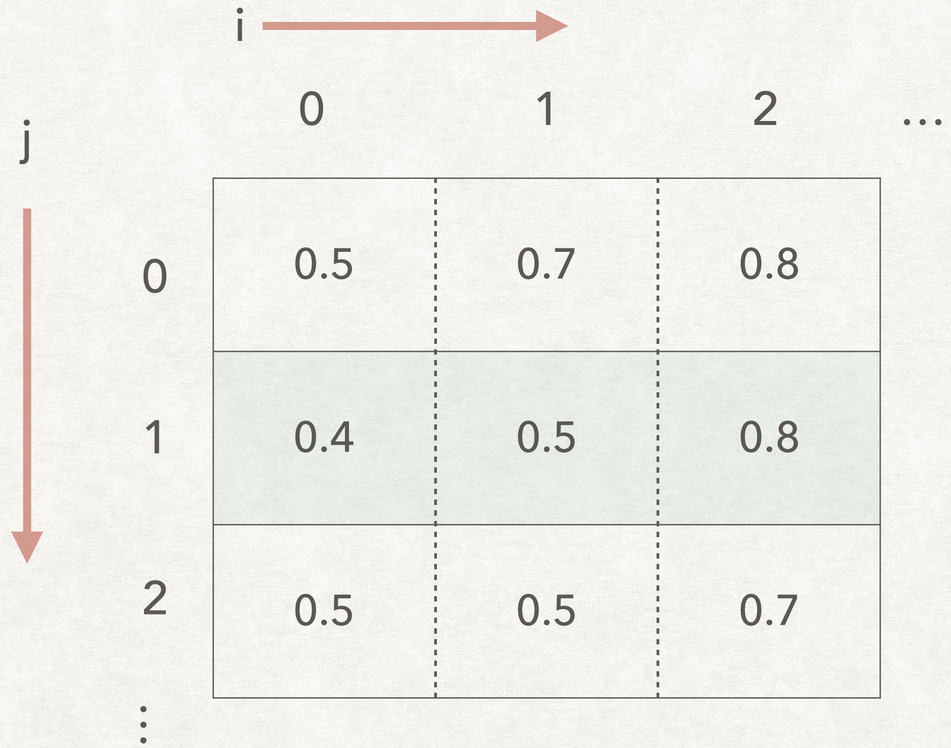
ASSIGNMENT 1 NOTES

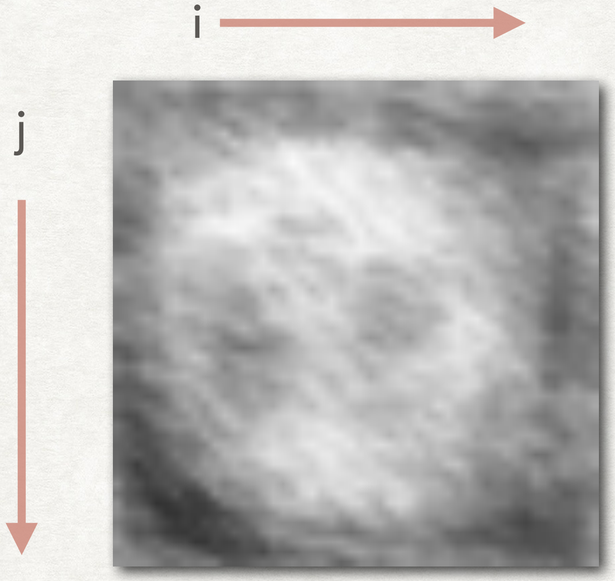
HEIGHT-MAPS

- Create a 3D surface using a height-map.
- The vertex coordinate for the surface are taken from a 2D array.
- The x and z values are taken from the array indices i and j for a[i][j].
- The y values are taken from the value stored in the array.

HEIGHT-MAPS

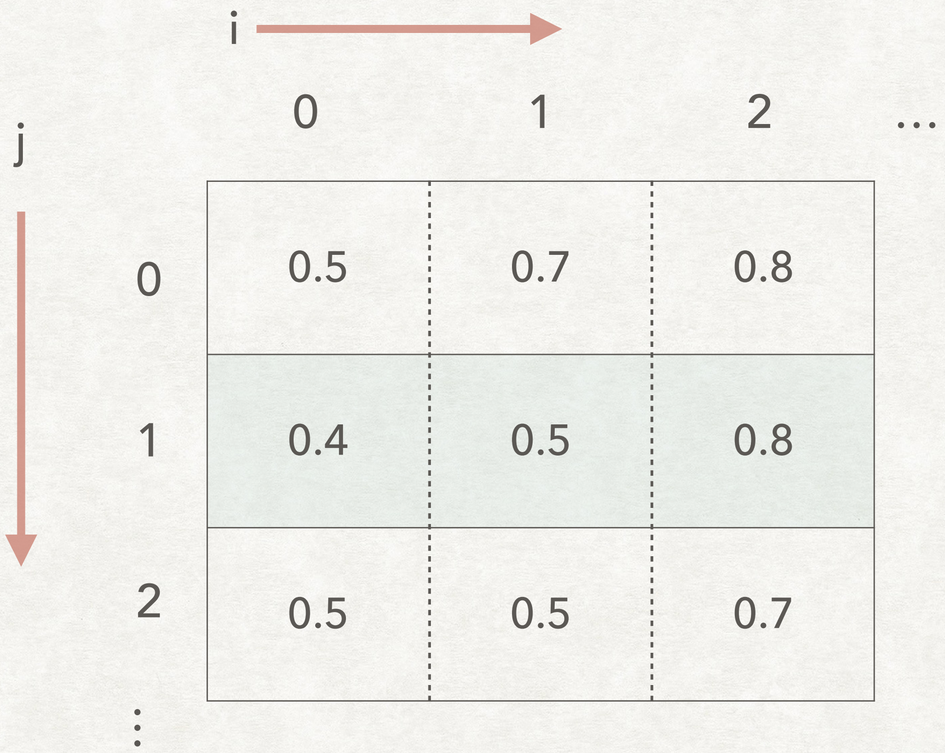
- The height map is a monochrome image loaded into a 2D array.





HEIGHT-MAPS

- Use the array information to create the vertices.



Use adjacent array elements to create the vertices.
-i is used for x coordinates
-j is used for z coordinates
-a[i][j] are used for y coordinates

e.g. The first vertex could be (0, 0.5, 0).

HEIGHT-MAPS

- Use the array information to create the vertices.

To create the geometry for a triangle you will need three adjacent vertices.

The vertices would be:
(0, 0.5, 0)
(0, 0.4, 1)
(1, 0.5, 1)

HEIGHT-MAPS

- Use the array information to create the vertices.

The pair adjacent array elements will create two triangles.

The vertices for the first triangle:
(0, 0.5, 0) (0, 0.4, 1) (1, 0.5, 1)

The vertices for the second triangle:
(1, 0.5, 1) (1, 0.7, 0) (0, 0.5, 0)

Make sure the order of the vertices is always the same. Clockwise or counter clockwise order.

HEIGHT-MAPS

- Use the array information to create the vertices.

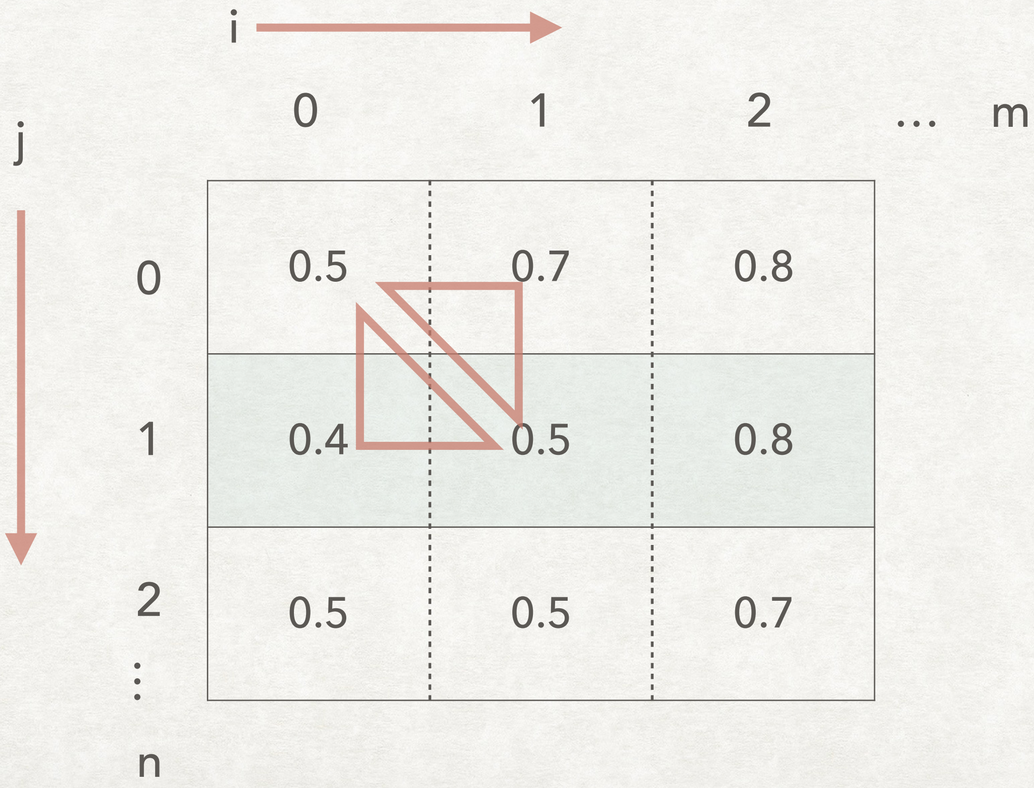
The result will be an array of vertices.

vertices = [x1, y1, z1, x2, y2, z2, …]

vertices = [
0, 0.5, 0, 0, 0.4, 1, 1, 0.5, 1,
1, 0.5, 1, 1, 0.7, 0, 0, 0.5, 0]

HEIGHT-MAPS

- You will need to scale the value so they fit in the view space for the program.
- Vertex values should be scaled between 0.0 and 1.0.



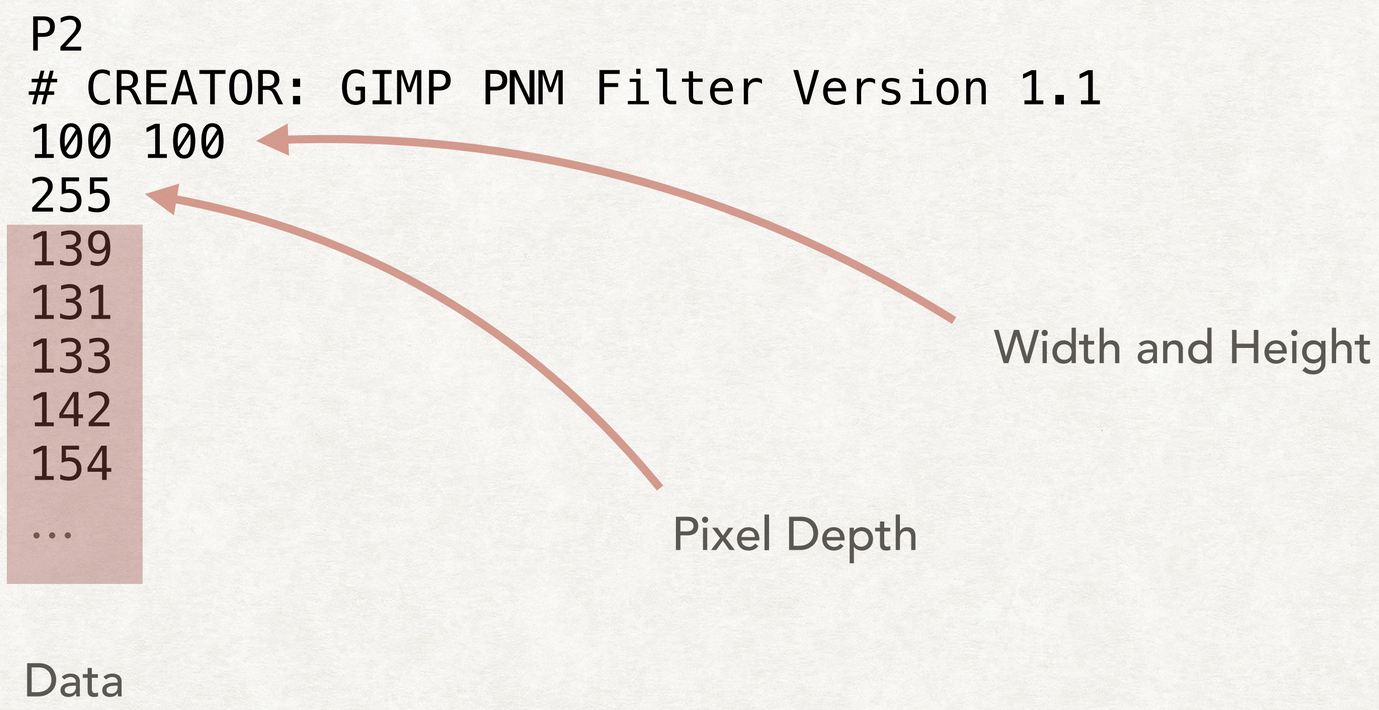
Stepsize between indices is the maximum array index value divided into 1.0.

e.g. if m = 10 and n = 20 then the stepsize would be 1.0/20 = 0.05.

From the previous example, the vertex:
(1, 0.5, 1) (1, 0.7, 0) (0, 0.5, 0)
would be scaled to:
(**0.05**, 0.5, **0.05**) (**0.05**, 0.7, 0) (0, 0.5, 0)

HEIGHT-MAPS

- Vertex y values are scaled in two ways. The values in the input ppm file have a maximum value in the file header.



The data values are divided by the pixel depth. This will scale them to the range of 0.0 to 1.0.

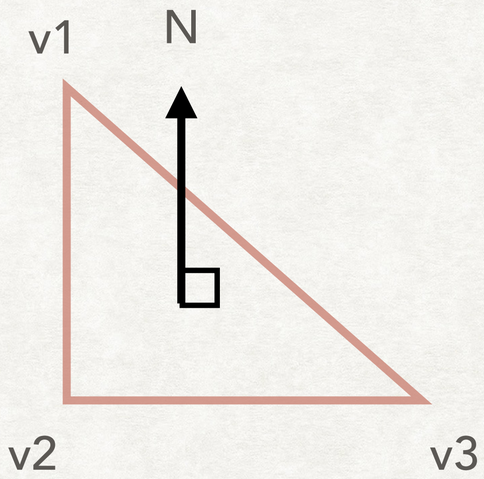
The first pixel is 139 which would be 139/255 so the array value would be 0.545.

HEIGHT-MAPS

- The second way in which y values are scaled is to make the surface they create less rough. If the raw y values are used it is difficult to see the shape of the surface because it is so bumpy.
- Scale the y values in the array so they use a smaller range than 0.0 to 1.0.
- Scaling by 0.3 works reasonably well.
- The previous array value of 0.545 would become 0.164 when scaled by 0.3.

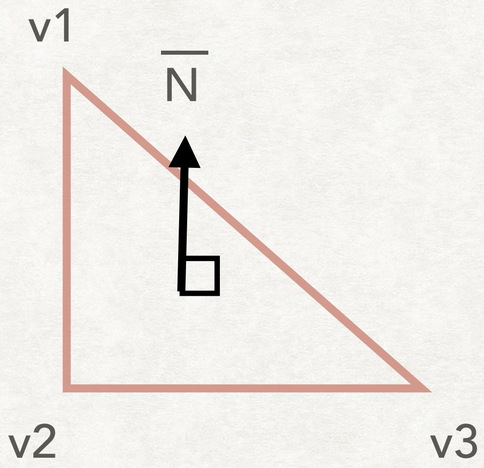
HEIGHT-MAPS

- The next step is to calculate the surface normals using the vertices.
- You calculate the normal using the cross-product and the three vertices for a triangle.



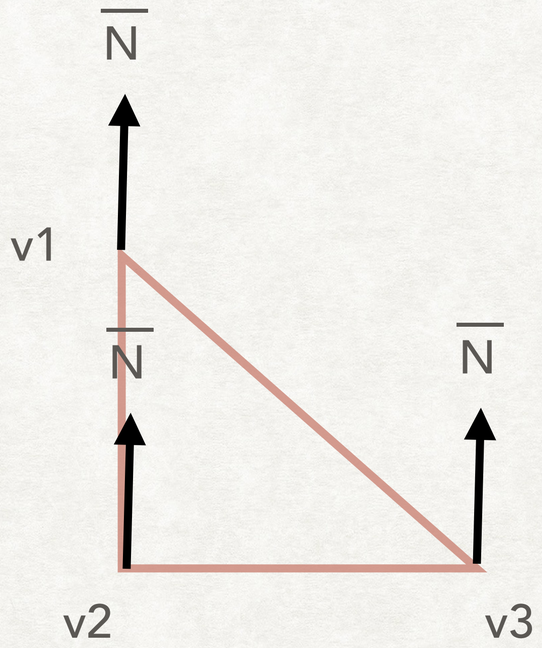
HEIGHT-MAPS

- The normal needs to be a unit normal for the lighting calculations. This means you need to divide the normal values by the normal's length so the length is 1.0.



HEIGHT-MAPS

- Surface normals are the same for all vertices that make up the surface.
- First draw the object using surface normals. Store them in the normals[] array.



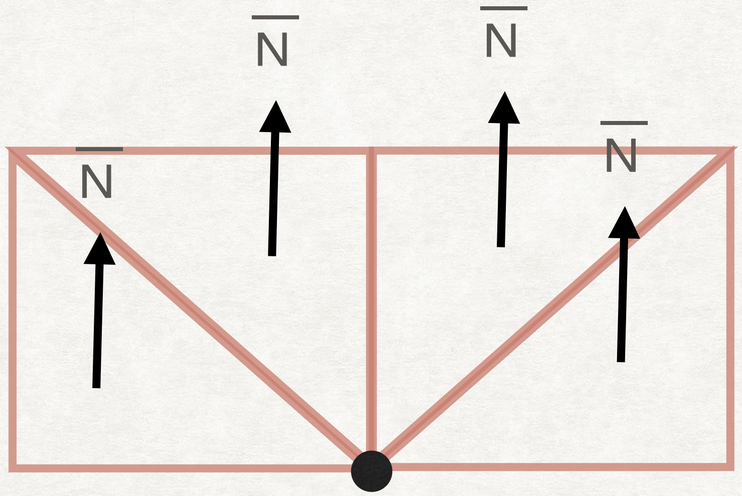
The result will be an (x,y,z) value for the normal.

For the surface normals, each vertex on the triangle will use the same normal.

v1 (x1, y1, z1) N (xn, yn, zn)
v2 (x2, y2, z2) N (xn, yn, zn)
v3 (x3, y3, z3) N (xn, yn, zn)

HEIGHT-MAPS

- Calculate vertex normals by averaging the normals for different surfaces that share the same vertex.

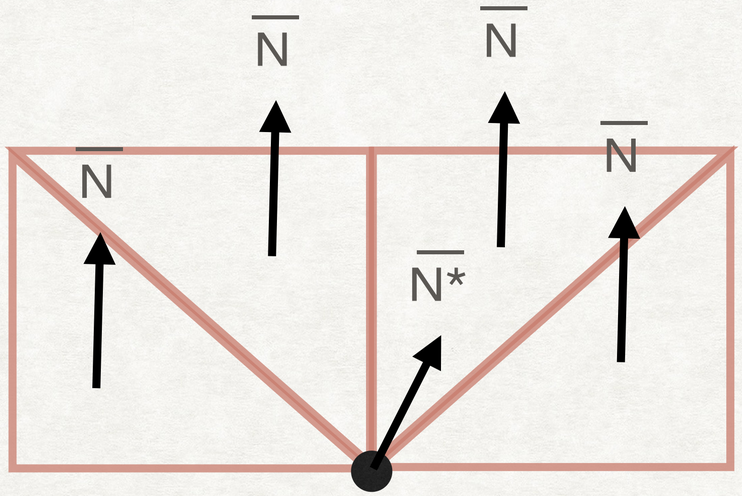


The four polygons share a vertex.

Each polygon has its own normal.

HEIGHT-MAPS

- Calculate vertex normals by averaging the normals for different surfaces that share the same vertex.



The four polygons share a vertex.

Each polygon has its own normal.

Average these to create the vertex normal N* for the shared vertex.

HEIGHT-MAPS

- The indices[] array identifies which vertices and normals are associated with each other.
- For the previous example of three vertices and normals:

v1 (x1, y1, z1)	N (xn, yn, zn)
v2 (x2, y2, z2)	N (xn, yn, zn)
v3 (x3, y3, z3)	N (xn, yn, zn)

- The vertex array would be: [x1, y1, z1, x2, y2, z2, x3, y3, z3]
- The normal array would be: [xn, yn, zn, xn, yn, zn, xn, yn, zn].
- The indices array would contain [0,1,2] which indicates that the first three vertex and normal values combine to create a triangle.