ANALISA TEKNIK CUBE MAPPING BERBASIS OPENGL

Yohan Angjaya
A11.2006.02722
Program Studi Ilmu Komputer
Universitas Dian Nuswantoro Semarang

ABSTRAK
Tulisan ini berisi tentang analisa Cube Mapping berbasis OpenGL yang telah dipelajari oleh peneliti-peneliti sebelumnya. Cube Mapping merupakan metode environment mapping selain sphere mapping dan saat ini banyak digunakan karena keefisienan dan kemudahannya untuk diimplementasikan. Cube Mapping memiliki kelebihan tersendiri dari pendahulunya, Sphere Mapping, yaitu kemampuan dalam memberikan data visual berupa 6 gambar berbeda yang seolah ditempelkan ke 6 sisi persegi sebuah kubus. Hal ini tidak akan menimbulkan efek distorsi pada gambar texture seperti yang sering terjadi pada Sphere Mapping jika pembuat model kurang berhati-hati. Selain itu, gambar texture pada Cube Mapping dapat memiliki detil yang lebih dalam dan realistik untuk sebuah virtual environment

Kata Kunci : komputer grafik, model 3D, texture mapping, cube mapping, OpenGL

1. PENDAHULUAN

Dalam komputer grafik, mapping adalah sebutan untuk memetakan texture ke permukaan objek. Teknik-teknik mapping secara umum dinamakan dengan texture mapping, yaitu suatu metode untuk menambahkan detail, tekstur permukaan (baik bitmap maupun raster image), atau warna untuk model komputer yang dihasilkan grafis atau 3D. Pemberian texture adalah merupakan upaya untuk mendapatkan efek realitas pada objek 3D.

Selain dengan menggunakan texture, adanya efek cahaya dan bayangan juga berperan besar untuk menambah kesan realistis sebuah object atau model. Oleh karena itu teknik penambahan texture erat hubungannya dengan teknik pencahayaan. Reflection Mapping atau Environment Mapping adalah teknik pemberian texture dan cahaya berbasis gambar yang efisien saat digunakan pada model 3D.

Cube Mapping adalah salah satu dari Environment Mapping yang berupa perkembangan lebih lanjut dari Sphere Mapping. Cube Mapping membuat sebuah obyek merepresentasikan lingkungan sekitarnya dengan cara menempatkan enam buah gambar 2D yang berbeda di keenam sisi obyek. Hal ini membuat obyek seolah-olah memiliki enam sisi pantul, yaitu depan, belakang, kanan, kiri, atas dan bawah. Dengan demikian Cube Mapping

tidak memiliki tampilan yang terdistorsi seperti yang terjadi pada Sphere Mapping.

OpenGL sebagai perangkat lunak yang memiliki kumpulan library, fungsi dan prosedur di bidang desain 3D sudah mendukung Cube Mapping sebagai salah satu metode Environment Mapping. OpenGL dan kumpulan library untuk Cube Mapping yang dimilikinya telah banyak berjasa dalam dunia permodelan untuk membuat sebuah model grafis 3D yang lebih realistis.

2. ENVIRONMENT MAPPING

Di bidang grafika komputer, Environment Mapping adalah teknik untuk mensimulasikan sebuah obyek agar dapat merefleksikan lingkungan (*environment*) sekitarnya. Pada bentuk yang paling sederhana, teknik ini biasanya memakai obyek yang permukaannya terlihat seperti krom. Teknik ini dicetuskan pertama kali oleh James Blinn dan Martin Newell pada tahun 1976 setelah menyempurnakan teori yang dicetuskan oleh Edwin Catmull 2 tahun sebelumnya.

Seiring dengan perkembangan teori Environment Mapping ini, muncullah beberapa metode yang banyak dipakai, yaitu :

2.1 Sphere Mapping

Dicetuskan oleh Gene Miller pada tahun 1982 di MAGI Synthavision. Sphere Mapping seolah-olah menggunakan sebuah bola kaca yang memantulkan lingkungan sekitarnya seperti dikelilingi oleh tembok berbentuk kubah yang sangat jauh. Gambar lingkungan ini disimpan sebagai tekstur yang menggambarkan semua sudut kecuali area yang berada tepat dibelakang bola.

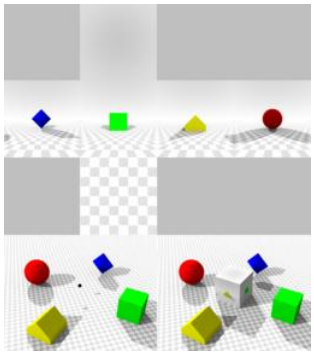
Berikut ini adalah penggambaran eksperimen yang dilakukan oleh Gene Miller. Miller menggunakan bola kaca sebagai media pemantulan gambar environment yang lalu digunakan sebagai tekstur di model 3D berbentuk anjing.

Gambar 1. Eksperimen Gene Miller

2.2 Cube Mapping

Cube Mapping adalah metode yang menggunakan enam sisi sebuah kubus sebagai bentuk dasar pemetaan. Gambar lingkungan diproyeksikan ke enam permukaan kubus dan disimpan dalam bentuk 6 gambar yang berbeda dari 6 sudut pandang.

Cube Mapping masih merupakan metode mapping yang paling banyak dipakai hingga sekarang. Karena selain menutupi kelemahan yang ada pada Sphere Mapping seperti keterbatasan sudut pandang, distorsi gambar dan titik buta, Cube Mapping juga menyediakan solusi efisien untuk mengaplikasikan pencahayaan dan hanya membutuhkan 1 kali rendering (dimana Sphere Mapping harus melakukan render berulang-ulang saat sudut pandang berubah). Selain itu, Cube Mapping juga tidak memerlukan perangkat keras yang sangat kuat seperti Ray Tracing, jadi Cube Mapping bisa digunakan oleh lebih banyak orang.



Gambar 2. Refleksi dengan Cube Map

Jika Cube Mapping memiliki kekurangan, hal itu adalah pada saat perlu menambahkan objek atau sumber cahaya baru, maka harus melakukan render ulang. Juga harus me-render ulang saat objek tersebut bergerak melalui area tertentu. Tapi hal itu tidak terlalu bermasalah jika menggunakan Cube Mapping pada benda-benda mati yang tidak perlu banyak bergerak, misalnya bebatuan, rumah atau pohon.

3. PEMBAHASAN

3.1. Mempersiapkan Array CubeMap

Seperti yang sudah dibahas sebelumnya, ide dasar Cube Mapping adalah memetakan tekstur ke enak sisi sebuah kubus. Untuk mencapai hal itu, OpenGL menggunakan perintah :

glBindTexture(GL_TEXTURE_CUBE_MAP_POSITIVE_X_EXT, CubeMap[0]);
glBindTexture(GL_TEXTURE_CUBE_MAP_NEGATIVE_X_EXT, CubeMap[1]);
glBindTexture (GL_TEXTURE_CUBE_MAP_POSITIVE Y EXT, CubeMap[2]);
glBindTexture(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y_EXT, CubeMap[3]);
glBlndTexture (GL_TEXTURE_CUBE_MAP POSITIVE_Z_EXT, CubeMap[4]);
glBindTexture(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z_EXT, CubeMap[5]);

Penjelasan :
glBindTexture : Mengaktifkan penggunaan tekstur.
GL_TEXTURE_CUBE_MAP_POSITIVE{NEGATIVE)_X{Y,Z)_EXT: Menempatkan gambar pada sumbu X/Y/Z positif/negatif dan akhiran EXT menandakan bahwa perintah ini lintas platform.
CubeMap[0..5] : Array tekstur atau gambar yang digunakan, sejumlah 6 buah.

3.2. Mengaktifkan Fitur Cube Mapping

Untuk mengaktifkan fitur Cube Mapping pada OpenGL, perlu digunakan perintah

glEnable (GL_TEXTURE_CUBE_MAP_EXT);

Dan untuk me-nonaktifkan fitur ini, digunakan perintah

glDisable(GL_TEXTURE_CUBE_MAP_EXT);

Apabila dinyalakan lebih dari satu tipe mapping, maka OpenGL akan menentukan fitur mana yang akan dipakai sesuai dengan prioritas. Prioritas tertinggi adalah Cube Mapping, diikuti oleh 3D lalu 2D dan 1D mapping.

3.3. Penempatan Koordinat Tekstur ke Vertex

Untuk penggunaan koordinat, tekstur dan obyek memiliki perbedaan. Obyek menggunakan koordinat kartesian seperti

pada umumnya yaitu +X, -X, +Y, -Y, +Z dan –Z, dan tekstur menggunakan koordinat STR yaitu +S, -S, +T, -T, +R dan –R. Untuk menggunakannya gunakan perintah :

glEnable (GL_TEXTURE_GEN_S);
glEnable (GL_TEXTURE_GEN_T);
glEnable (GL_TEXTURE_GEN_R);

Untuk me-nonaktifkan koordinat STR, cukup ganti glEnable menjadi glDisable. Tapi dinon-aktifkan koordinat ini setelah selesai membuat obyek.

3.4. Membuat Obyek

Proses pembuatan obyek dapat dipresentasikan berupa matrix yang disimpan dalam stack. Cara kerjanya adalah OpenGL menganggap semua masukan pasangan vertex sebagai matrix dan ditampung pada sebuah stack (penampung). Oleh karena itu OpenGL memiliki perintah *glPushMatrix* saat obyek tidak ditampilkan dan *glPopMatrix* saat ingin menampilkan obyek. Kedua perintah ini biasanya didampingi oleh perintah *SwapBuffer* atau *glFlush* sebagai eksekutornya.

Selanjutnya untuk menentukan posisi obyek, digunakan perintah

```
procedure glDraw();
var i : GLint;
begin
 glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT);
 glLoadIdentity();
 glTranslatef(0.0,0.0,-5);
 glPushMatrix;
 glRotatef(yAngle, 0.0, 1.0, 0.0);
```

Penjelasan

procedure glDraw() : nama prosedur yang dipanggil

var i :GLint; : mendefinisikan variable “i” dengan tipe GLint

glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT) : mengosongkan layar dan Depth Buffer

glLoadIdentity() : mengembalikan posisi kamera ke default.

glTranslatef(0.0, 0.0, 0.-5) : semakin besar nilai pada sumbu Z, maka semakin beasr juga gambar yang terlihat. Tapi jika nilai sumbu Z bernilai positif, maka set obyek akan terlihat kosong karena obyek berada diluar jangkauan layar. Sebaliknya jika nilai Z terlalu kecil maka obyek akan terlihat sangat kecil (jauh).

Untuk membentuk obyek, gunakan perintah-perintah seperti berikut ini

```
//untuk membuat bola dengan diameter 0,8, jumlah irisan 32 dan sisi 32.
gluSphere(SphereQuadratic, 0.8, 32, 32);

//membuat sebuah kotak
glBegin(GL_QUADS);

// sisi depan
glNormal3f( 0.0, 0.0, 1.0);
glVertex3f(-0.25, -1.0,  1.0);
glVertex3f( 0.25, -1.0,  1.0);
glVertex3f( 0.25,  1.0,  1.0);
glVertex3f(-0.25,  1.0,  1.0);
```

```
// sisi belakang
    glNormal3f( 0.0, 0.0,-1.0);
    glVertex3f(-0.25, -1.0, -1.0);
    glVertex3f(-0.25,  1.0, -1.0);
    glVertex3f( 0.25,  1.0, -1.0);
    glVertex3f( 0.25, -1.0, -1.0);

// sisi atas
    glNormal3f( 0.0, 1.0, 0.0);
    glVertex3f(-0.25,  1.0, -1.0);
    glVertex3f(-0.25,  1.0,  1.0);
    glVertex3f( 0.25,  1.0,  1.0);
    glVertex3f( 0.25,  1.0, -1.0);

// sisi bawah
    glNormal3f( 0.0,-1.0, 0.0);
    glVertex3f(-0.25, -1.0, -1.0);
    glVertex3f( 0.25, -1.0, -1.0);
    glVertex3f( 0.25, -1.0,  1.0);
    glVertex3f(-0.25, -1.0,  1.0);

// sisi kanan
    glNormal3f( 1.0, 0.0, 0.0);
    glVertex3f( 0.25, -1.0, -1.0);
    glVertex3f( 0.25,  1.0, -1.0);
    glVertex3f( 0.25,  1.0,  1.0);
    glVertex3f( 0.25, -1.0,  1.0);

// sisi kiri
    glNormal3f(-1.0, 0.0, 0.0);
    glVertex3f(-0.25, -1.0, -1.0);
    glVertex3f(-0.25, -1.0,  1.0);
    glVertex3f(-0.25,  1.0,  1.0);
    glVertex3f(-0.25,  1.0, -1.0);
    glEnd();
```

Penjelasan

glBegin(Quads) : untuk memulai fungsi menggambar kotak

glNormal3f (x, y, z) : menetapkan sisi yang sedang digambar, apakah berada di koordinat x positif/negatif, y positif/negatif atau zpositif/negatif.

glVertex3f (x, y, z) : menentukan koordinat titik-titik ujung kubus.

glEnd() : mengakhiri fungsi.

3.5. Inisialisasi OpenGL

Untuk memulai OpenGL, harus dilakukan inisialisasi menggunakan prosedur *glInit* yang berisi perintah-perintah berikut:

```
procedure glInit();
begin
glClearColor(0.0, 0.0, 0.0, 0.0);
glShadeModel(GL_SMOOTH
glClearDepth(1.0);
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LESS);
glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
glEnable(GL_TEXTURE_2D);
```

Penjelasan:

Procedure glInit() : memanggil prosedur glInit

glClearColor (0.0, 0.0, 0.0, 0.0) : Menentukan warna latar belakang dan dengan nilai tersebut, akan didapatkan warna hitam. Bentuk umum perintah ini adalah *glClearColor (RedIndex, GreenIndex, BlueIndex, AlphaIndex).*

glShadeModel (GL_SMOOTH) : Untuk menentukan tipe bayangan, dalam hal ini yang dipilih adalah tipe smooth.

glClearDepth (1,0) : setting Depth Buffer (buffer kedalaman)

glEnable (GL_DEPTH_TEST) : untuk mengaktifkan memory Depth Buffer.

glDepthFunc(GL_LESS) : tipe Depth Test yang akan dipakai

glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST) : untuk melakukan kalkulasi perspektif jarak, memakai yang paling bagus.

glEnable(GL_TEXTURE_2D) : menyalakan fitur texture mapping 2D.

Bentuk dasar objek 3D yang akan penulis gunakan sudah dapat dilihat seperti pada gambar 4.8. Tapi karena belum diberi tekstur, maka objek tersebut hanya berupa bongkahan putih.

3.6. Penempatan Tekstur di Cube Map

Untuk menentukan gambar mana saja yang akan menempati sisi kubus, digunakan perintah sebagai berikut :

Berikutnya untuk pemuatan gambar latar back.jpg dan gambar-gambar tekstur xpos.jpg, xneg.jpg, ypos.jpr, yneg.jpg, zpos.jpg zneg.jpg, digunakan perintah-perintah berikut yang sudah dibahas sebelumnya:

```
LoadTexture('back.jpg',Texture,false);
LoadTexture('xpos.jpg',CubeMap[0],false,GL_LINEAR_MIPMAP_LINEAR,GL_LINEAR, GL_TEXTURE_CUBE_MAP_POSITIVE_X_EXT);
LoadTexture('xneg.jpg',CubeMap[1],false,GL_LINEAR_MIPMAP_LINEAR,GL_LINEAR, GL_TEXTURE_CUBE_MAP_NEGATIVE_X_EXT);
LoadTexture('ypos.jpg',CubeMap[2],false,GL_LINEAR_MIPMAP_LINEAR,GL_LINEAR, GL_TEXTURE_CUBE_MAP_POSITIVE_Y_EXT);
LoadTexture('yneg.jpg',CubeMap[3],false,GL_LINEAR_MIPMAP_LINEAR,GL_LINEAR, GL_TEXTURE_CUBE_MAP_NEGATIVE_Y_EXT);
LoadTexture('zpos.jpg',CubeMap[4],false,GL_LINEAR_MIPMAP_LINEAR,GL_LINEAR, GL_TEXTURE_CUBE_MAP_POSITIVE_Z_EXT);
LoadTexture('zneg.jpg',CubeMap[5],false,GL_LINEAR_MIPMAP_LINEAR,GL_LINEAR, GL_TEXTURE_CUBE_MAP_NEGATIVE_Z_EXT);
```

Penjelasan :

LoadTexture (‘back.jpg’,Texture, False) : untuk memuat gambar back.jpg sebagai gambar latar belakang.

LoadTexture ('xpos.jpg’,CubeMap[0], false, GL LINEAR MIPMAP LINEAR, GL LINEAR, GL TEXTURE CUBE MAP POSITIVE X EXT) : untuk memuat gambar xpos.jpg pada array CubeMap[0].

LoadTexture ('xneg.jpg’,CubeMap[1], false, GL LINEAR MIPMAP LINEAR, GL LINEAR, GL TEXTURE CUBE MAP NEGATIVE X EXT) : untuk memuat gambar xneg.jpg pada array CubeMap[1].

LoadTexture ('ypos.jpg’,CubeMap[2], false, GL LINEAR MIPMAP LINEAR, GL LINEAR, GL TEXTURE CUBE MAP POSITIVE Y

EXT) : untuk memuat gambar ypos.jpg pada array CubeMap[2].

LoadTexture ('yneg.jpg',CubeMap[3], false, GL LINEAR MIPMAP LINEAR, GL LINEAR, GL TEXTURE CUBE MAP NEGATIVE Y EXT) : untuk memuat gambar yneg.jpg pada array CubeMap[3].

LoadTexture ('zpos.jpg',CubeMap[4], false, GL LINEAR MIPMAP LINEAR, GL LINEAR, GL TEXTURE CUBE MAP POSITIVE Z EXT) : untuk memuat gambar zpos.jpg pada array CubeMap[4].

LoadTexture ('zneg.jpg',CubeMap[5], false, GL LINEAR MIPMAP LINEAR, GL LINEAR, GL TEXTURE CUBE MAP NEGATIVE Z EXT) : untuk memuat gambar zneg.jpg pada array CubeMap[5].

Perintah-perintah diatas pada dasarnya adalah penempatan file gambar yang digunakan sebagai tekstur Cube Mapping pada array bernama CubeMap. Fungsi *Loadtexture* sebenarnya bukanlah fungsi rutin bawaan OpenGL, tapi adalah fungsi yang ada pada file Texture.pas. dengan syarat pemanggilan fungsi tersebut menggunakan perintah *glEnable (GL_TEXTURE_2D);* karena yang digunakan adalah tekstur 2D.

Dengan melihat urutan perintah diatas, maka akan diperoleh sebuah kubus yang mengitari objek 3D. Lalu penulis mendefinisikan koordinat S, T, R untuk menghasilkan sisi Reflection Map

```
glTexGeni(GL_S,
GL_TEXTURE_GEN_MODE,
GL_REFLECTION_MAP_EXT);
glTexGeni(GL_T,
GL_TEXTURE_GEN_MODE,
GL_REFLECTION_MAP_EXT);
glTexGeni(GL_R,
GL_TEXTURE_GEN_MODE,
GL_REFLECTION_MAP_EXT);
glTexParameteri(GL_TEXTURE_CUBE_M
AP_EXT,        GL_TEXTURE_WRAP_S,
GL_CLAMP_TO_EDGE);

glTexParameteri(GL_TEXTURE_CUBE_M
AP_EXT,        GL_TEXTURE_WRAP_T,
GL_CLAMP_TO_EDGE);
```

Perlu juga membuat permukaan bola sebagai sisi pantul dengan mnggunakan perintah:

```
SphereQuadratic          :=
gluNewQuadric();
gluQuadricNormals(SphereQuadr
atic, GLU_SMOOTH);
```

Penjelasan :

SphereQuadratic := gluNewQuadric() : membuat pointer ke objek bulat, kembalikan nilai 0 jika tidak ada memori

gluQuadricNormals(SphereQuadratic, GLU_SMOOTH) : membuat permukaan normal yang mulus.

3.7. Animasi

Selanjutnya untuk merotasi kubus, digunakan perintah

glRotatef(yAngle, 0.0, 1.0, 0.0);

Sumbu Y dimasukkan nilai 1 supaya kubus berotasi terhadap sumbu Y (horizontal) sebesar yAngle

Kecepatan rotasi kubus bisa diatur dengan menggunakan rumus

ySpeed := 0.1;

yAngle := yAngle + ySpeed

Dimana kecepatannya berubah-ubah sesuai nilai variable ySpeed yang bisa diatur dengan

procedure ProcessKeys;

begin

if (keys[VK_RIGHT]) then ySpeed := ySpeed + 0.002;

if (keys[VK_LEFT]) then ySpeed := ySpeed - 0.002;

end;

Dengan perintah tersebut, saat ditekan tombol arah kanan, maka kecepatan rotasi kubus akan bertambah sebanyak 0.002 fps, dan saat memencet tombol arah kiri, kecepatan rotasi akan berkurang sebanyak 0.002 fps. Arah rotasi bisa berubah arah (berputar ke kiri) jika variabel ySpeed mencapai nilai negatif.

User juga dapat mengubah ukuran windows dan mendapati ukuran gambar ikut menyesuaikan ukuran windows dengan menggunakan perintah :

if (Height : O) then Height := 1;

glViewport (0, 0, Width, Height);

glMatrixMode (GL PROJECTION);

glLoadldentity ( );

gluPerspective (45.0, Width/Height, 1.0, 100.0);

glMatrixMode (GL MODELVIEW);

glLoadldentity ( );

Penjelasan:

if (Height : O) then Height := 1 : untuk mencegah adanya pembagian oleh 0 yang menyebabkan error

glViewport (0, 0, Width, Height) : mengatur posisi pandangan untuk window OpenGL

glMatrixMode(GL_PROJECTION) : mengubah mode Matrix menjadi mode proyeksi

glLoadIdentity() : mengembalikan posisi kamera ke default

gluPerspective(45.0, Width/Height, 1.0, 100.0) : melakukan kalkulasi perspektif kejauhan.

glMatrixMode(GL_MODELVIEW) : mengembalikan mode Matirx ke mode Modelview

Perintah-perintah diatas akan terus dijalankan hingga user memencet tombol Esc dengan menggunakan perintah

If (keys [VK ESCAPE]) then finished := True

4. KESIMPULAN

Cube Mapping bukanlah teknik Texture Mapping yang sangat populer dengan tanpa alasan. Dibandingkan dengan teknik mapping lain seperti Sphere Mapping dan Paraboloid Mapping, Cube Mapping sangatlah sederhana tapi menghasilkan kualitas tekstur yang bagus. Hal ini dapat dilihat dari kebutuhan Cube Mapping yang hanya membutuhkan 1 unit tekstur dan 1 kali proses rendering. Selain itu, teknik Cube Mapping tidak mengurangi resolusi gambar tekstur seperti teknik yang lain (Sphere Mapping dapat mengurangi resolusi sampai dengan 78%)

5. SARAN

Setelah menyelesaikan penelitian ini, penulis memiliki saran yaitu menggunakan Cube Mapping disarankan untuk model 3D yang bentuknya sederhana. Jika memiliki bentuk yang lebih kompleks, lebih baik menggunakan teknik Polycube Mapping.

6. DAFTAR PUSTAKA

Arikunto, Suharsini. Manajemen Penelitian. Rineka Cipta, Jakarta, 2010.

Chang, Chin-Chen dan Chen-Yu Lin. Texture Tiling on 3D Models Using Automatic Polycube-maps and Wang Tiles. Journal of Information Science and Engineering, 26:291-305, 2008

Chopine, Ami. 3D Art Essentials. CRC Press, 2012

Hariwijaya, M dan Triton P.B. Pedoman Penulisan Ilmiah Skripsi dan Tesis, Oryza, Jakarta, 2011.

Hearn, Donald dan Baker. Computer Graphics with OpenGL Updated Version, 3rd Edition, Pearson Prentice Hall, 2004

Hill, F.S. Computer Graphics Using OpenGL. Prentice Hall Inc, 2001.

Kriglstein, Simone dan Gunter Wallner. Environment Mapping, 2001.

Kusrianto, Adi. Pengantar Desain Komunikasi Visual, Penerbit ANDI, Yogyakarta, 2007.

Martz, Paul. OpenGL Distilled. Addison-Wesley Professional, 2006.

Niebruegge, Bill dan Cindy M. Grimm. Continuous cube mapping. Journal of Graphics Tools, 2007.

NVIDIA Corporation. OpenGL Cube Map Texturing. http://www.nvidia.com/object/cube_map_ogl_tutorial.html, 1999

NVIDIA Developer Zone. Chapter 7, Environment Mapping Techniques. http://http.developer.nvidia.com/CgTutorial/cg_tutorial_chapter07.html, 2007

Tarini, Marco, Kai Hormann, Paolo Cignoni, dan Claudio Montani. Polycube Maps. ACM Trans. Graph., 2004.

Umar, Husein. Metode Penelitian untuk Skripsi dan Tesis Bisnis, Edisi Kedua, Rajawali Pers, Jakarta, 2011.