**BAB I**

* 1. **Latar Belakang Masalah**

Grafika komputer akhir-akhir ini mulai dirasa sangat penting dan mencakup hampir semua bidang kehidupan seiring dengan semakin pentingnya sistem komputer dalam berbagai kegiatan. Grafika komputer merupakan gambar atau grafik yang dihasilkan oleh komputer. Teknik-teknik yang dipelajari dalam grafika komputer adalah teknik-teknik bagaimana membuat atau menciptakan gambar dengan menggunakan komputer.

Stasiun kereta api adalah tempat di mana para penumpang dapat naik-turun dalam memakai sarana transportasi kereta api. Selain stasiun, pada masa lalu dikenal juga dengan halte kereta api yang memiliki fungsi nyaris sama dengan stasiun kereta api. Untuk daerah/kota yang baru dibangun mungkin stasiun portabel dapat dipergunakan sebagai halte kereta.

Stasiun berbentuk sebuah tempat yang memiliki tempat-tempat tertentu seperti bangunan serta jalur Rel Kereta api yang tentunya akan dipakai sebagai jalur kereta. Bangunan-bangunan yang ada pada stasiun layaknya seperti bangunan yang biasa kita masuki di mana ruanganya dibatasi oleh dinding dan atap, biasanya memiliki jalan masuk berupa pintu, bisa berjendela ataupun tidak. Lantainya bisa berupa tanah, ubin, keramik atau bahan lainnya. Banyak orang ingin melihat bangunan Stasiun tanpa harus datang ke lokasi nya secara langsung. Untuk itu, stasiun dapat digambar dengan komputer dengan animasi 3D agar tampak lebih menarik.

Project stasiun 3D ini akan dibuat dengan tools devc++ dengan menggunakan aplikasi interface yaitu Opengl. Objek rumah 3D ini akan dibuat dengan berbagai macam fungsi-fungsi yang memang dirasa diperlukan dalam pembuatan sebuah objek stasiun 3D serta objek-objek lain sebagai pendukunya, fungsi-fungsi yang akan digunakan misalnya, fungsi paling mendasar dari opengl itu sendiri, seperti fungsi glutWireCube, fungsi ini digunakan untuk membuat sebuah bentuk kubus dan objek ini pasti akan digunakan dalam pembuatan objek rumah 3D, fungsi lainya seperti fungsi glutWireCone, fungsi ini nantinya akan menghasilkan sebuah objek kerucut.

Untuk membuat sebuah objek stasiun 3D yang dinamis, rasanya penulis merasa bahwa harus ada fungsi-fungsi lainnya yang perlu ditambahkan, seperti fungsi pemberian warna pada objek, fungsi pemberian warna ini biasanya menggunakan glcolor. Efek 3D sebenarnya akan lebih terlihat timbul apabila objek 3D yang dibuat diberikan fungsi lagi yaitu fungsi pencahayaan, dan fungsi pencahayaan ini erat kaitannnya dengan peletakan posisi kamera nantinya, penulis nantinya akan menggunakan fungsi pencahayaan seperti glLightfv dan fungsi untuk menentukan letak kamera akan menggunakan fungsi gluLookAt.

Objek 3D dalam opengl sendiri akan lebih dinamis apabila adanya interkasi user dengan objek 3D tersebut, interaksi tersebut bisa berupa interaksi menggunakan keyboard maupun interaksi menggunakan mouse, Oleh karena itu penulis juga akan menambahkan salah satu interaksi yaitu interaksi menggunakan keyboard.

 Berdasarkan hal di atas, maka tim penulis tertarik untuk membuat objek 3D dengan judul : “OBJEK 3D Stasiun”.

**1.2 Objek List**

Daftar objek yang akan kami buat dalam menyelesaikan tugas ini kurang lebih adalah sebagai berikut :

* Tanah
* Pohon
* Bangunan Stasiun mini,
* Jalan
* Rel Kereta Api

**1.3 Objek Pembangun**

Sedangkan Objek pembangun dari masing-masing objek yang akan kami buat adalah sebagai berikut :

* Tanah = *Kotak*
* Pohon = *Cyllinder dan Dodecahedron*
* Bangunan Stasiun mini, = *kotak* untuk tembok dan jendelanya nya, *Segitiga* untuk atap dari bangunanya.
* Jalan = *Kotak*
* Rel Kereta Api = *Kotak* membentuk garis yg melingkar
* Kursi = *Kotak*
* Lantai = *Kotak*
* Kolam = *Donut*
* Air Kolam = *Donut*
* Kereta = *Bola,* dan *vertex GL\_POLYGON*

**1.4 Teknologi Pembangun**

Teknologi pembangun yang kami pakai untuk membuat objek bangun tersebut yaitu ; memanfaatkan pemograman OpenGL dan menggunakan bahasa C++ dengan compiler Dev C++.

**1.5 Metode Penelitian**

Metodologi yang digunakan dalam penulisan tugas besar ini adalah sebagai berikut :

1. Tahap Pengumpulan data

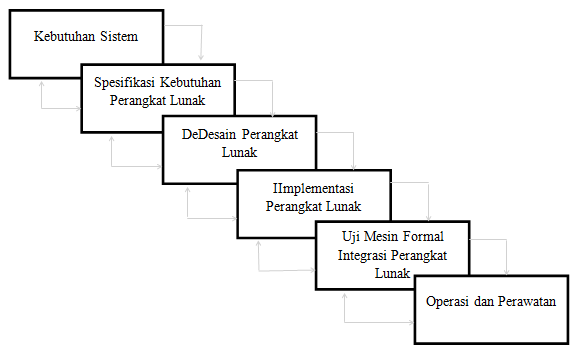
Metode pengumpulan data yang digunakan dalam penulisan tugas besar ini adalah sebagai berikut :

1. Studi Literatur

Pengumpulan data dengan cara mengumpulkan *literature*, jurnal, *paper* dan bacaan-bacaan baik dari buku atau pun media internet yang ada kaitannya dengan judul tugas besar.

1. Tahap pembuatan perangkat lunak

Teknik analisis data dalam pembuatan perangkat lunak menggunakan paradigma perangkat lunak secara *waterfall*, yang meliputi beberapa proses diantaranya :



Gambar 1.1 Waterfall

1. Kebutuhan Sistem

Karena sistem merupakan bagian dari sebuah sistem yang lebih besar, kerja dimulai dengan membangun syarat dari semua elemen sistem dan mengalokasikan beberapa subset dari kebutuhan ke *software* tersebut. Pandangan sistem ini penting ketika *software* harus berhubungan dengan elemen-elemen yang lain seperti *software*, manusia, dan *database*. Rekayasa dan anasisis sistem menyangkut pengumpulan kebutuhan pada tingkat sistem dengan sejumlah kecil analisis serta disain tingkat puncak. Rekayasa informasi mancakup juga pengumpulan kebutuhan pada tingkat bisnis strategis dan tingkat area bisnis.

*b*. Spesifikasi Kebutuhan Perangkat Lunak

Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khususnya pada *software*. Untuk memahami sifat program yang dibangun, analis harus memahami domain informasi, tingkah laku, unjuk kerja, dan *interface* yang diperlukan. Kebutuhan baik untuk sistem maupun *software* didokumentasikan dan dilihat lagi dengan pelanggan.

c. Desain Perangkat Lunak

Desain *software* sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda; struktur data, arsitektur *software*, representasi *interface*, dan detail (algoritma) prosedural. Proses desain menterjemahkan syarat/kebutuhan ke dalam sebuah representasi *software* yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, desain didokumentasikan dan menjadi bagian dari konfigurasi *software*.

d. Implementasi Perangkat Lunak

Untuk dapat dimengerti oleh mesin, dalam hal ini adalah komputer, maka desain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh mesin, yaitu kedalam bahasa pemrograman melalui proses *coding.* Tahap ini merupakan implementasi dari tahap *design* yang secara teknis dikerjakan oleh programmer.

e. Uji Mesin Formal Integrasi Perangkat Lunak

Sekali program dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika internal *software*, memastikan bahwa semua pernyataan sudah diuji, dan pada eksternal fungsional, yaitu mengarahkan pengujian untuk menemukan kesalahan – kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil aktual yang sesuai dengan hasil yang dibutuhkan.

f. Operasi dan Perawatan

*Software* akan mengalami perubahan setelah disampaikan kepada pelanggan (perkecualian yang mungkin adalah *software* yang dilekatkan). Perubahan akan terjadi karena kesalahan – kesalahan ditentukan, karena *software* harus disesuaikan untuk mengakomodasi perubahan – perubahan di dalam lingkungan eksternalnya (contohnya perubahan yang dibutuhkan sebagai akibat dari perangkat peripheral atau sistem operasi yang baru), atau karena pelanggan membutuhkan perkembangan fungsional atau unjuk kerja. Pemeliharaan *software* mengaplikasikan lagi setiap fase program sebelumnya dan tidak membuat yang baru lagi.

**BAB II**

* 1. **Pembagian Tugas**

Dalam pembuatan tugas besar ini kami membagi tugas pembuatan program maupun laporan dengan merata. Dengan demikian, dapat disimpulkan bahwa pembagian tugasnya adalah sebagai berikut :

Laporan : Supyana, Husnil, Arif

Program : Arif, Supyana, Husnil

* 1. **Source Code**

Adapun Source Code dari program yang kami buat adalah sebagai berikut :

***Main.cpp***

***Imageloader.h***

#include <assert.h>

#include <fstream>

#include "imageloader.h"

using namespace std;

Image::Image(char\* ps, int w, int h) : pixels(ps), width(w), height(h) {

}

Image::~Image() {

delete[] pixels;

}

namespace {

//Converts a four-character array to an integer, using little-endian form

int toInt(const char\* bytes) {

return (int)(((unsigned char)bytes[3] << 24) |

((unsigned char)bytes[2] << 16) |

((unsigned char)bytes[1] << 8) |

(unsigned char)bytes[0]);

}

//Converts a two-character array to a short, using little-endian form

short toShort(const char\* bytes) {

return (short)(((unsigned char)bytes[1] << 8) |

(unsigned char)bytes[0]);

}

//Reads the next four bytes as an integer, using little-endian form

int readInt(ifstream &input) {

char buffer[4];

input.read(buffer, 4);

return toInt(buffer);

}

//Reads the next two bytes as a short, using little-endian form

short readShort(ifstream &input) {

char buffer[2];

input.read(buffer, 2);

return toShort(buffer);

}

//Just like auto\_ptr, but for arrays

template<class T>

class auto\_array {

private:

T\* array;

mutable bool isReleased;

public:

explicit auto\_array(T\* array\_ = NULL) :

array(array\_), isReleased(false) {

}

auto\_array(const auto\_array<T> &aarray) {

array = aarray.array;

isReleased = aarray.isReleased;

aarray.isReleased = true;

}

~auto\_array() {

if (!isReleased && array != NULL) {

delete[] array;

}

}

T\* get() const {

return array;

}

T &operator\*() const {

return \*array;

}

void operator=(const auto\_array<T> &aarray) {

if (!isReleased && array != NULL) {

delete[] array;

}

array = aarray.array;

isReleased = aarray.isReleased;

aarray.isReleased = true;

}

T\* operator->() const {

return array;

}

T\* release() {

isReleased = true;

return array;

}

void reset(T\* array\_ = NULL) {

if (!isReleased && array != NULL) {

delete[] array;

}

array = array\_;

}

T\* operator+(int i) {

return array + i;

}

T &operator[](int i) {

return array[i];

}

};

}

Image\* loadBMP(const char\* filename) {

ifstream input;

input.open(filename, ifstream::binary);

assert(!input.fail() || !"Could not find file");

char buffer[2];

input.read(buffer, 2);

assert(buffer[0] == 'B' && buffer[1] == 'M' || !"Not a bitmap file");

input.ignore(8);

int dataOffset = readInt(input);

//Read the header

int headerSize = readInt(input);

int width;

int height;

switch(headerSize) {

case 40:

//V3

width = readInt(input);

height = readInt(input);

input.ignore(2);

assert(readShort(input) == 24 || !"Image is not 24 bits per pixel");

assert(readShort(input) == 0 || !"Image is compressed");

break;

case 12:

//OS/2 V1

width = readShort(input);

height = readShort(input);

input.ignore(2);

assert(readShort(input) == 24 || !"Image is not 24 bits per pixel");

break;

case 64:

//OS/2 V2

assert(!"Can't load OS/2 V2 bitmaps");

break;

case 108:

//Windows V4

assert(!"Can't load Windows V4 bitmaps");

break;

case 124:

//Windows V5

assert(!"Can't load Windows V5 bitmaps");

break;

default:

assert(!"Unknown bitmap format");

}

//Read the data

int bytesPerRow = ((width \* 3 + 3) / 4) \* 4 - (width \* 3 % 4);

int size = bytesPerRow \* height;

auto\_array<char> pixels(new char[size]);

input.seekg(dataOffset, ios\_base::beg);

input.read(pixels.get(), size);

//Get the data into the right format

auto\_array<char> pixels2(new char[width \* height \* 3]);

for(int y = 0; y < height; y++) {

for(int x = 0; x < width; x++) {

for(int c = 0; c < 3; c++) {

pixels2[3 \* (width \* y + x) + c] =

pixels[bytesPerRow \* y + 3 \* x + (2 - c)];

}

}

}

input.close();

return new Image(pixels2.release(), width, height);

}

***vc3f.cpp***

#include <math.h>

#include "vec3f.h"

using namespace std;

Vec3f::Vec3f() {

}

Vec3f::Vec3f(float x, float y, float z) {

v[0] = x;

v[1] = y;

v[2] = z;

}

float &Vec3f::operator[](int index) {

return v[index];

}

float Vec3f::operator[](int index) const {

return v[index];

}

Vec3f Vec3f::operator\*(float scale) const {

return Vec3f(v[0] \* scale, v[1] \* scale, v[2] \* scale);

}

Vec3f Vec3f::operator/(float scale) const {

return Vec3f(v[0] / scale, v[1] / scale, v[2] / scale);

}

Vec3f Vec3f::operator+(const Vec3f &other) const {

return Vec3f(v[0] + other.v[0], v[1] + other.v[1], v[2] + other.v[2]);

}

Vec3f Vec3f::operator-(const Vec3f &other) const {

return Vec3f(v[0] - other.v[0], v[1] - other.v[1], v[2] - other.v[2]);

}

Vec3f Vec3f::operator-() const {

return Vec3f(-v[0], -v[1], -v[2]);

}

const Vec3f &Vec3f::operator\*=(float scale) {

v[0] \*= scale;

v[1] \*= scale;

v[2] \*= scale;

return \*this;

}

const Vec3f &Vec3f::operator/=(float scale) {

v[0] /= scale;

v[1] /= scale;

v[2] /= scale;

return \*this;

}

const Vec3f &Vec3f::operator+=(const Vec3f &other) {

v[0] += other.v[0];

v[1] += other.v[1];

v[2] += other.v[2];

return \*this;

}

const Vec3f &Vec3f::operator-=(const Vec3f &other) {

v[0] -= other.v[0];

v[1] -= other.v[1];

v[2] -= other.v[2];

return \*this;

}

float Vec3f::magnitude() const {

return sqrt(v[0] \* v[0] + v[1] \* v[1] + v[2] \* v[2]);

}

float Vec3f::magnitudeSquared() const {

return v[0] \* v[0] + v[1] \* v[1] + v[2] \* v[2];

}

Vec3f Vec3f::normalize() const {

float m = sqrt(v[0] \* v[0] + v[1] \* v[1] + v[2] \* v[2]);

return Vec3f(v[0] / m, v[1] / m, v[2] / m);

}

float Vec3f::dot(const Vec3f &other) const {

return v[0] \* other.v[0] + v[1] \* other.v[1] + v[2] \* other.v[2];

}

Vec3f Vec3f::cross(const Vec3f &other) const {

return Vec3f(v[1] \* other.v[2] - v[2] \* other.v[1],

v[2] \* other.v[0] - v[0] \* other.v[2],

v[0] \* other.v[1] - v[1] \* other.v[0]);

}

Vec3f operator\*(float scale, const Vec3f &v) {

return v \* scale;

}

ostream &operator<<(ostream &output, const Vec3f &v) {

cout << '(' << v[0] << ", " << v[1] << ", " << v[2] << ')';

return output;

}

**BAB III**

**3.1 Kritik dan saran**

Program yang kami guna mengerjakan tugas besar ini sangatlah jauh dari kata sempurna. Maka dari itu penulis sangat menerima kritik dan saran yang akan diberikan oleh para pembaca kepada kami. Agar harapan kami kedepannya bisa membuat program yang lebih baik lagi dari yang kami buat.

Adapun saran dari penulis adalah sebagai berikut :

1. Sebagian texture kurang lembut dan bahkan terlalu lembut sehingga mengurangi efek nyata pada tampilan program
2. Masih ada beberapa objek yang belum dimasukan.
3. Tidak semua fungsi-fungsi yang ada pada *OpenGL* kami masukan pada program ini.