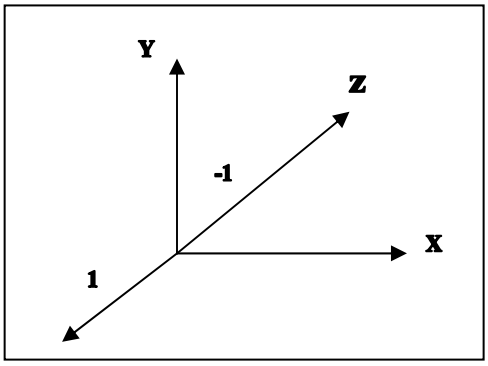
2. TEORI PENUNJANG

Pada bab ini akan dijelaskan mengenai teori penunjang yang digunakan dalam pembuatan Tugas Akhir.

2.1 *3D Computer Graphics*

3D Computer Graphics merupakan pengembangan dari *2D Computer Graphics*. Perbedaan antara penggambaran obyek dua dimensi dan obyek tiga dimensi yaitu pada sistem koordinat yang digunakan. Jika dalam penggambaran obyek secara dua dimensi menggunakan sistem koordinat x dan y yang mewakili lebar dan tinggi maka penggambaran obyek secara tiga dimensi menggunakan sistem koordinat x,y dan z dimana z mewakili kedalaman dari obyek. Dengan adanya penggambaran kedalaman dari suatu obyek maka obyek yang digambar akan terlihat mendekati aslinya.



Gambar 2.1. Sistem Koordinat 3 Dimensi

Dalam pembuatan Tugas Akhir ini aplikasi dibuat dengan menggunakan *OpenGL* (*Open Graphics Library*) untuk membuat obyek tiga dimensi. *OpenGL* adalah suatu API (*Application Programming Interface*) yang digunakan untuk membuat aplikasi *Computer Graphics* secara dua dimensi maupun tiga dimensi. Berikut ini akan dijelaskan beberapa hal yang digunakan untuk menampilkan dan membuat suatu obyek tiga dimensi beserta fungsi *OpenGL* yang digunakan:

2.1.1 Proyeksi pada Obyek Tiga Dimensi

Proyeksi pada obyek tiga dimensi digunakan untuk menampilkan obyek tiga dimensi pada bidang dua dimensi karena layar pada komputer menggunakan sistem dua dimensi. Ada dua macam proyeksi yaitu *parallel projection* dan *perspective projection*[1]. *Parallel projection* dilakukan dengan cara menarik garis lurus pada setiap koordinat obyek sampai memotong bidang xy pada layar sesuai dengan sudut pandang pengamat. *Perspective projection* menggunakan satu titik pusat sebagai sudut pandang, semua koordinat obyek yang tampak ditarik garis lurus menuju titik pusat sehingga akan menghasilkan gambar pada bidang xy. Dalam *OpenGL* ada beberapa *function* yang digunakan untuk melakukan proyeksi tiga dimensi salah satunya adalah menggunakan *function* gluPerspective:

- void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar).

fovy = sudut pandang pengamat.
aspect = perbandingan antara tinggi dan lebar dari layar.
= lebar layar / tinggi layar.
zNear = jarak terdekat ke layar.
zFar = jarak terjauh ke layar

2.1.2 Transformasi Tiga Dimensi

Transformasi tiga dimensi merupakan perubahan bentuk maupun perubahan posisi dari obyek yang disebabkan oleh proses translasi, skala dan rotasi. Masing-masing akan dijelaskan sebagai berikut:

- Translasi, yaitu memindahkan posisi obyek dari posisi awal menuju posisi tertentu sejauh x terhadap sumbux, sejauh y terhadap sumbu y dan sejauh z terhadap sumbu z.

  void glTranslatef(GLfloat x, GLfloat y, GLfloat z).
- Skala, yaitu melakukan skala pada obyek menjadi lebih besar atau lebih kecil. *Scaling* dilakukan berdasarkan perkalian sebesar x terhadap ukuran x semula,

[1] Harrington, Steven. "*Computer Graphics A Programming Approach"*.Mc-Graw-Hill,Inc

perkalian sebesar y terhadap ukuran y semula dan perkalian sebesar z terhadap ukuran z semula.

void glScalef(GLfloat x,GLfloat y, GLfloat z).

- Rotasi, yaitu memutar obyek searah maupun berlawanan arah jarum jam. Rotasi dapat dilakukan dengan memutar pada sumbu x, y maupun z. Rotasi dilakukan berdasarkan besar sudut rotasi yang sudah ditentukan terhadap sumbu x,y atau z.

  glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z).

2.1.3 *Shading* dan *Lightning*

Shading dan *Lightning* merupakan salah satu teknik yang digunakan untuk melakukan *rendering* pada obyek sehingga terlihat lebih realistis. *Shading* dilakukan supaya terjadi interaksi antara obyek dengan cahaya yang ada, sehingga bagian yang terkena cahaya akan terlihat lebih terang dari bagian yang lain dan selain itu obyek akan terlihat lebih halus. Ada beberapa macam *shading* yaitu *flat shading, smooth shading* dan *gouraud shading*.

Untuk mendapatkan intensitas warna dari obyek yang akan terkena cahaya maka diperlukan vektor normal dari sisi obyek yang terkena cahaya. Pada obyek *mesh* sisi dari obyek dibentuk oleh kumpulan *polygon* segitiga sehingga setiap sisi atau *face* dari obyek memiliki tiga titik atau sering disebut dengan *vertex* yang membentuk *face* tersebut. Untuk mendapatkan vektor normal dari *face* yang terkena cahaya digunakan rumus *cross product* pada *vertex-vertex face* tersebut. Persamaan untuk mendapatkan vektor normal dari *face* adalah sebagai berikut:

vektor normal *vertex* a = a(ax,ay,by)
vektor normal *vertex* b = b(bx,by,bz)
vektor normal *vertex* c = c(cx,cy,cz)
vektor normal *face* abc = (a + b + c) / 3 (2.1)

Function yang digunakan pada *OpenGL*:

- glEnable(GL_NORMALIZE) : untuk melakukan normalisasi vektor-vektor yang sudah didapat dari *cross product*.
- void glShadeModel(GLenum mode) : mode *shading* yang akan dipakai

- void glLightfv(GLenum light, GLenum pname, const GLfloat *params) : menentukan parameter cahaya yang akan digunakan.
- void glColorMaterial(GLenum face, GLenum mode) : memberikan warna material pada obyek.

2.1.4 Obyek *Mesh*

Obyek *mesh* merupakan suatu obyek yang terdiri dari kumpulan *polygon-polygon* yang membentuk suatu bentuk geometri. Biasanya *polygon* yang digunakan untuk membentuk suatu *mesh* obyek berbentuk segitiga. *Polygon* merupakan suatu kurva tertutup yang terdiri dari tiga atau lebih garis yang membentuk suatu bidang. *Polygon* dibentuk oleh titik-titik koordinat yang disebut *vertex* yang dihubungkan dengan *2 vertex* lain sehingga akan membentuk segitiga. *Polygon* yang terbentuk dinamakan *face* dari obyek tersebut. Jika *polygon-polygon* yang terhubung membentuk suatu obyek dan *polygon* tersebut tidak diberi warna hanya terdiri dari garis-garis yang terhubung maka *polygon-polygon* tersebut membentuk *wireframe* dari obyek. Jadi *wireframe* merupakan gambaran obyek tiga dimensi yang memperlihatkan garis-garis atau *edges* yang membentuk obyek tersebut.

Function yang digunakan untuk membuat obyek *mesh*:

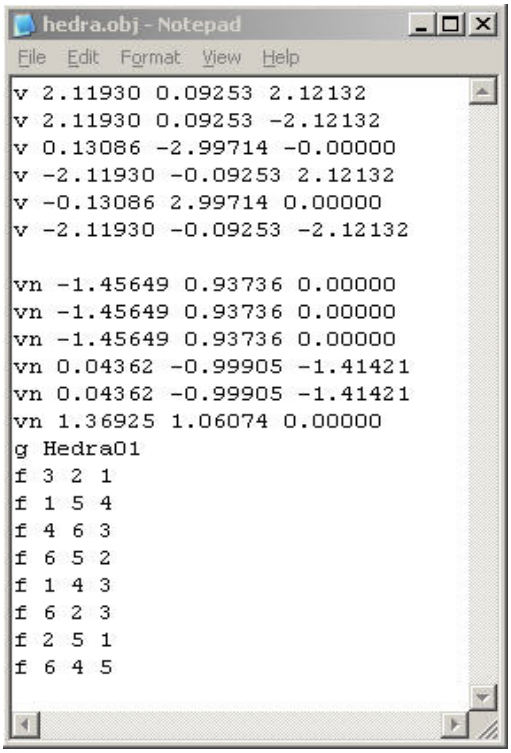
- void glVertex3f(GLfloat x, GLfloat y, GLfloat z) : koordinat 3D dari *vertex*
- void glNormal3f(GLfloat nx, GLfloat ny, GLfloat nz) : normal dari *vertex*
- glBegin(GL_*POLYGON*) : untuk membuat *polygon*
- void gl*Polygon*Mode(GLenum face, GLenum mode) : menentukan jenis *polygon* yang dibuat yaitu *lines* atau *fill*

2.2 Format *File* .obj

Dalam pembuatan obyek tiga dimensi informasi mengenai obyek seperti *vertex*, tekstur, warna, normal dari *vertex*, *face* dapat disimpan dalam suatu *file* tersendiri. Tujuannya adalah supaya mempermudah dalam penggambaran obyek tersebut karena kita hanya perlu membuat suatu *procedure* yang berfungsi mengambil dan menyimpan informasi dari obyek tersebut. Salah satu format *file* yang paling mudah digunakan adalah dengan menggunakan format *file* .obj karena

dalam penulisannya menggunakan sistem ASCII sehingga lebih mudah dalam pembacaan data yang ada. Format *file* .obj dapat dibuat dan dibuka dengan menggunakan fasilitas *notepad* yang telah disediakan oleh sistem operasi *Microsoft Windows*.

Format penulisan *file* .obj dapat dijelaskan sebagai berikut:



Gambar 2.2 Contoh Format *File* .obj

Gambar 2.2 merupakan contoh format *file* .obj dari obyek hedra. Setiap baris yang diawali dengan huruf "v" menunjukkan koordinat x,y dan z dari *vertex* yang ada pada obyek. Baris yang diawali dengan "vn" menunjukkan vektor normal dari *vertex-vertex* yang ada pada obyek. Baris yang diawali dengan menggunakan huruf "f" menunjukkan *face* pada obyek. *Face* obyek dibentuk dari tiga buah *vertex* pada obyek, sebagai contoh penulisan "f 3 2 1" menandakan *face* tersebut dibentuk dari *vertex* 3, 2 dan 1.

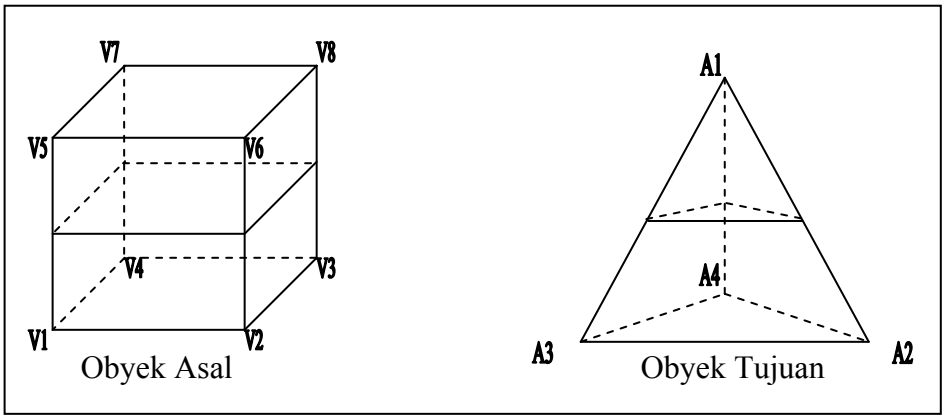
2.3 *Morphing*

Metamorfosis atau *morphing* tiga dimensi merupakan salah satu teknik yang merealisasikan perubahan atau transformasi bentuk suatu obyek menjadi bentuk obyek yang lain. Permasalahan utama dalam melakukan proses *morphing*

pada obyek adalah menentukan korespondensi permukaan obyek asal dengan obyek tujuan. Tujuan dari mencari korespondensi permukaan atau *vertex* antar obyek asal dengan obyek tujuan adalah untuk melakukan *mapping* setiap *vertex* dari obyek asal dengan obyek tujuan. Dalam proses *morphing,* posisi *vertex* pada obyek asal akan berpindah menuju posisi *vertex* pada obyek tujuan sehingga menyebabkan perubahan bentuk pada obyek asal sampai menjadi bentuk objek tujuan. Berikut ini akan dijelaskan metode yang digunakan untuk mencari korespondensi *vertex* obyek asal dengan obyek tujuan dan algoritma yang digunakan untuk melakukan proses *morphing*.

2.3.1 Pembagian Area Obyek

Untuk mencari korespondensi *vertex* antar obyek asal dengan obyek tujuan maka dilakukan pembagian obyek asal dan obyek tujuan menjadi beberapa area. Pembagian dilakukan berdasarkan koordinat x,y dan z dari obyek. Pembagian area obyek dilakukan dengan cara membagi dua koordinat sumbu x,y dan z pada obyek. Pembagian area obyek dibagi berdasarkan koordinat terbesar dan terkecil sumbu x, y dan z pada masing-masing obyek. Pembagian area obyek dilakukan dengan tujuan supaya dalam menentukan korespondensi *vertex* obyek asal dengan tujuan dilakukan dalam area yang lebih kecil. Jadi *vertex* yang ada pada area pertama dari obyek asal akan berkorespondensi dengan *vertex* yang ada pada obyek tujuan yang juga ada pada area pertama. Pembagian pada Gambar 2.3 dilakukan pembagian area obyek asal dan obyek tujuan menjadi dua bagian.



Gambar 2.3. Pembagian Obyek Menjadi Dua Bagian Berdasarkan Sumbu Y

Pada Gambar 2.3. terdapat dua buah obyek yang memiliki bentuk yang berbeda dan juga jumlah *vertex* yang berbeda. Dengan melakukan pembagian seperti pada Gambar 2.3. diharapkan *vertex*-*vertex* yang ada pada grup 1 obyek asal akan menjadi *vertex* pada grup 1 obyek tujuan sehingga hasil perubahan nantinya yang akan terjadi adalah *vertex* V5,V6,V7 dan V8 akan membentuk *vertex* A1, sedangkan *vertex* V1,V2,V3,V4 akan membentuk *vertex* A2,A3 dan A4.

Untuk menentukan *mapping vertex* obyek asal dan obyek tujuan maka dicari jarak terdekat dari *vertex* pada obyek asal dan *vertex* pada obyek tujuan pada masing-masing grup. Perubahan bentuk yang terjadi akan lebih baik jika kedua obyek memiliki topologi bentuk yang hampir sama atau obyek asal memiliki jumlah *vertex* yang lebih banyak dari jumlah *vertex* obyek tujuan. Pembagian tidak hanya sebatas pada pembagian menjadi dua bagian tetapi dapat dibagi menjadi lebih kecil lagi tetapi disesuaikan juga dengan banyaknya *vertex* pada masing-masing obyek. Setelah didapatkan korespondensi antar *vertex* maka langkah selanjutnya adalah melakukan proses *morphing*.

2.3.2 Algoritma *Liniear Interpolation*

Algoritma *liniear interpolation* digunakan untuk melakukan proses *morphing* setelah *vertex* pada obyek asal sudah memiliki korespondensi pada *vertex* obyek tujuan. Algoritma *liniear interpolation* digunakan untuk mendapatkan posisi *vertex* dari obyek asal menuju obyek tujuan. Persamaan *liniear interpolation* dapat dilihat pada persamaan 2.2.

$$\mathbf{V_i(t) = (1 - t)\, V_1 + t\, V_2;} \tag{2.2}$$

V_i = *vertex* dari *intermediate* obyek

V_1 = *vertex* obyek asal

V_2 = *vertex* obyek tujuan

t = waktu

Intermediate obyek adalah obyek yang terbentuk antara perubahan obyek asal menjadi obyek tujuan. *Intermediate* obyek merupakan bentuk transisi dari obyek asal sampai menjadi bentuk obyek tujuan. Waktu pada algoritma *liniear interpolation* menunjukkan perubahan bentuk yang terjadi pada saat waktu t.

2.4 3D Studio Max

3D Studio Max merupakan *software* bantu yang digunakan untuk membuat obyek tiga dimensi. Untuk mendapatkan format *file* .obj dari obyek yang dibuat dengan menggunakan 3D Studio Max dapat dilakukan dengan cara sebagai berikut:

- Obyek yang telah dibuat disimpan dalam format *file* .3ds dengan cara pada menu File pilih Export kemudian simpan dalam bentuk .3ds.
- Setelah disimpan dengan menggunakan format *file* .3ds, lakukan konversi dari format .3ds menjadi format .obj dengan menggunakan *software* bantu. Pada pembuatan aplikasi ini digunakan *software* bantu WCVT2POV.
- Setelah dilakukan konversi dengan menggunakan WCVT2POV akan didapatkan format *file* .obj seperti pada Gambar 2.2.