CSCI 420 Computer Graphics

Helper slides, hw1 (height field)

Jernej Barbic
University of Southern California

Important first steps

- There must be glutSwapBuffers() at the end of displayFunc()

- There must be glutPostRedisplay() at the end of idleFunc()

Understanding modelview and projection matrices

- 4x4 matrices
- You compute them using the OpenGLMatrix class
- You send them to the shader using glUniformMatrix4fv

- There are two OpenGLMatrix modes: ModelView and Projection
- Use OpenGLMatrix::SetMatrixMode to set the mode

Computing the projection matrix

- Compute in reshape()
- Change the mode to Projection
- Clear the matrix to identity (OpenGLMatrix::LoadIdentity)
- Then, call OpenGLMatrix::Perspective
- Good habit to then set the mode back to ModelView

Uploading the projection matrix to GPU

Inside displayFunc():

```
float p[16];
openGLMatrix->SetMatrixMode(OpenGLMatrix::Projection);
openGLMatrix->GetMatrix(p);
```

- Then, upload the array p to the GPU:
  See the "Setting up uniform variables" slides
  in the "Shaders" lecture.

Computing the modelview matrix

- Compute in displayFunc()
- Change the mode to ModelView
  openGLMatrix->SetMatrixMode(OpenGLMatrix::ModelView);
- Clear the matrix to identity (OpenGLMatrix::LoadIdentity)
- Then, call OpenGLMatrix::LookAt()
- Then, call OpenGLMatrix::Translate,Rotate,Scale
- Then

```
float m[16];
openGLMatrix->GetMatrix(m);
```

- Then, upload the array m to the GPU

Initialization

- Init and bind the pipeline program:
  pipelineProgram->BuildShadersFromFiles(“../openGLHelper”, "vertexShader.glsl", "fragmentShader.glsl”);
- pipelineProgram->Bind();

- Generate the VBO and VAO, and properly upload them to the GPU

  See the "Vertex Array Object" slides ("Shaders" lecture), and the "Vertex Buffer Object" slides (”Colors and Hidden Surface Removal" lecture).

Write the vertex and fragment shaders

- See “Shaders”:
  “Basic Vertex Shader in GLSL” and
  “Basic Fragment Shader”

Note: basic shaders are already written in the starter code.

Heightfield VBOs and VAOs

- 2 VBOs + 1 VAO for solid mode
  2 VBOs + 1 VAO for wireframe mode
  2 VBOs + 1 VAO for point mode
  One VBO for positions and one for colors

- 6 VBOs + 1 VAO for “smoothing mode”
  Positions: center, left, right, down, up; and Color

Gotchas to avoid

- First, initialize OpenGL.
- VAO must be initialized AFTER the pipeline program has been initialized and bound.
- VAO must be initialized AFTER setting up VBO.
- The order of setting up the VBO and the pipeline program does not matter.

- Data sent to VBO must be contiguous.

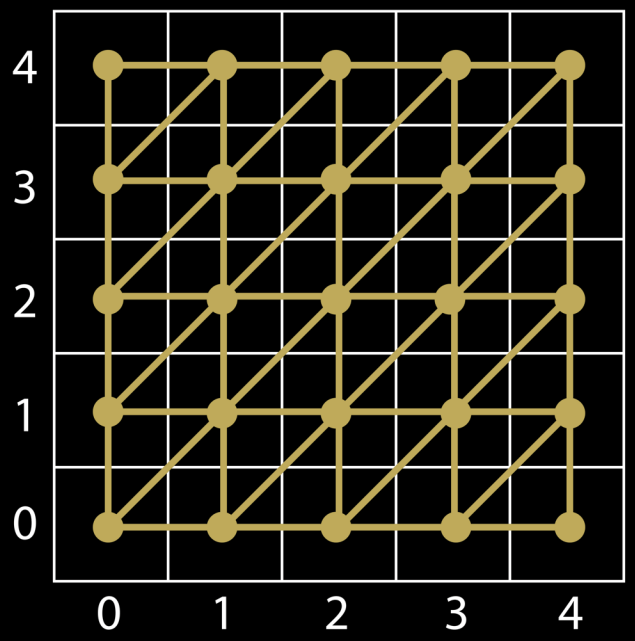
```
float* vertices[36];
vertices[0] = new float[3];
vertices[1] = new float[3];
…
```

Rendering (in displayFunc)

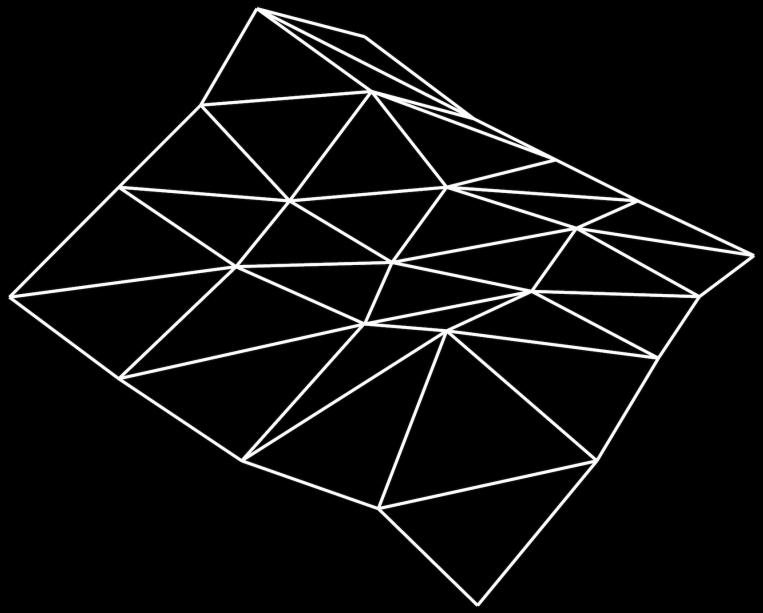
- Setup modelview and projection matrices
  (as shown in the previous slides in this presentation)
- Bind the VAO
- Render using glDrawArrays()
- See "Use the VAO" slide in "Shaders"

Rendering the height field

Understanding the Height Field



The image (5x5)



The heightfield

Pixel (i,j)

3D vertex
x = i / (resolution – 1)
y = height
z = -j / (resolution – 1)

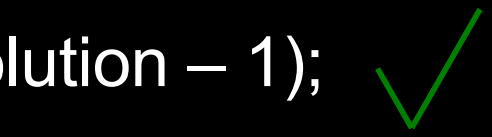
height = heightmapImage->getPixel(i, j, 0) / 255.0f

Warning

```
int i = …
int resolution = …
float x = i /  (resolution – 1);
```

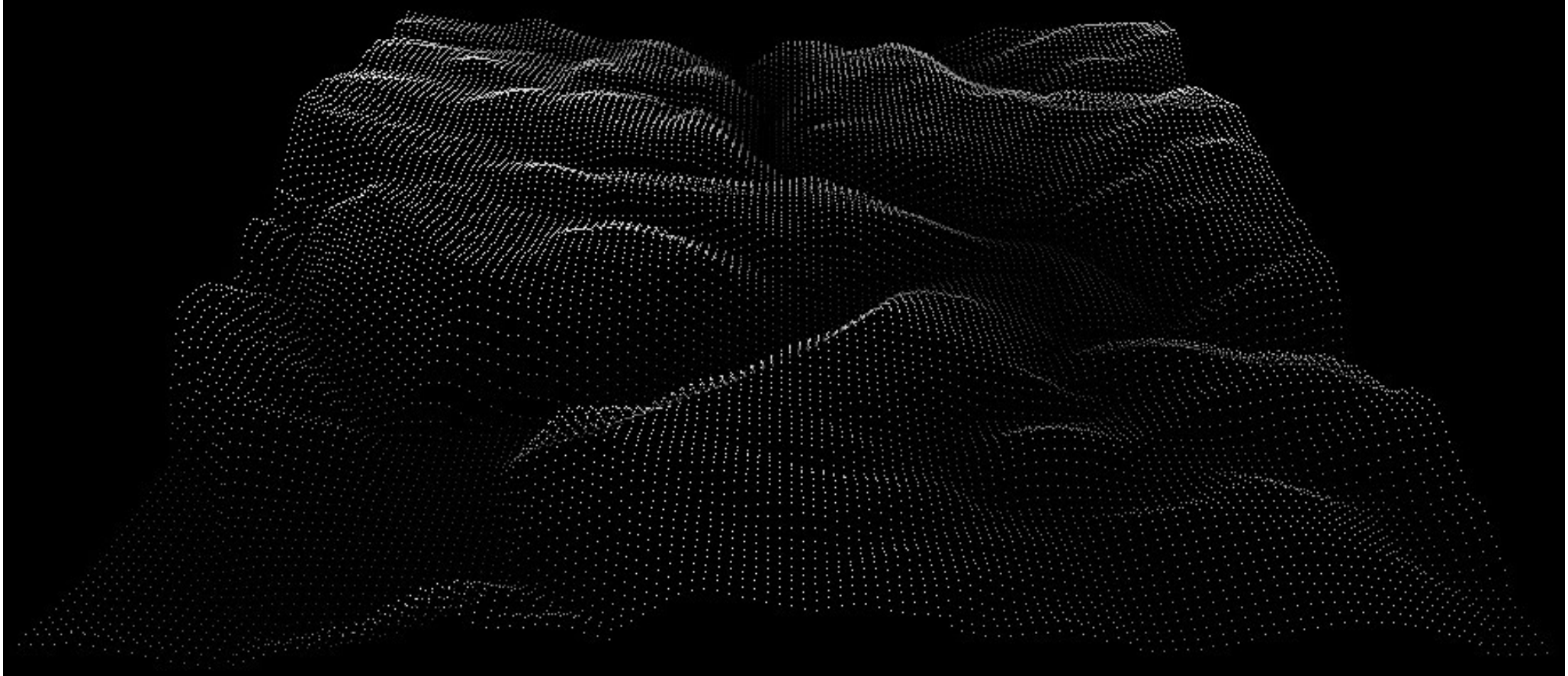
Malfunction due
to integer division!

```
float x = 1.0 * i /  (resolution – 1);
```

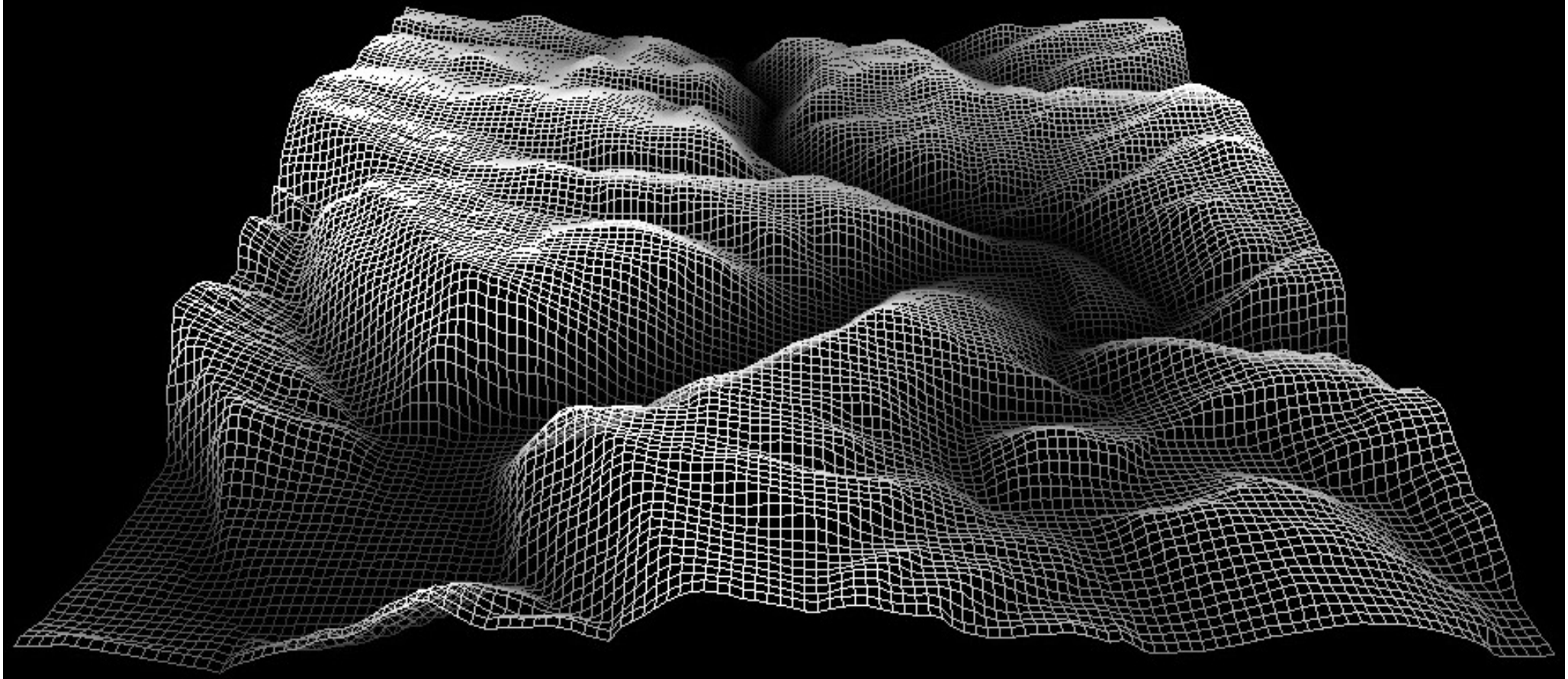


In other words, be careful about not accidentally triggering integer division in C !

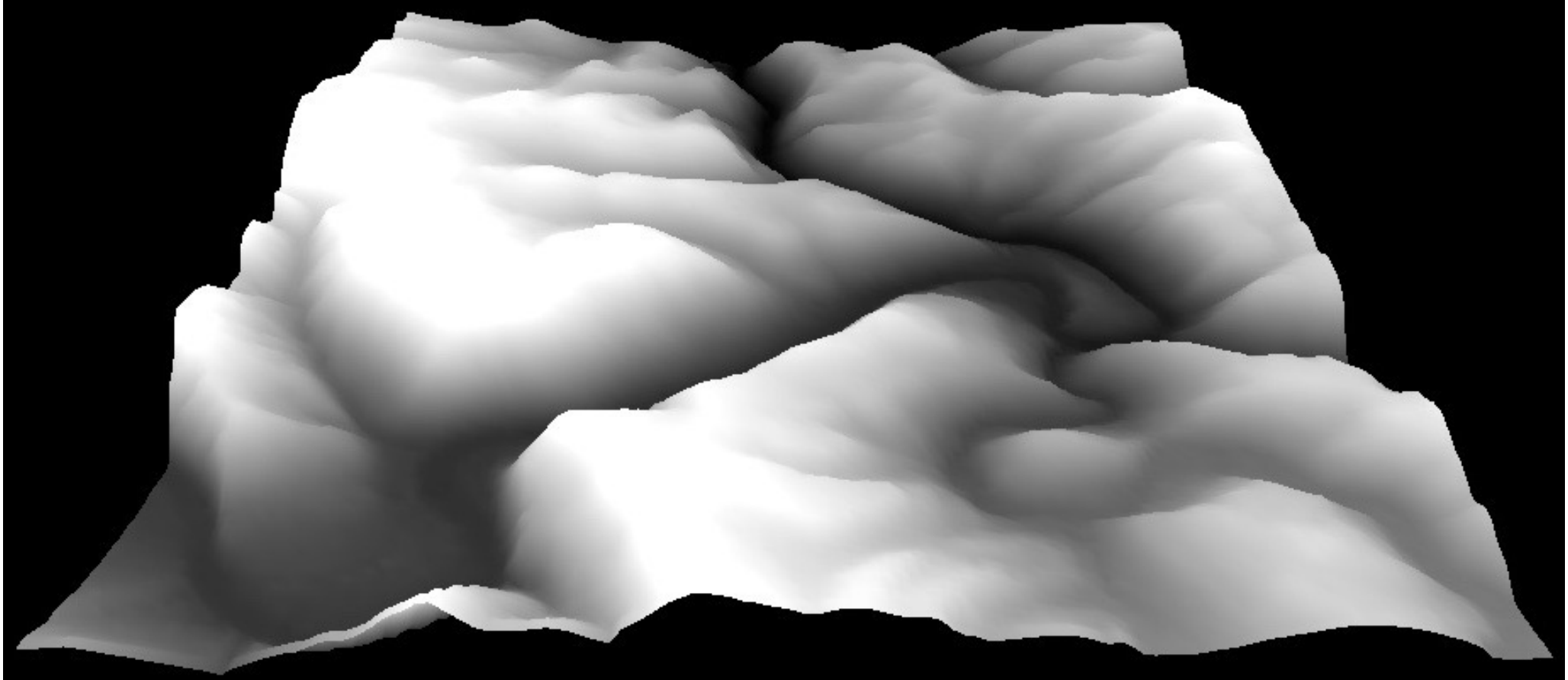
Point mode (“1”)



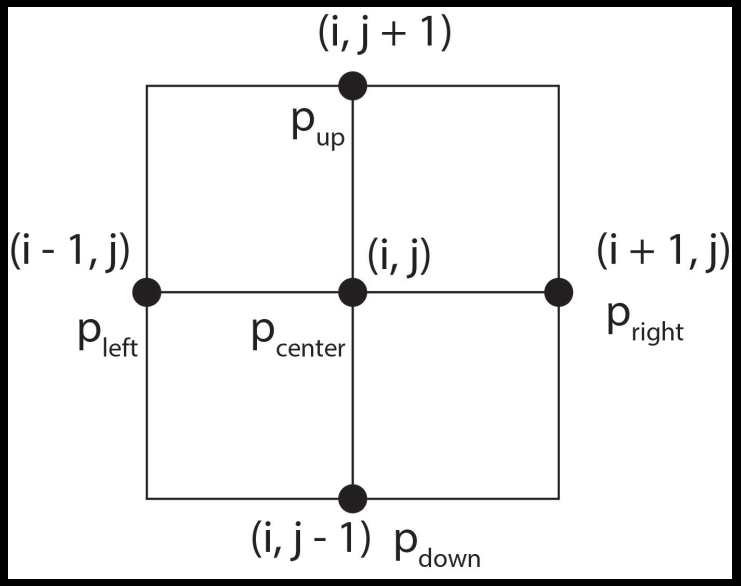
Line mode (“2”)



Triangle mode (“3”)



Smoothing mode (“4”)



1 VBO for center positions and colors

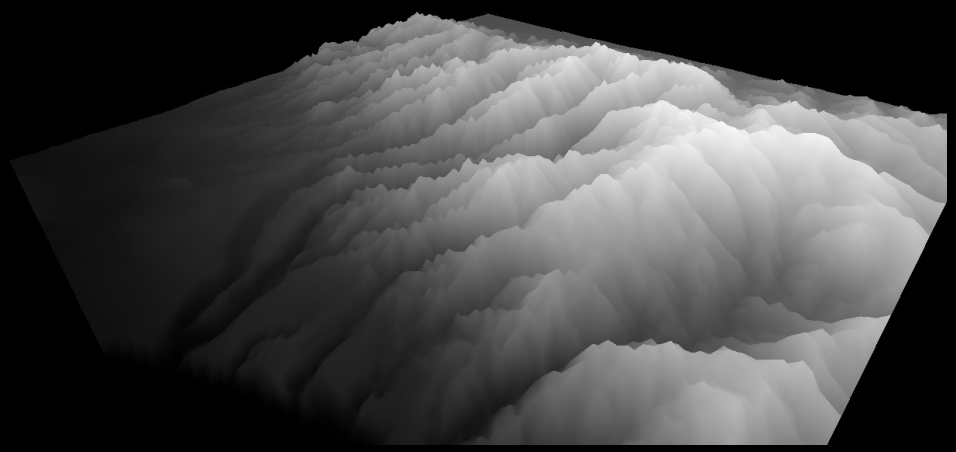
1 VBO for p_{left}
1 VBO for p_{right}
1 VBO for p_{down}
1 VBO for p_{up}

1. Replace (in vertex shader) p_{center} with $(p_{center} + p_{left} + p_{right} + p_{down} + p_{up}) / 5$

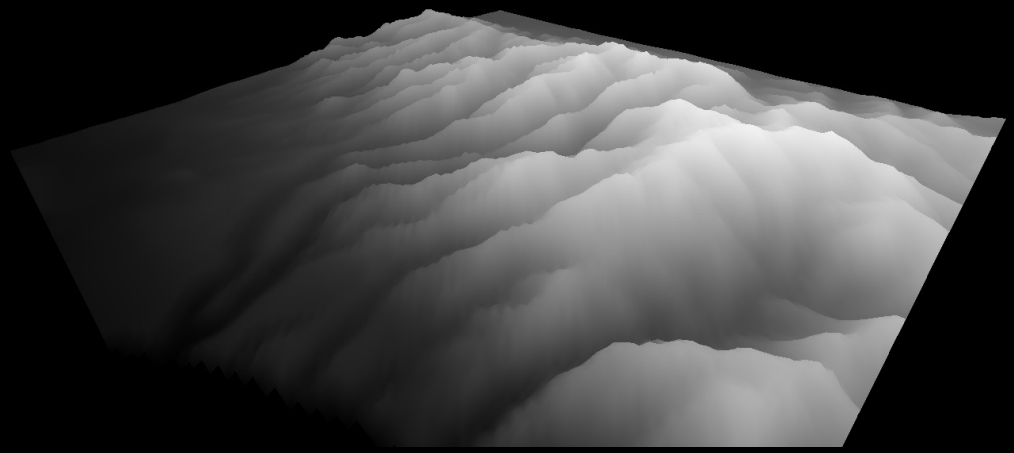
2. Set the grayscale vertex color to the y-coordinate of result.

3. Then, modify the y-coordinate:
y <-- scale * pow(y, exponent)

Smoothing mode (“4”)



unsmoothed



smoothed

The starter code renders a single triangle

- Do not attempt to render a heightfield until you can compile the starter code and can see the single triangle! ☺

- Please read the assignment description (in detail)

- MUST use the OpenGL core profile
  Do not use glBegin(), glEnd(), glVertex3f, etc.