MODUL 7
Material Texture

A. KOMPETENSI DASAR

- Memahami Inisialisasi Material Texture
- Memahami Texture Mapping.
- Memahami dasar menampilkan susunan obyek yang dilengkapi dengan texture mapping.

B. ALOKASI WAKTU

4 js (4x50 menit)

C. PETUNJUK

- Awali setiap aktivitas dengan do'a, semoga berkah dan mendapat kemudahan.
- Pahami Tujuan, dasar teori, dan latihan-latihan praktikum dengan baik dan benar.
- Kerjakan tugas-tugas dengan baik, sabar, dan jujur.
- Tanyakan kepada asisten/dosen apabila ada hal-hal yang kurang jelas.

D. DASAR TEORI

1. Pada konsep texturing, perlu dilakukan inisialisasi yang menghubungkan antara *source code* untuk texture mapping dengan program utama. *Source code* dan *header* yang digunakan pada praktikum texturing adalah **'imageloader.h' dan 'imageloader.cpp'**.
2. File **'imageloader.h' dan 'imageloader.cpp'** harus disimpan dalam folder project yang sedang digunakan.
3. Untuk menampilkan texture pada openGL, lakukan konfigurasi project texture mapping di program utama dengan menambahkan kode untuk memanggil file header, inisialisasi variabel dan penambahan fungsi sebagai berikut:
   a. Header
   ```
   #include "imageloader.h"
   ```
   b. Variabel
   ```
   GLuint _textureId; //The id of the texture
   ```
   c. Fungsi **loadTexture**

```
GLuint loadTexture(Image* image) {
    GLuint textureId;
    glGenTextures(1, &textureId); //Make room for our texture
    glBindTexture(GL_TEXTURE_2D, textureId); //Tell OpenGL which texture to edit
    //Map the image to the texture
    glTexImage2D(GL_TEXTURE_2D,                //Always GL_TEXTURE_2D
                 0,                            //0 for now
                 GL_RGB,                       //Format OpenGL uses for image
                 image->width, image->height,  //Width and height
                 0,                            //The border of the image
                 GL_RGB, //GL_RGB, because pixels are stored in RGB format
                 GL_UNSIGNED_BYTE, //GL_UNSIGNED_BYTE, because pixels are stored
                                   //as unsigned numbers
                 image->pixels);               //The actual pixel data
    return textureId; //Returns the id of the texture
}
```

d. Source code yang dibutuhkan untuk image loader

Header (imageloader.h)

```
#ifndef IMAGE_LOADER_H_INCLUDED
#define IMAGE_LOADER_H_INCLUDED
//Represents an image
class Image {
      public:
            Image(char* ps, int w, int h);
            ~Image();
            /* An array of the form (R1, G1, B1, R2, G2, B2, ...)
indicating the
             * color of each pixel in image.  Color components range
from 0 to 255.
             * The array starts the bottom-left pixel, then moves
right to the end
             * of the row, then moves up to the next column, and so
on.  This is the
             * format in which OpenGL likes images.
             */
            char* pixels;
            int width;
            int height;
};
//Reads a bitmap image from file.
Image* loadBMP(const char* filename);

#endif
```

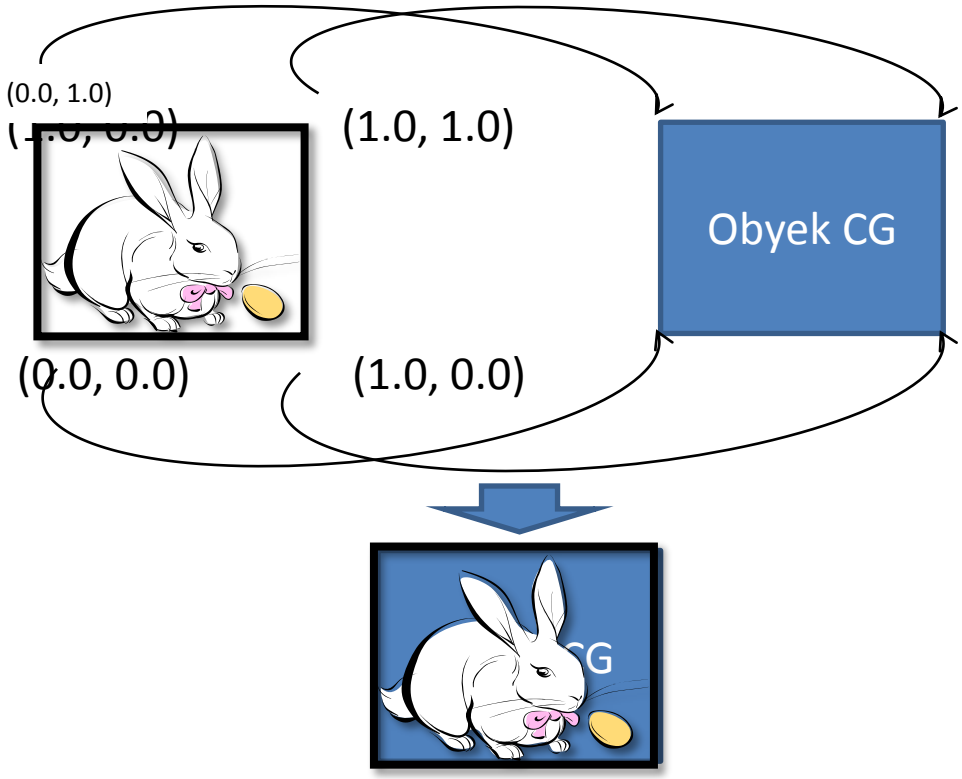
Source (imageloader.cpp)

```
#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 {
	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]);
	}

	short toShort(const char* bytes) {
		return (short)(((unsigned char)bytes[1] << 8) |
						(unsigned char)bytes[0]);
	}

	int readInt(ifstream &input) {
		char buffer[4];
		input.read(buffer, 4);
		return toInt(buffer);
	}

	short readShort(ifstream &input) {
		char buffer[2];
		input.read(buffer, 2);
		return toShort(buffer);
	}

	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;
			}
```

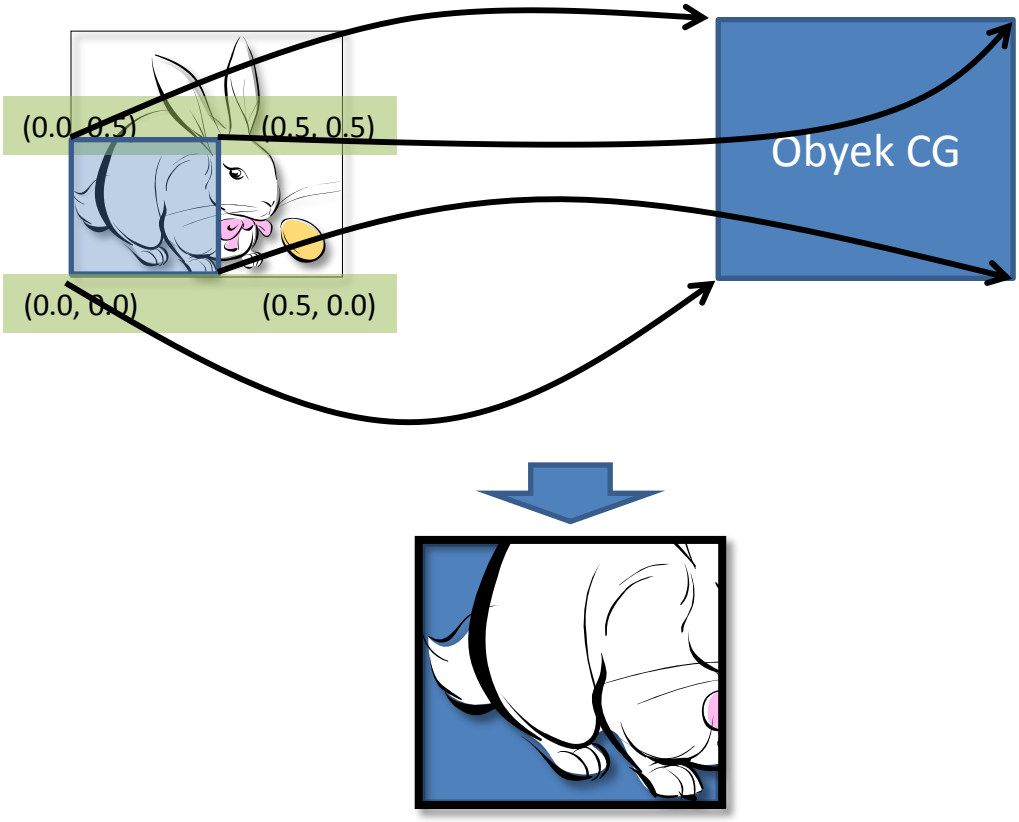
```
                        void operator=(const auto_array<T> &aarray) {
                                if (!isReleased && array != NULL) {
                                        delete[] array;
                                }
                                array = aarray.array;
                                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 = readInt(input);
                        height = readInt(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);
}
```

5. Proses texture maping



6. Proses texture mapping 2 (menampilkan sebagian dari gambar texture ke objek openGL)



E. AKTIFITAS KELAS PRAKTIKUM

1. Buatlah project dan beri nama texturing.
2. Buka kembali kode program pada praktikum lighting dan material yang telah berjalan, kemudian copykan kode program tersebut pada project texturing.
3. Letakkan file header imageloader.h ke dalam folder project texuring yang telah dibuat, kemudian tambahkan kode untuk memanggil file header

```
#include "imageloader.h"
```

4. Letakkan file texture gambar “earth.bmp” ke dalam folder profect texturing yang telah dibuat.
5. Inisialisasi variabel global textureId dan letakkan dibawah kode program include

```
#include "imageloader.h"
GLuint _textureId; //variabel textureId
```

6. Buka file imageloader.cpp kemudian tambahkan semua kode pada imageloader.cpp ke dalam project texturing (kode untuk image texturing diletakkan dibagian atas).

7. Modifikasi fungsi init pada program, seperti kode porgram berikut ini. (perhatian: ubah nama file bmp (“flower.bmp”) pada contoh kode di bawah, dengan nama file gambar texture yang dibuat)

```
void init(){
    GLfloat LightPosition[] =   {-10.0f, 20.0f, 20.0f, 0.0f};
    glShadeModel(GL_SMOOTH);
    glClearColor(0.0f,0.0f,0.0f,0.5f);
    glClearDepth(1.0f);
    glEnable(GL_DEPTH_TEST); /* ... */
    glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);

    glLightfv(GL_LIGHT0, GL_POSITION, LightPosition);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_COLOR_MATERIAL);
    glBlendFunc(GL_SRC_ALPHA,GL_ONE);
    glColorMaterial(GL_FRONT, GL_DIFFUSE);

    Image* image = loadBMP("flower.bmp");
    _textureId = loadTexture(image);
    return;
}
```

8. Buatlah fungsi **mydisplay** seperti kode program di bawah ini. Tampilkan screenshot dan berilah kesimpulan.

```
void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, _textureId);  ///menghubungkan texture dg variabel textur
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTranslatef(0,0,z_pos);
    glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 0.0f);
            glVertex2f(-2.0f, -2.0f);
        glTexCoord2f(1.0f, 0.0f);
            glVertex2f(2.0f, -2.0f);
        glTexCoord2f(1.0f, 1.0f);
            glVertex2f(2.0f, 2.0f);
        glTexCoord2f(0.0f, 1.0f);
            glVertex2f(-2.0f, 2.0f);
    glEnd();
    glDisable(GL_TEXTURE);
    glutSwapBuffers();
}
```

9. Untuk ‘menempelkan’ sebuah texture yang telah dirotasi ke obyek CG adalah dengan mengubah pasangan **glTexCoord2f** dan fungsi **glVertex2f** dengan urutan berbeda. Ubahlah fungsi display seperti di bawah ini, Tampilkan hasil renderingnya dan beri kesimpulan.

```
void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, _textureId);  ///menghubungkan texture
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTranslatef(0,0,z_pos);
    glBegin(GL_QUADS);
        glTexCoord2f(1.0f, 0.0f);
            glVertex2f(-2.0f, -2.0f);
        glTexCoord2f(1.0f, 1.0f);
            glVertex2f(2.0f, -2.0f);
        glTexCoord2f(0.0f, 1.0f);
            glVertex2f(2.0f, 2.0f);
        glTexCoord2f(0.0f, 0.0f);
            glVertex2f(-2.0f, 2.0f);
    glEnd();
    glDisable(GL_TEXTURE);
    glutSwapBuffers();
}
```

10. Ubahlah kembali fungsi **mydisplay** seperti di bawah ini. Program di bawah adalah mengambil sebagian kecil image untuk dipasangkan pada obyek CG.

```
void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, _textureId);  ///menghubungkan textur
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTranslatef(0,0,z_pos);
    glBegin(GL_QUADS);
        glTexCoord2f(0.4f, 0.6f);
            glVertex2f(-2.0f, -2.0f);
        glTexCoord2f(0.5f, 0.6f);
            glVertex2f(2.0f, -2.0f);
        glTexCoord2f(0.5f, 0.7f);
            glVertex2f(2.0f, 2.0f);
        glTexCoord2f(0.4f, 0.7f);
            glVertex2f(-2.0f, 2.0f);
    glEnd();
    glDisable(GL_TEXTURE);
    glutSwapBuffers();
}
```

F. TUGAS ASISTENSI

1. Buatlah program yang menampilkan sebuah kubus dengan texture tiap sisinya menampilkan texture nama tiap anggota. Image texture mengandung semua tampilan nama anggota kelompok. Tampilkan source code, texture, dan tampilan kubus yang nampak semua sisinya (3 tampilan @ 2 sisi).