Computer Graphics

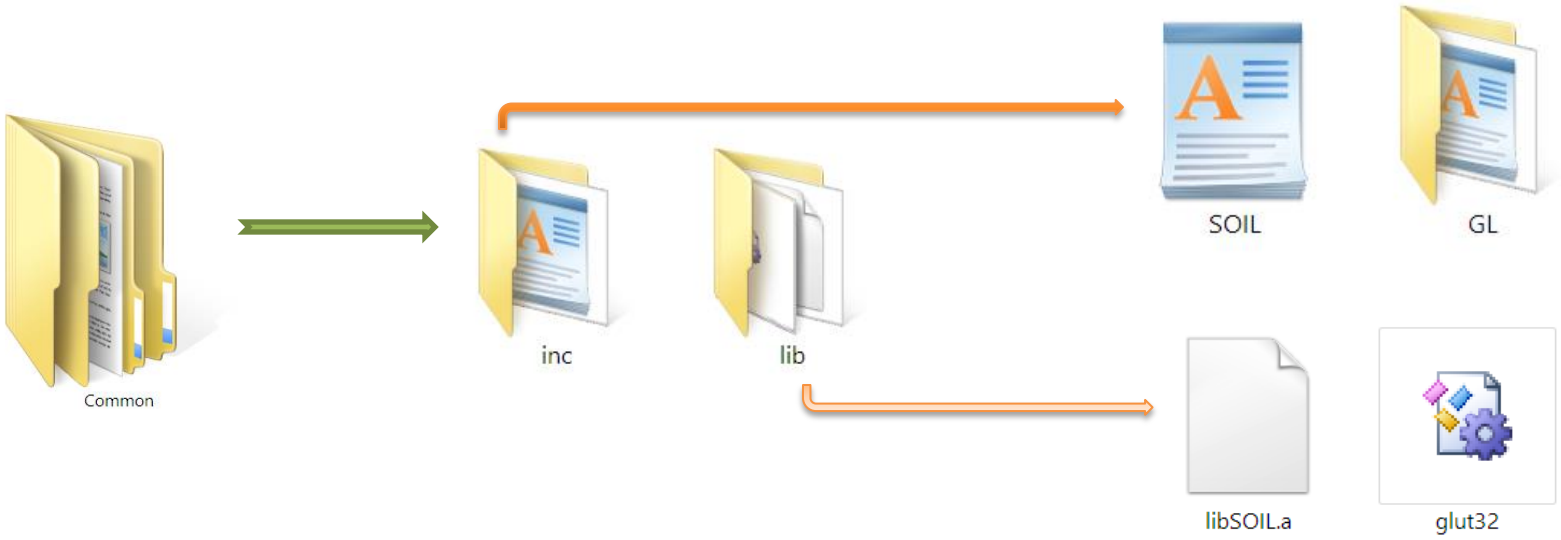
Lecture 19

Texture mapping with OpenGL

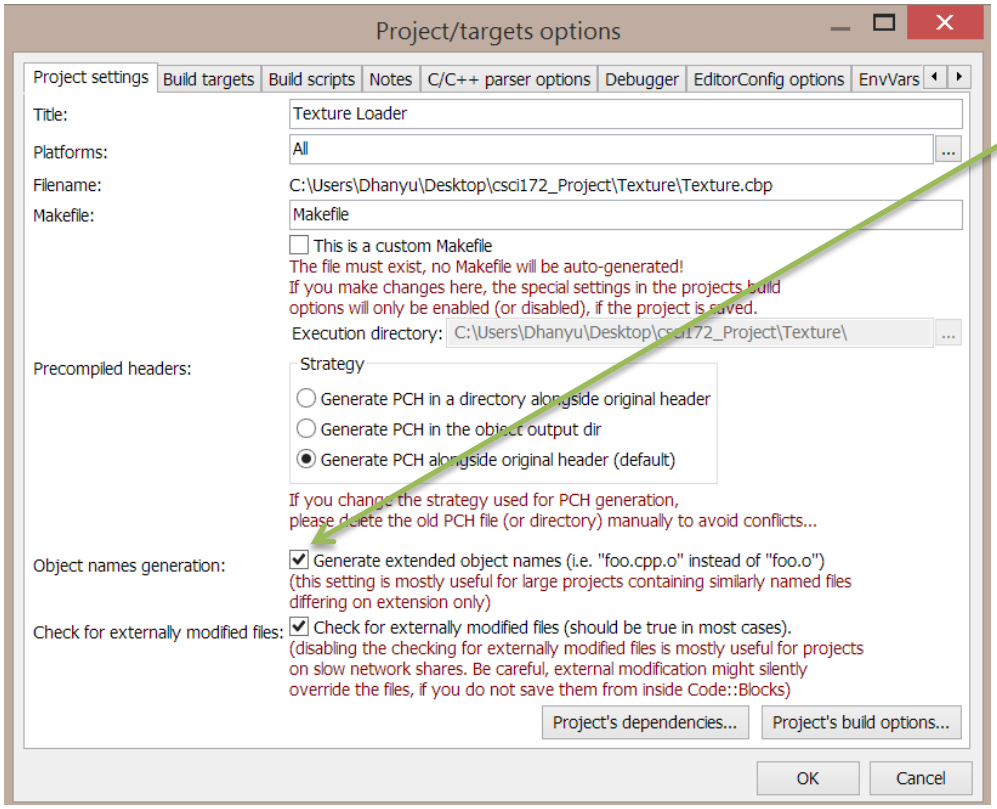
Set Up:

Download SOIL (Simple OpenGL Image Library)

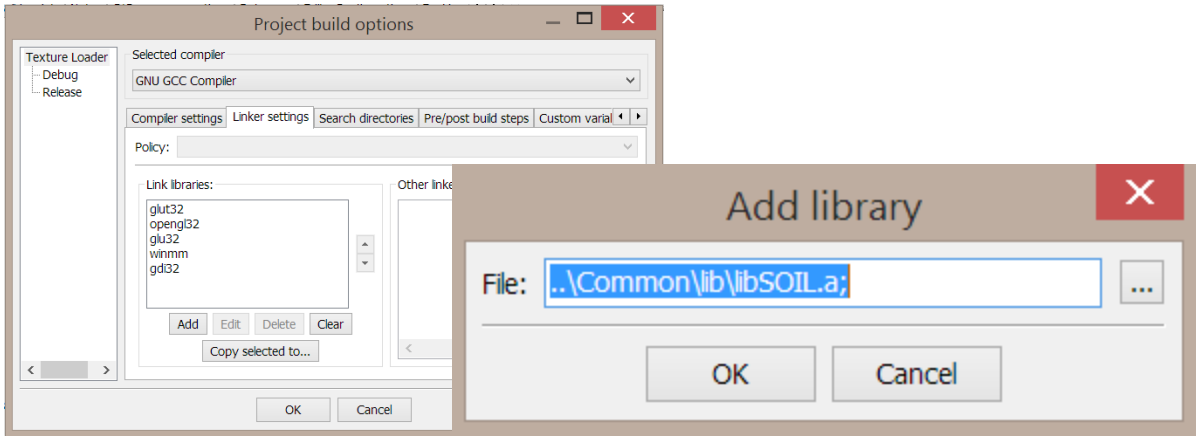
Copy **SOIL.H** and **LibSOIL.a** into appropriate folders

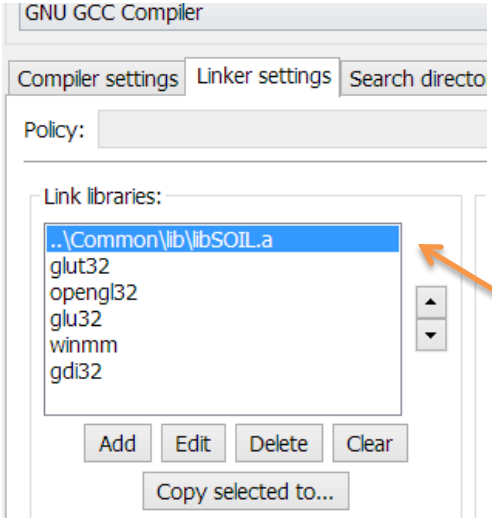


Setting up CodeBlocks



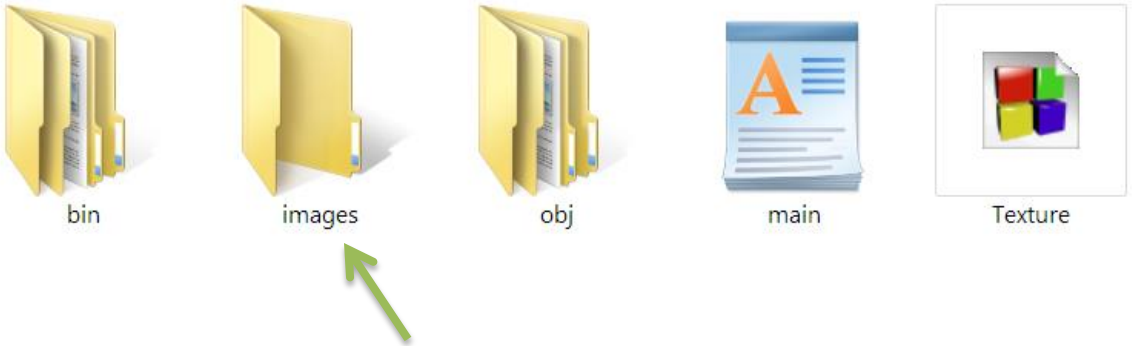
Extend for object names





Move up in the list

Image folder



Create images folder under the project folder
Place an image (*.bmp, *.png, *.jpg, *.tga)

Textures objects and parameters

```
#include <SOIL.h>
#include <math.h>
using namespace std;

bool WireFrame= false;

GLuint tex;                          // Texture pointer
GLdouble TranslateX,TranslateY,Zoom,RotateX,RotateY,RotateZ;
```

Add Texture to Teapot

Set gluLookAt(0,0,10,0.0,0.0,0.0,0.0,1.0,0.0);

void glutSolidTeapot (1.5);

Keyboard Settings

```
void Specialkeys(int key, int x, int y)
{
   switch(key)
   {
                    case GLUT_KEY_END:
                      Zoom += (float) 1.0f;                          // Zoom in
                                break;
                    case GLUT_KEY_HOME:
                      Zoom -= (float) 1.0f;                          // Zoom Out
                                break;
                     case GLUT_KEY_UP:                               //Clockwise rotation over X
                                RotateX = ((int)RotateX +2)%360;
                                break;
                    case GLUT_KEY_DOWN:                             //Counter Clockwise rotation over X
                                RotateX = ((int)RotateX -2)%360;
                                 break;
                    case GLUT_KEY_LEFT:                             // Clockwise rotation over Y
                    RotateY =((int)RotateY +2)%360;
                                break;
                    case GLUT_KEY_RIGHT:
                                RotateY = ((int)RotateY -2)%360;    //Counter Clockwise rotation over Y
                                 break;
  }

glutPostRedisplay();

}
```

Texture Setup

```
static void init(void)
{
   glEnable(GL_TEXTURE_2D);
   glGenTextures(1, &tex);

   glTexEnvi(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_MODULATE);
   glBindTexture(GL_TEXTURE_2D, tex);  // images are 2D arrays of pixels, bound to the GL_TEXTURE_2D target.

   int width, height;                  // width & height for the image reader
   unsigned char* image;               //Place your file name

   image = SOIL_load_image("images/back.jpg", &width, &height, 0, SOIL_LOAD_RGB);

   glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, image);
                                       // binding image data
  SOIL_free_image_data(image);

   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

```
glPushMatrix();
   glTranslated(TranslateX,TranslateY,Zoom);
   glRotated(RotateX,1,0,0);
   glRotated(RotateY,0,1,0);
   glRotated(RotateZ,0,0,1);

   glBegin(GL_QUADS);
            glNormal3f(0.0f, 0.0f, 1.0f);              // Normal Front Face
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);  // Bottom Left Of The Texture and Quad
            glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);  // Bottom Right Of The Texture and Quad
            glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f,  1.0f);  // Top Right Of The Texture and Quad
            glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f,  1.0f);  // Top Left Of The Texture and Quad
            glNormal3f(0.0f, 0.0f, -1.0f);             // Normal Back Face
            glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);  // Bottom Right Of The Texture and Quad
            glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);  // Top Right Of The Texture and Quad
            glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);  // Top Left Of The Texture and Quad
            glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f, -1.0f);  // Bottom Left Of The Texture and Quad
            glNormal3f(0.0f, 1.0f, 0.0f);              // Normals Top Face
            glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);  // Top Left Of The Texture and Quad
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f,  1.0f,  1.0f);  // Bottom Left Of The Texture and Quad
            glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f,  1.0f,  1.0f);  // Bottom Right Of The Texture and Quad
            glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);  // Top Right Of The Texture and Quad
            glNormal3f(0.0f, -1.0f, 0.0f);             // Bottom Face
            glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);  // Top Right Of The Texture and Quad
            glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f, -1.0f, -1.0f);  // Top Left Of The Texture and Quad
            glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);  // Bottom Left Of The Texture and Quad
            glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);  // Bottom Right Of The Texture and Quad
            glVertex3f(1.0f, 0.0f, 0.0f);              // Right face
            glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f, -1.0f);  // Bottom Right Of The Texture and Quad
            glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f,  1.0f, -1.0f);  // Top Right Of The Texture and Quad
            glTexCoord2f(0.0f, 1.0f); glVertex3f( 1.0f,  1.0f,  1.0f);  // Top Left Of The Texture and Quad
            glTexCoord2f(0.0f, 0.0f); glVertex3f( 1.0f, -1.0f,  1.0f);  // Bottom Left Of The Texture and Quad
            glNormal3f(-1.0f, 0.0f, 0.0f);             // Left Face
            glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);  // Bottom Left Of The Texture and Quad
            glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);  // Bottom Right Of The Texture and Quad
            glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f,  1.0f,  1.0f);  // Top Right Of The Texture and Quad
            glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);  // Top Left Of The Texture and Quad
   glEnd();
   glPopMatrix();
```

GL_TEXTURE_2D:

Image bind to the “Texture space”.

These coordinates range from 0.0 to 1.0 where (0,0) is conventionally the bottom-left corner and (1,1) is the top-right corner of the texture image. The operation that uses these texture coordinates to retrieve color information from the pixels is called *sampling*.

Wrapping:

Determine how the texture should be sampled when a coordinate outside the range of 0 to 1 is given. OpenGL offers 4 ways of handling this:

GL_REPEAT:	The integer part of the coordinate will be ignored and a repeating pattern is formed
GL_MIRRORED_REPEAT:	The texture will also be repeated, but it will be mirrored when the integer part of the coordinate is odd.
GL_CLAMP_TO_EDGE:	The coordinate will simply be clamped between 0 and 1.
GL_CLAMP_TO_BORDER:	The coordinates that fall outside the range will be given a specified border color

https://open.gl/textures

equivalent of (x,y,z) in texture coordinates is called (s,t,r). Texture parameter are changed with the glTexParameter* functions as demonstrated here.

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```

Note:
GL_CLAMP_TO_BORDER associate with GL_TEXTURE_BORDER_COLOR as well

Filtering:

- Texture coordinates are resolution independent. they won't always match a pixel exactly.
- This happens when a texture image is stretched beyond its original size or when it's sized down.
- OpenGL offers various methods to decide on the sampled color when this happens.

This process is called filtering and the following methods are available:

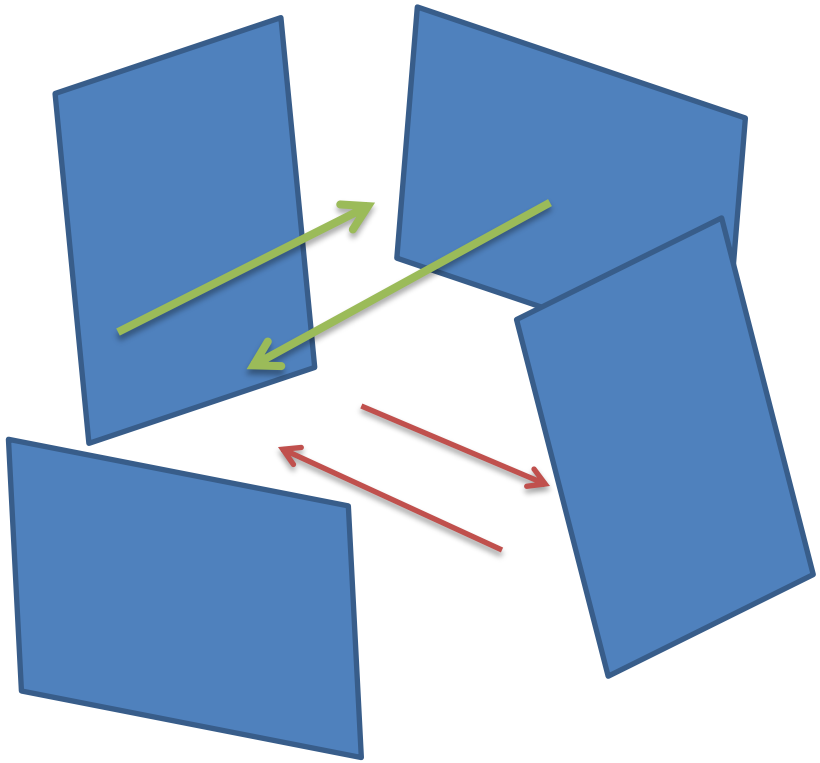
- GL_NEAREST: Returns the pixel that is closest to the coordinates.
- GL_LINEAR: Returns the weighted average of the 4 pixels surrounding the given coordinates.

MipMap Sampling

GL_NEAREST_MIPMAP_NEAREST,
GL_LINEAR_MIPMAP_NEAREST,
GL_NEAREST_MIPMAP_LINEAR,
GL_LINEAR_MIPMAP_LINEAR


```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

Making Inverse Box



Change appropriate vertex value

glVertex3f(-1.0f, -1.0f, 1.0f);