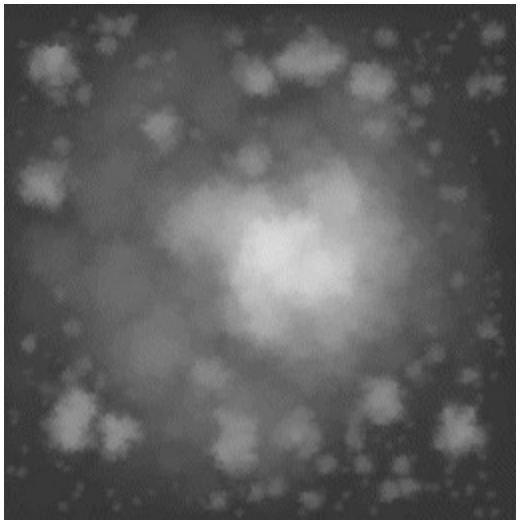
LEI / LCC
Departamento de Informática
Universidade do Minho

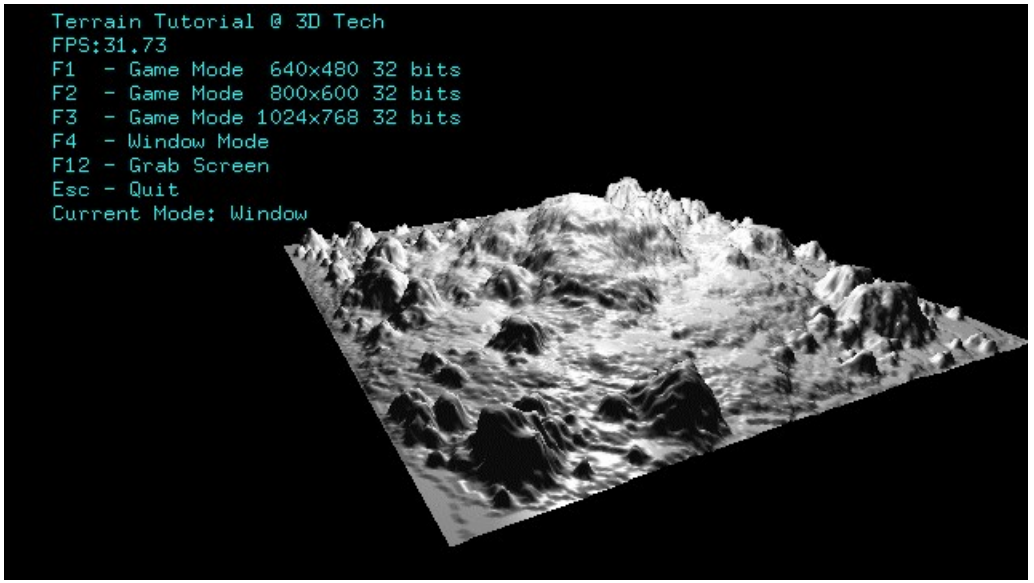
Generating Geometry

Terrains

Height Maps

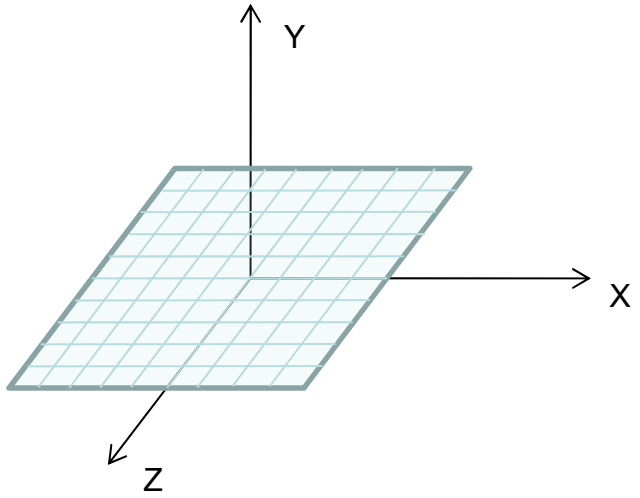
Pixel intensity represents height in a grid





Terrains from Images

- Goal:
  - Given an image, create a regular grid such that the height of each grid point matches the corresponding pixel's intensity.
- Tasks:
  - Load the image
  - Create the grid based on the matrix of pixels read from the image.



DevIL

- Cross-platform image loading library
  (http://openil.sourceforge.net/)

- Convert an image to single channel greyscale (height map)

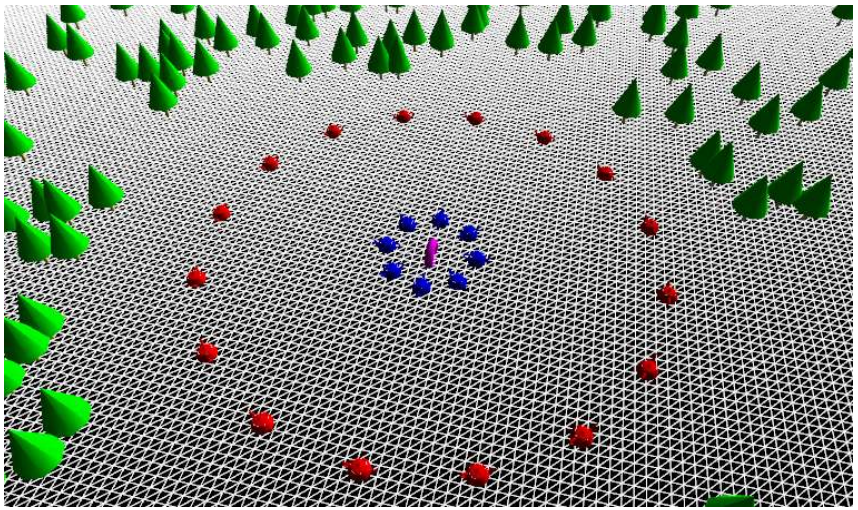
```
ilConvertImage(IL_LUMINANCE,IL_UNSIGNED_BYTE);
```

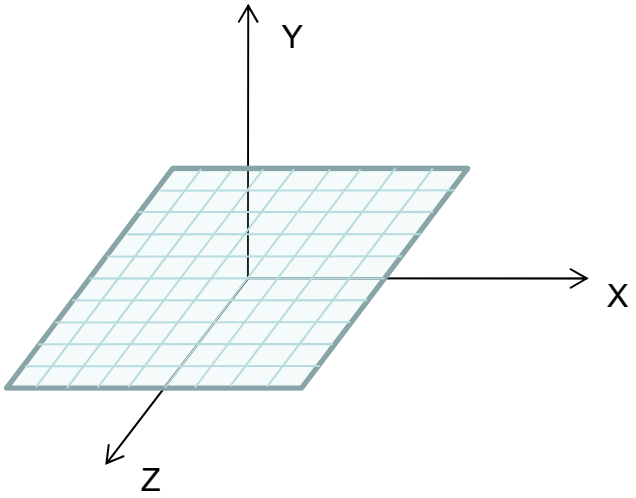
DevIL – Loading an Image

```
#include <IL/il.h>
...
unsigned int t, tw, th;
unsigned char *imageData;

ilGenImages(1,&t);
ilBindImage(t);
ilLoadImage((ILstring)"terreno.jpg");
ilConvertImage(IL_LUMINANCE, IL_UNSIGNED_BYTE);

tw = ilGetInteger(IL_IMAGE_WIDTH);
th = ilGetInteger(IL_IMAGE_HEIGHT);
imageData = ilGetData();
```

Terrains from Images

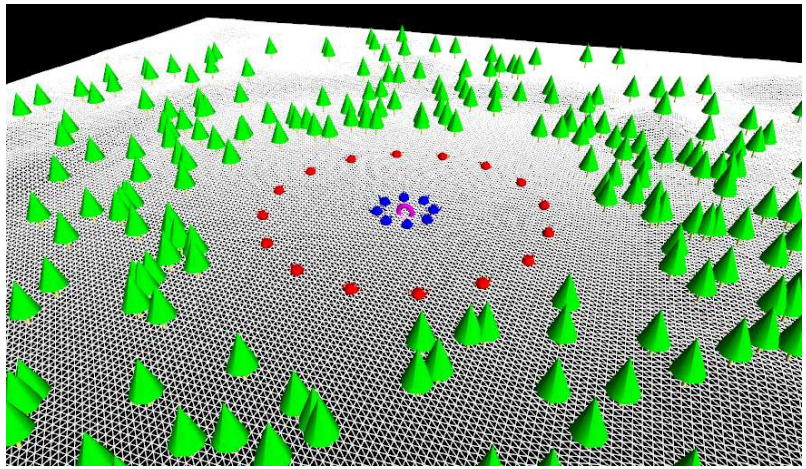
- Step 1:
  - Build a regular grid with height 0.0 for every point. Use the with and height of the image as the input dimensions for the grid: there are as many vertices in the grid as pixels in the image





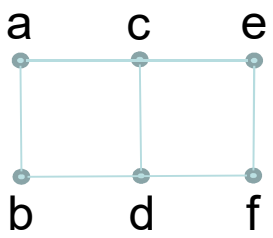
Terrains from Images

- Step 2:
  - Set the height for the grid's vertices according to the pixels intensity
    - `float h(int i, int j)` function to return the value of pixel in column `i`, row `j` (required to build the terrain geometry).
    - `float hf(float x, float z)` function to return the height at any point in the terrain (required to place the trees on the terrain).
    - Note: scale the heights from 0-255 (pixel intensity) to 0 - 30 meters for a more appropriate rendering



Triangle Strips

- Array with triangle vertices: {a,b,c,d,e,f}
- `glDrawArrays(GL_TRIANGLE_STRIP, first, count)`
- Number of strips = image height - 1



VBO - Init

- Step 1 – Allocate and fill arrays with vertices

```
// array for vertices
float *vertexB;
// fill arrays with vertex values
```

- Step 2 - Enable Buffers

```
glEnableClientState(GL_VERTEX_ARRAY);
```

- Step 3: Generate Vertex Buffer Objects

```
GLuint buffers[1];
...
glGenBuffers(1, buffers);
glBindBuffer(GL_ARRAY_BUFFER,buffers[0]);
glBufferData(GL_ARRAY_BUFFER,arraySize, vertexB, GL_STATIC_DRAW);
```

VBO - Render

- Step 1: Define the semantic for each buffer

```
glBindBuffer(GL_ARRAY_BUFFER,buffers[0]);
glVertexPointer(3,GL_FLOAT,0,0);
```

- Step 2 : Draw VBOs

```
glDrawArrays(GL_TRIANGLE_STRIP, first, count);
```

  - first – the starting index
  - count – the number of vertices (not triangles) to draw

Assignment

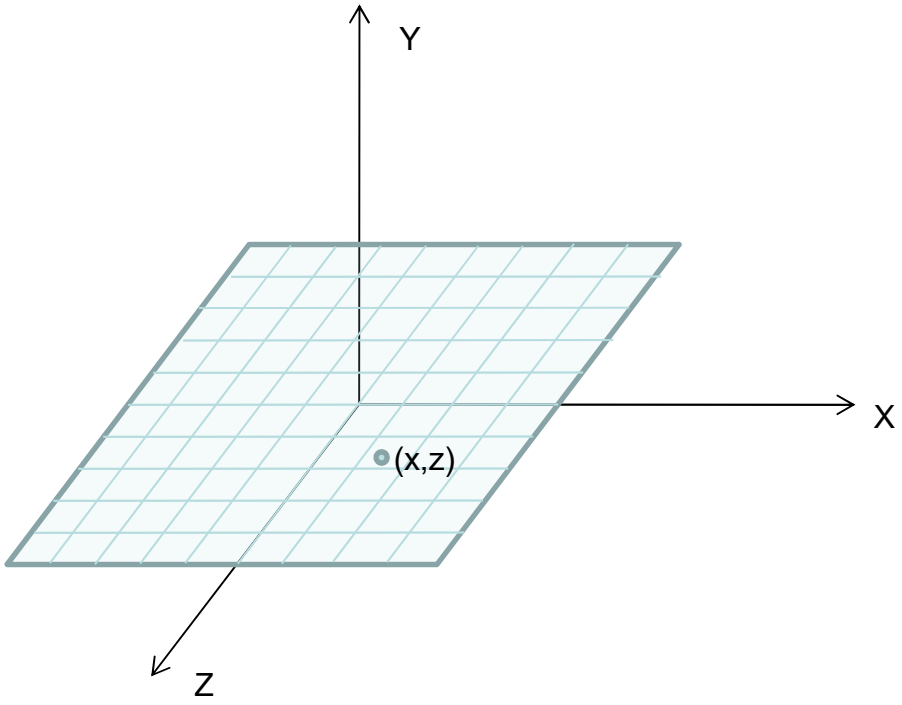
- Given na image, interpret it as a height map, and generate the corresponding terrain.

- Place the trees at the appropriate height

Height at any point in the terrain

- Problem: compute the height of point (x,z) in the grid.

Considering `h(i,j)` as the function that returns the height at the vertices of the grid, we need to be able to compute the height for any point inside the grid.

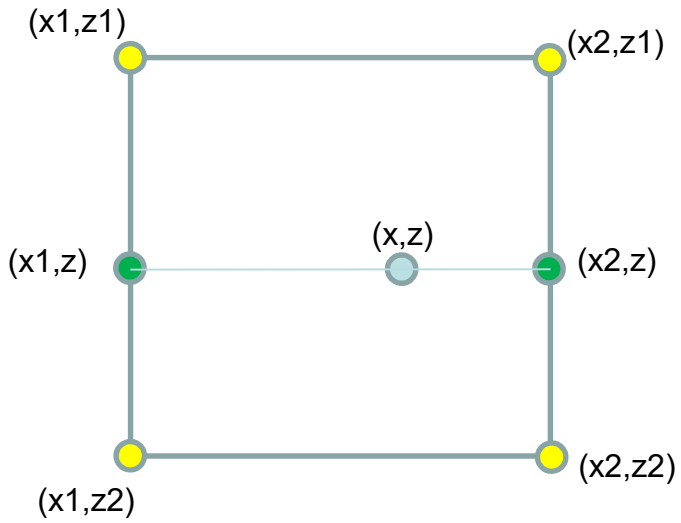


Height at any point in the terrain

- Assume that the grid cells are square and unit length

```
x1 = floor(x); x2 = x1 + 1;
Z1 = floor(z); z2 = z1 + 1;
```

- Function `h` provides access to vertex height values of a grid cell (yellow dots).
- The height at `(x1,z)` can be obtained through linear interpolation of the heights at `(x1,z1)` and `(x1,z2)`. A similar process is used to compute the height at `(x2,z)`.
  - Let `fz` be the fraction part of `z`:
    - `fz = z - z1;  // 0 <= fz <= 1`
  - `h_x1_z = h(x1,z1) * (1-fz) + h(x1,z2) * fz`
  - `h_x2_z = h(x2,z1) * (1-fz) + h(x2,z2) * fz`
- The height at `(x,z)` (blue dot) is computed using linear interpolation between the heights for `(x1,z)` e `(x2,z)` `(green dots)`
  - `height_xz = h_x1_z * (1 - fx) + h_x2_z * fx`



DevIL

Developers Image Library - Usage

In the cpp file:

- `#include <IL/il.h>`

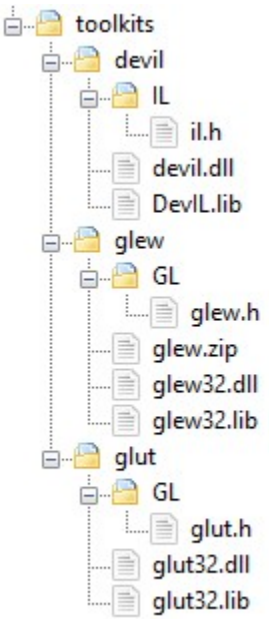
- After GLUT callback registration do:
  - `ilInit();`

- Other Image Libraries can be found here: https://www.opengl.org/wiki/Image_Libraries

DevIL (Windows)

- Add the files in devil.zip to the toolkits folder

- Run Cmake and set “TOOLKITS_FOLDER”



DevIL (Linux & Mac)

- Install Linux
  - `sudo apt-get install libdevil-dev`
- Install MacOS
  - `brew install devil`
- Run CMake