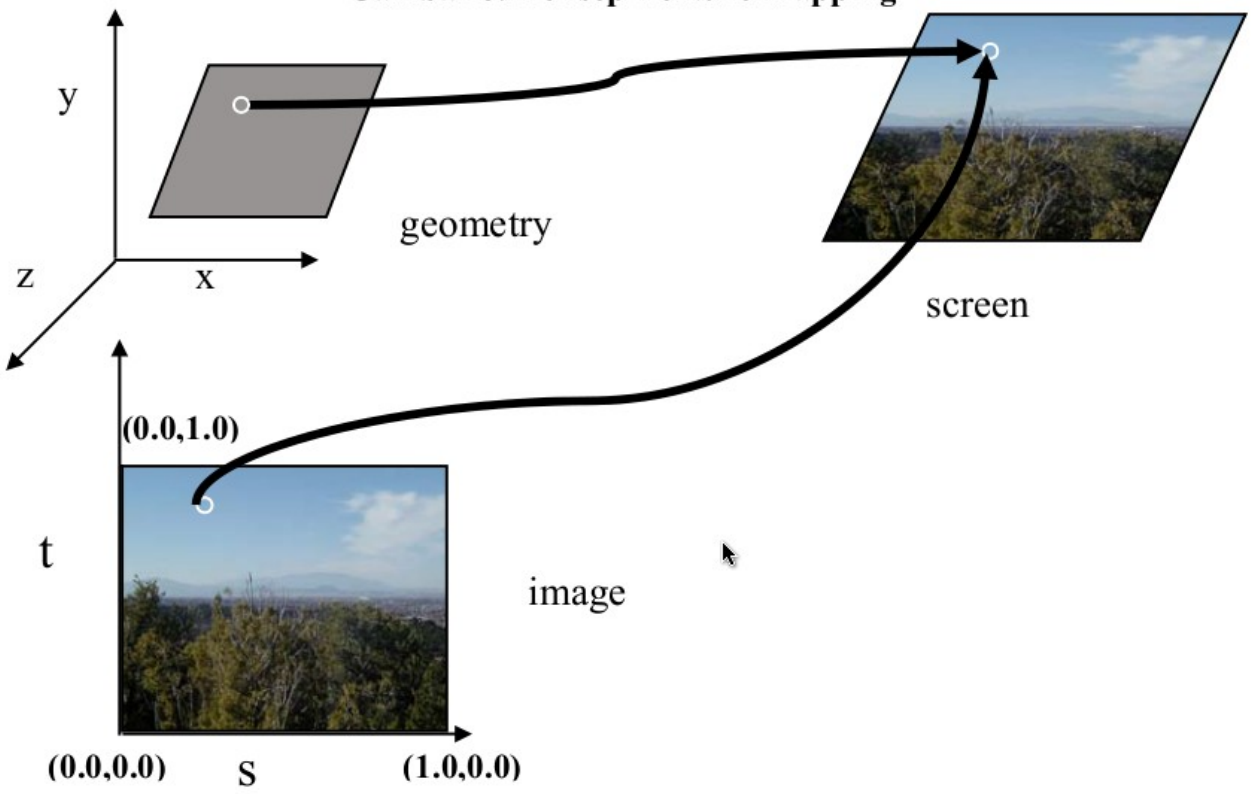
Tutorial 06.
Texture Mapping & Blending

Hingga tahap ini, geometric primitive digambar dengan warna solid atau warna hasil interpolasi warna-warna vertex-nya. Texture mapping memungkinkan untuk menaruh gambar pada geometric primitive tersebut dan sekaligus mengikuti transformasi yang diterapkan kepada polygon tersebut (cf. Gambar 8).

Gambar 8. Konsep Texture Mapping



Texture merupakan data segi-empat sederhana yang berada pada bidang texture. Bidang texture diwakili oleh dua sumbu koordinat yaitu sumbu s dan sumbu t. Setiap texture akan memenuhi bidang koordinat (0.0,0.0) sd. (1.0,1.0). Nilai individual dari array texture biasanya dikenal dengan istilah texels (texture pixels).

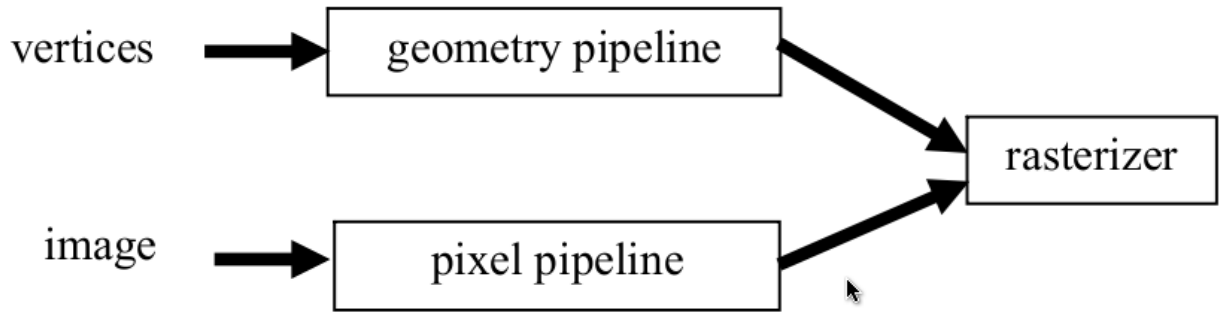
Yang membuat texture mapping sedikit rumit adalah bagaimana proses pemetaan antara bentuk segi-empat texture ke polygon menginngat secara umum bentuk poligon biasanya non-rectangular.
Beberapa contoh penggunaan texture mapping antara lain:

- mensimulasikan aspek visual dari material seperti tampakan kayu, batu bata, atau

granit
- mengurangi kompleksitas (jumlah polygon yang dibutuhkan) dari suatu obyek geometri.
- teknik pemrosesan citra seperti image warping dan rectification, rotation dan scaling
- mensimulasikan berbagai efek permukaan seperti efek reflektif seperti cermin atau lantai yang telah digosok mengkilat, efek tonjolan dll.

Salah satu keuntungan dari texture mapping adalah bahwa detail visual itu berada di citra bukan di geometri. Dan sekompleks apapun citra, selama tidak merubah ukuran citra, tidak berpengaruh pada kinerja keseluruhan, yaitu kompleksitas cari citra tidak berpengaruh kepada pipeline geometric (transformasi, clipping) dari OpenGL.

Texture ditambahkan saat rasterisasi ketika geometric pipeline dan pixel pipeline bertemu seperti diilustrasikan pada Gambar berikut.



Secara konseptual ada tiga langkah dalam melakukan texture mapping, yaitu:

1. Penentuan texture
    - Baca image dari file
    - Generate texture id untuk image tersebut glGenTextures(3, &texture[0])
2. Pemberian koordinat texture ke vertex
3. Penentuan parameter texture (wrapping / filtering)

```
// Texture Mapping
// Program ini memetakan sebuah texture pada permukaan object 3D
// Image Loadernya menggunakan library tambahan yaitu SOIL (Simple OpenGL Image Library)
// Untuk meangaktifkan state BLENDING uncomment perintah glEnable(GL_BLEND);

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>

#include <SOIL/SOIL.h>

#include <GL/glut.h>
```

```
float z_pos = -5.0f;
float xRot, yRot, zRot;
float rot = 0.0f;


GLfloat LightAmbient[] = {0.5f, 0.5f, 0.5f, 1.0f};
GLfloat LightDiffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
GLfloat LightPosition[] = {0.0f, 0.0f, 2.0f, 1.0f};

// storage for one texture
GLuint tex_2d;

/* storage for one texture  */
GLuint texture[1];


void init();
void myKeyboard(unsigned char, int, int);
void myDisplay(void);
void myTimeOut(int);
void resize(int, int);



int main(int argc, char* argv[])
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH);
   glutInitWindowSize(500, 500);
   glutInitWindowPosition(0, 0);
   glutCreateWindow("Texture Mapping");
   glutDisplayFunc(myDisplay);
   glutIdleFunc(myDisplay);
   glutKeyboardFunc(myKeyboard);
   glutReshapeFunc(resize);
   glutTimerFunc(100, myTimeOut, 0);
   init();
   glutMainLoop();

   return 0;
}

GLuint LoadGLTextures()                    // Load Bitmaps And Convert To Textures
{
   /* load an image file directly as a new OpenGL texture */
   tex_2d = SOIL_load_OGL_texture("background.bmp", SOIL_LOAD_AUTO,
SOIL_CREATE_NEW_ID, SOIL_FLAG_INVERT_Y);
```

```
    /* check for an error during the load process */
    if(tex_2d == 0)
    {
        printf( "SOIL loading error: '%s'\n", SOIL_last_result() );
    }

    // Typical Texture Generation Using Data From The Bitmap
    glBindTexture(GL_TEXTURE_2D, tex_2d);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);

    return tex_2d;                              // Return Success
}

void init()
{

    LoadGLTextures();
    glEnable(GL_TEXTURE_2D);
    glShadeModel(GL_SMOOTH);
    glClearColor(0.0f, 1.0f, 1.0f, 0.5f);
    glClearDepth(1.0f);
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LEQUAL);
    glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
    glLightfv(GL_LIGHT1, GL_AMBIENT, LightAmbient);
    glLightfv(GL_LIGHT1, GL_DIFFUSE, LightDiffuse);
    glLightfv(GL_LIGHT1, GL_POSITION, LightPosition);
    glEnable(GL_LIGHT1);
}

void myKeyboard(unsigned char key, int x, int y)
{
    switch(key)
    {
        case '<':
        case ',':
                z_pos -= 0.1f;
                break;
        case '>':
        case '.':
                z_pos += 0.1f;
                break;

        case 27:
                exit(0);
                break;
        default:
                break;
```

```
    }
}

void myDisplay(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);

    glLoadIdentity();
    glTranslatef(0.0f, 0.0f, z_pos);
    glRotatef(xRot,1.0f,0.0f,0.0f);             // Rotate On The X Axis
    glRotatef(yRot,0.0f,1.0f,0.0f);             // Rotate On The Y Axis
    glRotatef(zRot,0.0f,0.0f,1.0f);             // Rotate On The Z Axis

    //glEnable(GL_BLEND);             // enabling BLENDING state
    //glColor3f(0.0, 0.0, 1.0);         // Setting BLENDING color

    glBegin(GL_QUADS);
          // Front Face
       glNormal3f( 0.0f, 0.0f, 1.0f);
       glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
       glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f, 1.0f);
       glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f, 1.0f);
       glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f, 1.0f);
       // Back Face
       glNormal3f( 0.0f, 0.0f,-1.0f);
       glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
       glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);
       glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);
       glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f, -1.0f);
       // Top Face
       glNormal3f( 0.0f, 1.0f, 0.0f);
       glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);
       glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f,  1.0f,  1.0f);
       glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f,  1.0f,  1.0f);
       glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);
       // Bottom Face
       glNormal3f( 0.0f,-1.0f, 0.0f);
       glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
       glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f, -1.0f, -1.0f);
       glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);
       glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);
       // Right face
       glNormal3f( 1.0f, 0.0f, 0.0f);
       glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f, -1.0f);
       glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);
       glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f,  1.0f,  1.0f);
       glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);
       // Left Face
```

```
        glNormal3f(-1.0f, 0.0f, 0.0f);
        glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
        glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
        glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);
        glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);
    glEnd();
    glFlush();

    xRot+=0.1f;                         // X Axis Rotation
    yRot+=0.1f;                         // Y Axis Rotation
    zRot+=0.1f;                         // Z Axis Rotation
    glutSwapBuffers();


}
void myTimeOut(int id)
{

    rot += 5.0f;
    glutPostRedisplay();
    glutTimerFunc(100, myTimeOut, 0);
}

void resize(int width, int height)
{
    glViewport(0, 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (GLdouble)width / (GLdouble)height, 1.0, 300.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
```

Seringkali, efek yang diinginkan dapat diperoleh dengan mencampur lebih dari satu texture. Proses pencampuran lebih dari satu texture disebut dengan istilah blending. Salah satu efek blending yang paling sederhana adalah dengan memblending texture dengan warna.

Untuk mengaktifkan BLENDING state pada program diatas, buka comment pada statemen //glEnable(GL_BLEND); dan //glColor3f(0.0, 0.0, 1.0); pada fungsi void mydisplay() jika kita ingin melakukan pencampuran antara texture dan warna. Di sini texture di blend dengan warna biru.