                        if(vecChaseAngOff.tilt         >         6)
{ vecChaseAngOff.tilt -= 2* time_step; }
                        if(vecChaseAngOff.tilt         <         5)
{ vecChaseAngOff.tilt += 2* time_step; }

                  }

                  vec_set(cameraTPos,     vecChaseOffset);      //
pindahkan kamera ke belakang dan di atas obyek
                  var offset;
                  offset= clamp(0.01*linSpeed*linSpeed, -100,100);
                  cameraTPos[0] -= offset;
                  vec_rotate(cameraTPos,
vector(ang(pFocus.pan),0,0));
                  vec_add(cameraTPos, pFocus.x);

                  // kolisi dengan obyek lain di tengah jalan

      c_trace(pFocus.x,cameraTPos,(IGNORE_PASSABLE+IGNORE_PASSENTS
+IGNORE_MODELS));
                  if(trace_hit == 1)
                  {

                        vec_set(cameraTPos,target);

      vec_normalize(normal,(camera.clip_near/2));
                        vec_add(cameraTPos,normal);
```

```
                }
                else
                {
                        // pastikan untuk tidak terjepit ke dalam
dinding yang ada di dekatnya..
                        vec_diff(temp,pFocus.x,cameraTPos);
                        vec_normalize(temp,camera.clip_near);
                        vec_rotate(temp,vector(90,0,0));
                        vec_set(temp2,cameraTPos);
                        vec_add(temp2,temp);

      c_trace(cameraTPos,temp2,(IGNORE_PASSABLE+IGNORE_PASSENTS+IG
NORE_MODELS));
                        if(trace_hit == 1)
                        {

                              vec_set(cameraTPos,target);

      vec_normalize(normal,(camera.clip_near/2));
                              vec_add(cameraTPos,normal);

                        }
                        else
                        {

                              vec_rotate(temp,vector(-180,0,0));
                              vec_set(temp2,cameraTPos);
                              vec_add(temp2,temp);

      c_trace(cameraTPos,temp2,(IGNORE_PASSABLE+IGNORE_PASSENTS+IG
NORE_MODELS));
                              if(trace_hit == 1)
                              {

                                    vec_set(cameraTPos,target);

      vec_normalize(normal,(camera.clip_near/2));
                                    vec_add(cameraTPos,normal);

                              }
                        }

                }

                // Menghaluskan gerakan transisi
                vec_diff(temp,cameraTPos,camera.x);
                var vscale = clamp((time_step*0.5),0.001,1);
                vec_scale(temp,vscale);
                vec_add(camera.x,temp);
                // menghitung sudut dari kamera ke target
                vec_diff(temp,pFocus.x,cameraTPos);
                vec_to_angle(cameraTAng,temp);
                vec_add(cameraTAng,   vecChaseAngOff.pan);     //
putar
                // Menghaluskan gerakan rotasi...
                vec_diff(temp,cameraTAng,camera.pan);
                //catatan  hanya  atribut  pan  yang  dibutuhkan
dalam aplikasi ini
                temp[0] = ang(temp[0]);
                vec_scale(temp,vscale);
                vec_add(camera.pan,temp);

            }

            if (camMode==2) // kamera langsung dari sepeda (stang)
            {
```

```
                  vec_set(camera.pan, pFocus.pan);
                  vec_set(camera.x, pFocus.x);
                  vec_set(temp, vecDriverOffset);
                  vec_rotate(temp, pFocus.pan);
                  vec_add(camera.x, temp);
            }

            if (KEY_CAMERA || JOY_CAMERA) {
                  while (KEY_CAMERA || JOY_CAMERA) {  wait(1); }
                  camMode+=1; if (camMode>2) { camMode=0; }
                  // reset if previously rolled around
                  vec_set(camera.pan, cam_ang);
            }
            wait(1);
      }
}
action BikeInit()
{
      // buat sepeda dan menunggu sampai roda siap
      InitRangka();
      my.material = mat_metal;

      while (wheelCounter<2) { wait(1); }
      UpdateKecepatan();   //  update   kecepatan   roda   belakang
(kontinyu)
      KontrolKecepatan();// mengubah kecepatan (kontinyu)
      KontrolStir();// mengubah heading (kontinyu)
}
function UpdateKecepatan()
{
      while (wheelCounter<2) { wait(1); } //wait for wheels to be
init'ed

      var   rotSign;
      rotSign = 1;

      var angRR;
      VECTOR vecLin;
      while(1) {
            if (DoShutdown) {
                  return;
            }

            // get kecepatan linear
            phent_getvelocity(pChassis,    vecLin,    nullvector);
//kecepatan  linear
            downSpeed= vecLin.z * 0.091; vecLin.z=0;
            linSpeed= vec_length(vecLin) * 0.091; // mengkonversi
quant/sec ke kmh
            if (maxAngSpeed>0)
            {
                  var freq; freq=300*abs(angSpeed)/maxAngSpeed;

            }

            // ambil vector velocity angular dari roda belakang
            phent_getangvelocity(pRR, temp); angRR=temp[1];
            angSpeed=rotSign * angRR;
      wait(1);
      }
}
function KontrolKecepatan()
{
      var direction;
      var usejoy=0;
      while (1)
```

```
	{
		if (DoShutdown) { return; } // keluar?
		if (!DoSteering)
		{
			wait(1); continue;
		}

		if (JOYRAW_Y<-joyNullzone.y || JOYRAW_Y>joyNullzone.y)
		{
			// joystick
			if (USE_GAMEPAD) {
				usejoy=0; direction= -(JOYRAW_Y / 255);
			} else {
				usejoy=1;	direction=	-(JOYRAW_Y-
joyNullzone.y)/(255-joyNullzone.y);
			}
		} else {
			usejoy=0;  direction=  (KEY_MAJU-KEY_MUNDUR)  +
joy_force.y;
		}
		// jika ingin mengendarai pada satu arah dan ingin
mengubah arah, aplikasikan brake
		// (angSpeed>0 checks jika roda belakang dimajukan
		//  direction<0  checks  jika  user  menekan  key  yang
sebaliknya
		if  ((angSpeed>5  &&  direction<0)  ||  (angSpeed<-5  &&
direction>0))
		{
			targetSpeed=0; // brake

			if (abs(angSpeed)>0.25*maxAngSpeed)
			{
				phcon_setmotor(wheelRR,		nullvector,
vector(angSpeed*0.5, 0.5*maxTorqueBrake,0), nullvector);
				if (abs(angSpeed)>0.50*maxAngSpeed)
				{
					EventBrake();
				}
			} else {
				phcon_setmotor(wheelRR,		nullvector,
vector(0, maxTorqueBrake,0), nullvector);
			}
		} else {
			var oldSpeed; oldSpeed= targetSpeed;
			// pengendaraan regular
			if (usejoy) {
				targetSpeed	=	maxAngSpeed*direction;
//nilai absolute untuk stick
			} else {
				//  kecepatan  bertambah  untuk  keyboard
(relative acceleration)
				targetSpeed  +=  direction*accelKeyboard*
time_step;
			}

			var torque; torque=maxTorque;
			//  kecepatan  menurun  jika  tidak  ada  key  yang
ditekan
			if (!usejoy && direction==0)
			 {
				targetSpeed		*=1-(time_step*0.02);
torque=maxTorque/2;
				if (abs(linSpeed)<20)
				{ torque=0; targetSpeed *=0.5;}
			} else {
```

```
                        if                (usejoy                &&
abs(targetSpeed)<abs(angSpeed))
                        {
                              torque=maxTorque/2;
                        }

                  }
                  targetSpeed=        clamp(targetSpeed,        -
maxAngSpeed*0.25, maxAngSpeed);
                  phcon_setmotor(wheelRR,              nullvector,
vector(targetSpeed, torque,0), nullvector);
            }
            if (KEY_BRAKE|| JOY_BRAKE) // brake
            {
                  phcon_setmotor(wheelRR,   nullvector,   vector(0,
maxTorqueBrake,0), nullvector);
                  if (abs(linSpeed)>20)
                  { EventBrake(); }
            }

            wait(1);
      }
}
function EventImpact(){

      if (!DoSteering || DoShutdown)

      {EventHorn(); return; }

      if (bDentBike!=0 && linSpeed>(CRASH_SPEED*0.7) )
      {

            var i=0; var k=0; var tttt[3];

            var min_val=500; var min_i;

            while (i < ent_vertices (my))
            {
                  vec_for_vertex(tttt,my,i);

                  k=vec_dist(tttt,target.x);
                  if(k<min_val)
                {min_val=k; min_i=i;}
                  i+=1;
            }

            vec_for_mesh(temp,my,min_i);
            vec_scale(temp,0.9);
            vec_to_mesh(temp,my,min_i);
            ent_fixnormals(my,0);
            ////////////////////////////
      }

      if((linSpeed>CRASH_SPEED && abs(normal.z)<0.1)) {
            DoSteering=0;

            wait(1);
            phcon_remove( wheelFR );
            phcon_remove( wheelRR );
            phent_setdamping( pFR, 80,50);
            phent_setdamping( pRR, 80,50);
            phent_setdamping( pChassis, 80,50);
            var timer= 80;
            while (timer>0) {
                  timer-= 1;
                  vec_set(temp, pChassis.x);
                  wait(-0.1);
```

```
            }
      }
}


function EventBrake(){

                  var wheelDir[3];
                  vec_set(wheelDir, vector(-15,random(16)-8,-15));
                  var speed; speed= (abs(angSpeed)/(maxAngSpeed));
                  vec_rotate(wheelDir, pChassis.pan);
                  vec_add(temp, wheelDir);

}

function InitRoda()
{
    set(my,SHADOW);
      if (commonWheelHeight<=-100000) {
          commonWheelHeight=my.z; // simpan z untuk roda berikut
        } else {
          my.z=commonWheelHeight; // Set semua roda untuk tinggi
yang sama
        }
      vec_set(my.skill1,  my.x);  //  simpan  pos/orientation  untuk
reset
      vec_set(my.skill4, my.pan);
      phent_settype(my, PH_RIGID, PH_SPHERE);
      phent_setmass(my, massRoda, PH_SPHERE);
      phent_setgroup(my, 2);
      phent_setfriction(my, fricWheel);
      phent_setdamping(my, 15,dampWheel);
      phent_setelasticity(my, 0, 100);
      wheelCounter+=1;
}

action RodaDepanInit()
{     set(my,PASSABLE);
      while (!pChassis) { wait(1);}
      pFR=my;
      InitRoda();
      // untuk membuat constraint roda ter-pointing ke atas dan
terarah ke pusat
      wheelFR= phcon_add(PH_WHEEL, pChassis, my);
      phcon_setparams1(wheelFR, my.x, vecUp, vecRight);
      phcon_setparams2(wheelFR,     vector(0,0,0),     nullvector,
vector(suspensionERP, suspensionCFM,0));
      reset(my,PASSABLE);
}

action RodaBelakangInit()
{
      set(my,PASSABLE);
      while (!pChassis) { wait(1);}
      pRR=my;
      InitRoda();
      // membuat constraint roda ter-pointing
      wheelRR= phcon_add(PH_WHEEL, pChassis, my);
      phcon_setparams1(wheelRR, my.x, vecUp, vecRight);
      phcon_setparams2(wheelRR,     vector(0,0,0),     nullvector,
vector(suspensionERP, suspensionCFM,0));
      reset(my,PASSABLE);
}
function InitRangka()
{
            DoShutdown=0;
            DoSteering=1;
```

```
            vec_set(my.skill1,  my.x);  //  simpan  pos/orientation
untuk reset
            vec_set(my.skill4, my.pan);
            pFocus=my;
            set(my,SHADOW|PASSABLE);

            vec_set(vecRight,vector(0,-1,0));
            vec_rotate(vecRight, my.pan);
            phent_settype(my, PH_RIGID, PH_BOX);
            phent_setmass(my, massChassis, PH_BOX);
            phent_setgroup(my, 2);
            phent_setfriction(my, fricChassis);
            phent_setdamping(my, dampChassis,dampChassis);
            phent_setelasticity(my, 10, 100);
            pChassis=my;
            pChassis.event= EventImpact;
            pChassis.emask = ENABLE_FRICTION;
            vec_set(temp, my.x);
            while (wheelCounter<2) { wait(1); }
            reset(my,PASSABLE);
            ph_setgravity(vector(0,0,-gravity));
            InitMarks();

// shutdown
            while (1) {
                  if (DoShutdown) {
                        wheelCounter=0;
                        targetSpeed=0; angSpeed=0;
                        ph_setgravity(vector(0,0,0));
                        phcon_remove(wheelFR);
                        phcon_remove(wheelRR);
                        phent_settype(pChassis, 0,0);
                        phent_settype(pFR, 0,0);
                        phent_settype(pRR, 0,0);
                        return;
                  }
                  wait(1);
            }
}

function InitMarks()
{
      while (!pChassis) { wait(1); }
      var counter; counter=NUM_TRACKS - 1;
      while (counter>=0) {
            you=   ent_createlocal(dummy_pcx,   vector(-5000,   -
900,20),0);
            arrpTracks[counter]=handle(you);
            counter-=1; //you.overlay= OFF;
            you.scale_y=trackLen;
            you.scale_x=10;
            set(you,INVISIBLE|PASSABLE|TRANSLUCENT|UNLIT);
            you.pan = 0.1;
            you.alpha=90;
      }
}


function KalkulasiSkalaStir()
{
      return (1-(abs(angSpeed)/(maxAngSpeed+60)));
}

function KontrolStir()
{
      var direction=0;
      var usejoy;
```

```
	while (1)
	{
		if (DoShutdown) { return; }
		if (!DoSteering)
		{
			wait(1); continue;
		}

		if (JOYRAW_X<-joyNullzone.x || JOYRAW_X>joyNullzone.x)
		{

			if (USE_GAMEPAD) {
				usejoy=0; direction= (JOYRAW_X / 255);
				direction*= KalkulasiSkalaStir();
			} else {
				// menggunakan joystick
				usejoy=1;	direction=	(JOYRAW_X-
joyNullzone.x) / (255-joyNullzone.x);
				direction*= KalkulasiSkalaStir();
			}
		} else {
			usejoy=0;   direction=   (KEY_KANAN-KEY_KIRI)   +
joy_force.x;
			direction*= KalkulasiSkalaStir();
		}

		// kontrol stir
		if (0==usejoy)
		{
			targetSteer+=		direction*accelSteering*
time_step;
		} else {
			targetSteer= 30*direction;
		}
		targetSteer=clamp(targetSteer,-30,30);

		if ((angSpeed>maxAngSpeed*0.95) && abs(targetSteer)>7)
		{
			EventBrake();
		}

		if (direction!=0)
		{
			phcon_setparams2(wheelFR,
vector(targetSteer,targetSteer,0),			nullvector,
vector(suspensionERP, suspensionCFM,0));
		} else {
			targetSteer*=1-(time_step*0.25);
			if (abs(targetSteer)<1) { targetSteer=0; }
				phcon_setparams2(wheelFR,
vector(targetSteer,targetSteer,0),			nullvector,
vector(suspensionERP, suspensionCFM,0));
		}

		wait(1);
	}
}
```