Fall 2015

CSCI 420: **Computer Graphics**

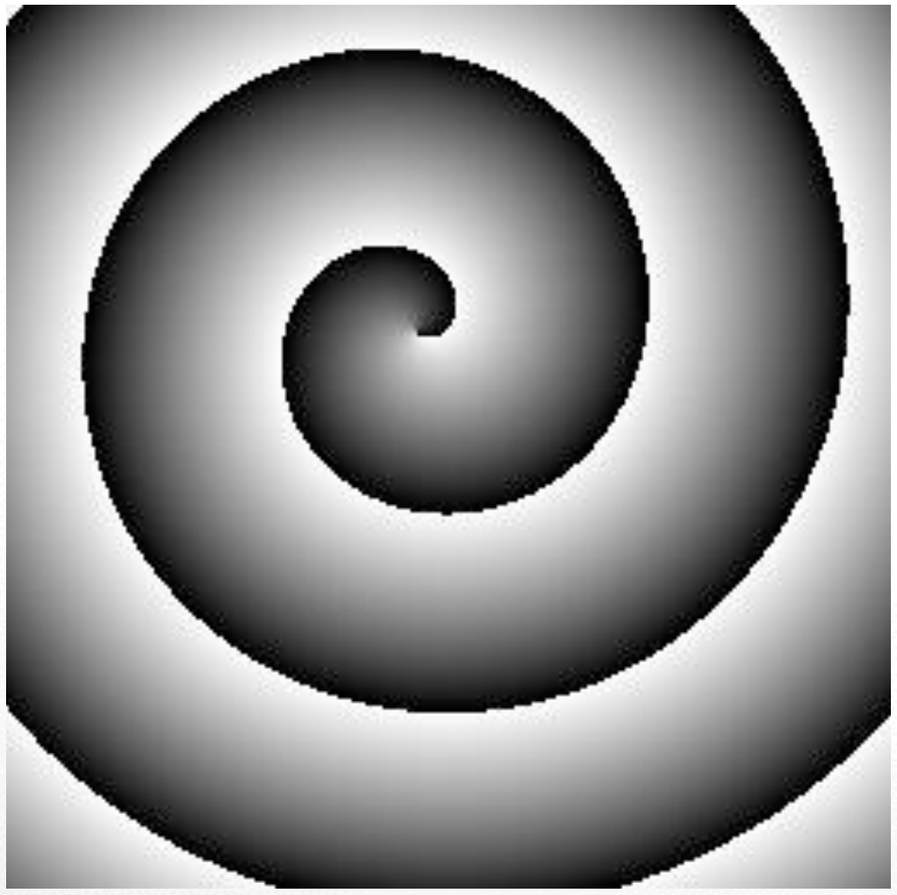
Homework #1 Introduction and Tips

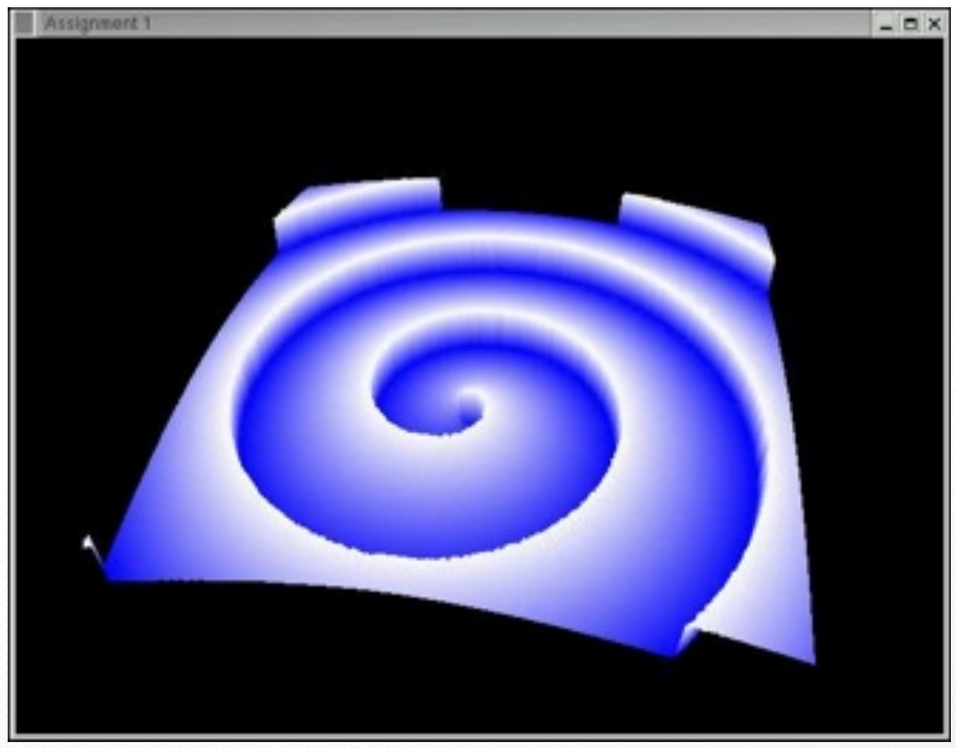
Kyle Olszewski

http://cs420.hao-li.com

Goal

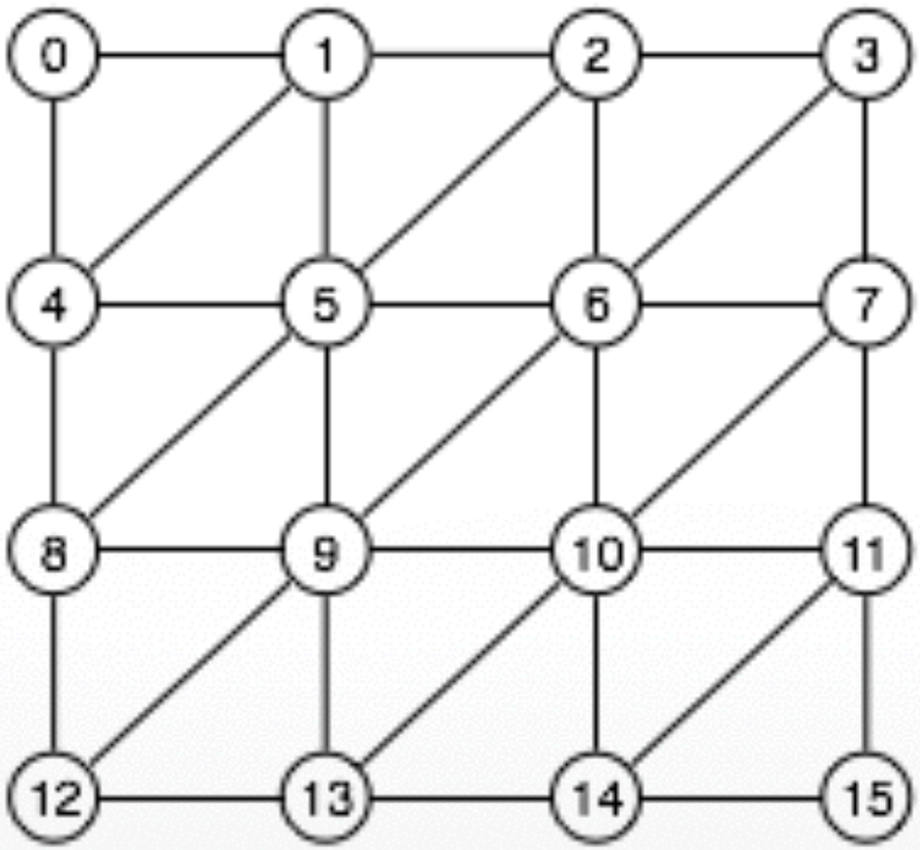
- Create a height field based on an input image
- Height field is a function f(x, y) = z
  - Intensity of pixel (x, y) defines height z at that point





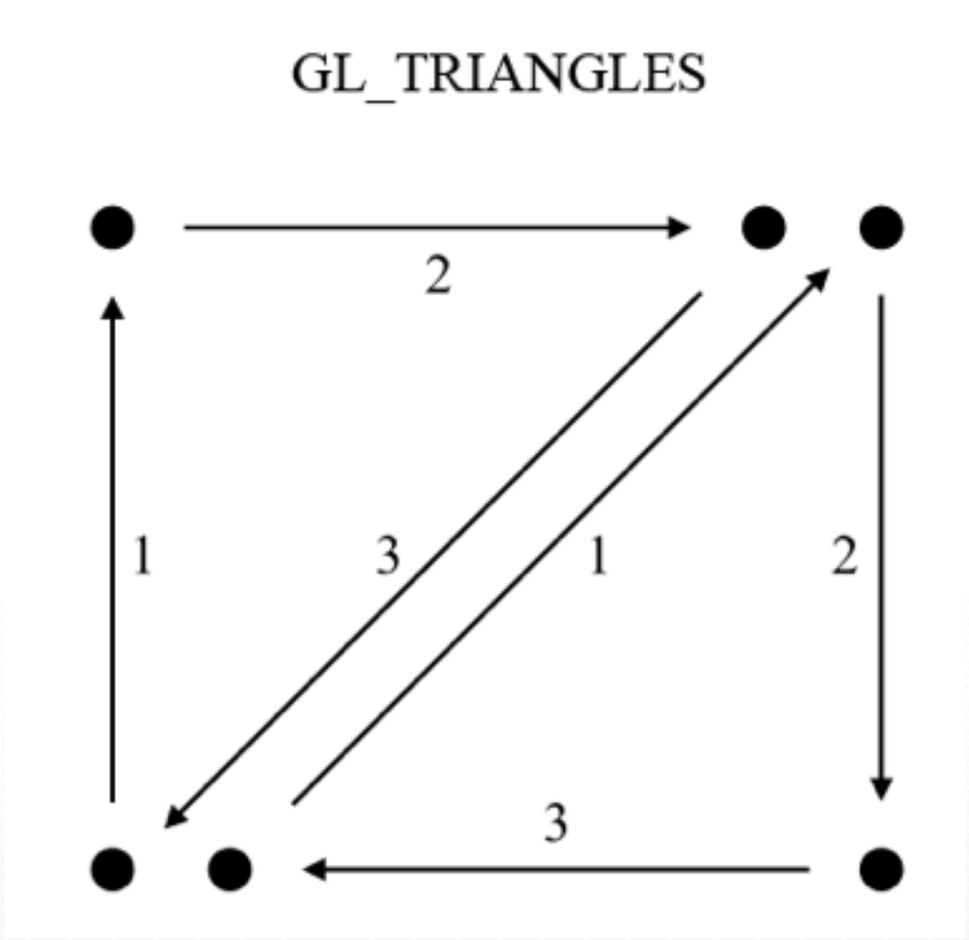
Approach

- Height defined at discrete points (each pixel)
- Represent the 3D positions of these points as vertices
- Render these vertices with OpenGL using triangles



Simple Approach

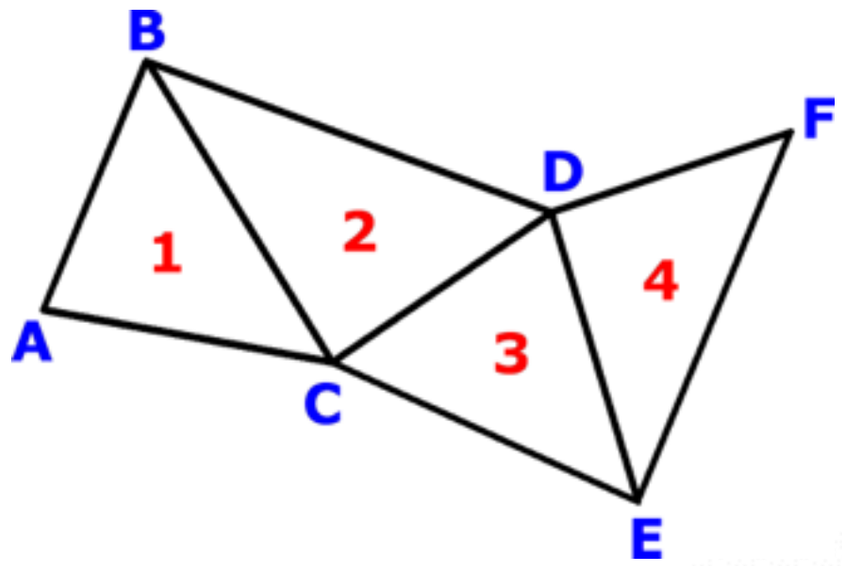
- Use GL_TRIANGLES to specify vertices for each triangle separately
- Must specify 3n vertices for n triangles



Using GL_TRIANGLES

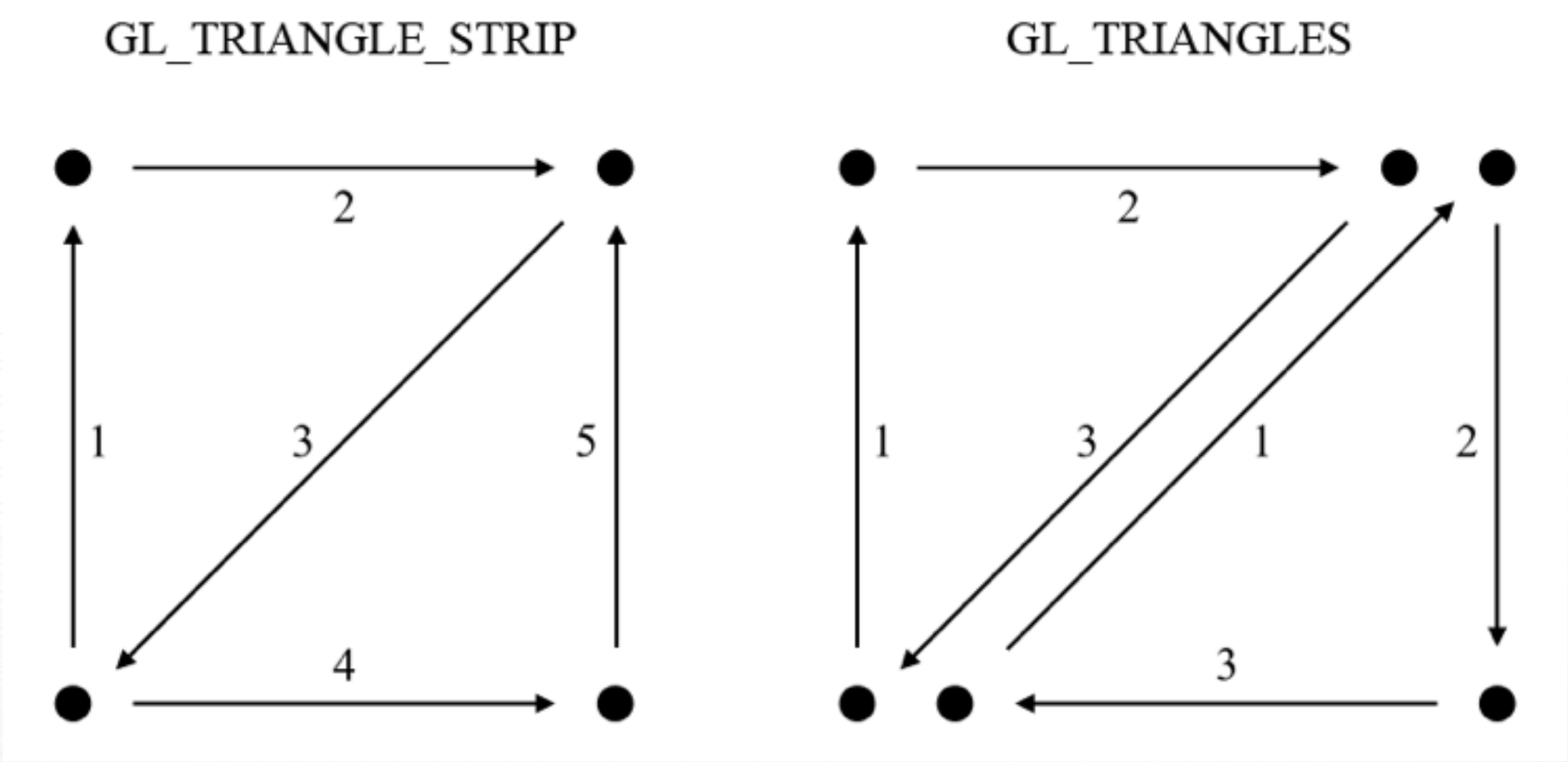
```
glBegin(GL_TRIANGLES); // specify type of shapes to draw
    // draw first triangle
    glColor3f(A_Red, A_Green, A_Blue); // first vert color
    glVertex3f(A_X, A_Y, A_Z); // first vert pos
    glColor3f(B_Red, B_Green, B_Blue);
    glVertex3f(B_X, B_Y, B_Z);
    glColor3f(C_Red, C_Green, C_Blue);
    glVertex3f(C_X, C_Y, C_Z);

    // draw second triangle
    glColor3f(C_Red, C_Green, C_Blue);
    glVertex3f(C_X, C_Y, C_Z);
    glColor3f(B_Red, B_Green, B_Blue);
    glVertex3f(B_X, B_Y, B_Z);
    glColor3f(D_Red, D_Green, D_Blue);
    glVertex3f(D_X, D_Y, D_Z);
      …
glEnd(); // finish drawing
```



Optimized Approach

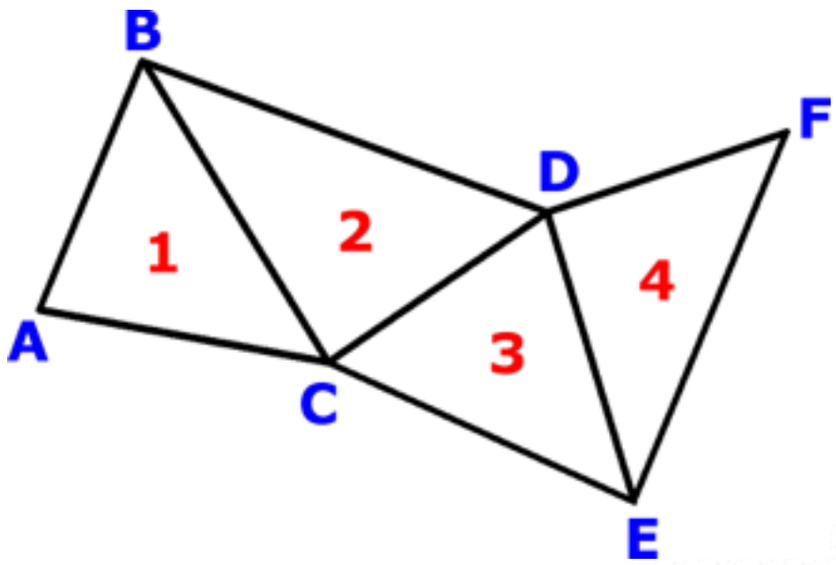
- Use GL_TRIANGLE_STRIP to draw adjacent triangles
- Must specify 2 + n vertices for n triangles



Using GL_TRIANGLE_STRIP

```
glBegin(GL_TRIANGLE_STRIP); // specify type of shapes to draw
    // specify initial 2 vertices
    glColor3f(A_Red, A_Green, A_Blue); // first vert color
    glVertex3f(A_X, A_Y, A_Z); // first vert pos
    glColor3f(B_Red, B_Green, B_Blue);
    glVertex3f(B_X, B_Y, B_Z);

     // specify last vertex for first triangle
    glColor3f(C_Red, C_Green, C_Blue);
    glVertex3f(C_X, C_Y, C_Z);
    // specify new vertex for second triangle
    glColor3f(D_Red, D_Green, D_Blue);
    glVertex3f(D_X, D_Y, D_Z);
   // specify new vertex for third triangle
    glColor3f(E_Red, E_Green, E_Blue);
    glVertex3f(E_X, E_Y, E_Z);
      …
glEnd(); // finish drawing
```



Lines of Triangle Strips

- We can turn pairs of pixel rows ("scanlines") into tri-strips.

```
for(int i=0;i<pic->ny-1;i++) {
  OGL_initialize tri-strip creation
  for(int j=0;j<pic->nx;j++) {
    indx0 = (j, i, z from PIC_PIXEL()) // 'top' vertex
    indx1 = (j, i+1, z from PIC_PIXEL()) // 'bottom' vertex
    // sequential top,bottom vert pairs generates a tri-strip
    OGL_specify() vertex with z=indx0
    OGL_specify() vertex with z=indx1
  }// next pixel in current row
  OGL_end current tri-strip
}// next row
```

Rendering Modes

- Use glPolygonMode() to control render mode (vertices, wireframe, or solid triangles)

```
// render vertices
glPolygonMode( GL_FRONT_AND_BACK, GL_POINT );

// render wireframe
glPolygonMode( GL_FRONT_AND_BACK, GL_LINE );

// render solid triangles
glPolygonMode( GL_FRONT_AND_BACK, GL_FILL );
```

Other Tips

- Use glEnable(GL_DEPTH_TEST) during initialization to make use of depth buffer
  - Ensures that objects farther from camera will be occluded by nearer objects
- Clear color and depth buffer before rendering each frame:

  glClear(GL_COLOR_BUFFER_BIT |GL_DEPTH_BUFFER_BIT);
- Tips on optimized triangle strip rendering:

  http://dan.lecocq.us/wordpress/2009/12/25/triangle-strip-for-grids-a-construction/

Common Questions

- How are height values represented?
- How is the data stored in ‘pix’?
- What does the PIC_PIXEL macro do?
- I need help regarding C errors.

Height Values

- Heights are specified as grayscale, 8 bits/channel.
- Each height value is simply an 'unsigned char' (0 - 255).
- The pixel values are held in the 'pix' array in the 'Pic' data structure.

```
typedef struct {
    int nx, ny;
    int bpp;
    Pixel1 *pix; /* array of pixels*/
}
```

Pixel Values

- Consider the following 4 X 4 (16 Pixel) Image:

100 110 120 130

200 215 230 245

250 200 150 100

0 30 60 90

- The data would be laid out in ‘pix’ in “row major” order:

100 110 120 130 200 215 230 245 250 200 150 100 0 30 60 90

Accessing Pixel Values

- To access a pixel value at any (x,y) [eg. at (2,3)], use the 'PIC_PIXEL' macro definition (defined in pic.h, with function signature PIC_PIXEL(pic, x, y, chan)), with chan=0.

```
for(int i=0;i<pic->ny;i++) {
  for(int j=0;j<pic->nx;j++) {
    // chan=0, since we're accessing the first/only channel
    unsigned char heightVal = PIC_PIXEL(pix,j,i,0);
    // use heightVal..
  }// next pixel in current row
}// next row
```

Creating Filenames

- There's a good way to create filenames with 4-digit-padding

```
char myFilenm[2048];

for (int i=0;i<1000;i++) {
 sprintf(myFilenm, "anim.%04d.jpg", i);
 // myFilenm will be anim.0001.jpg, anim.0002.jpg.....anim.0999.jpg
 // ..
}}
```

C Errors

- Here is a guide for catching C errors. (Particularly helpful for those students whose ‘first language’ is not C/C++)

http://www.drpaulcarter.com/cs/common-c-errors.php

http://cs420.hao-li.com

Thanks!