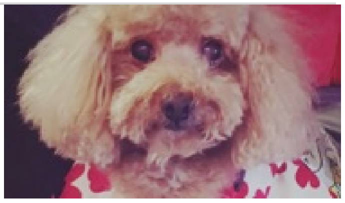
USC University of Southern California

CSCI 420: Exercise 1 [Computer Graphics FS 2018]



Administrative

Lecture URL	http://cs420.hao-li.com
Exercises / Q&A	http://blackboard.usc.edu
Tips	[Ex. 1 tips]

Height Fields

DUE MONDAY SEP 24TH 2018 BY 11:59 PM

Overview

Height fields may be found in many applications of computer graphics. They are used to represent terrain in video games and simulations, and also often utilized to represent data in three dimensions. This assignment asks you to create a height field based on the data from an image which the user specifies at the command line, and to allow the user to manipulate the height field in three dimensions by rotating, translating, or scaling it. After the completion of your program, you will use it to create an animation.

Motivation

This assignment is intended as a hands-on introduction to OpenGL and programming in three dimensions. The starter code we provide is minimal, giving only the functionality to read and write a JPEG image and handle mouse and keyboard input. You must write the code to create a window, handle camera transformations, perform rendering, and handle any other functionality you may desire. We highly recommend the use of GLUT--please see the OpenGL Programming Guide for information, or, **[OpenGL.org]** and **[a page of OpenGL tutors]** or the **[on-line red book]** .

Background

A height field is a visual representation of a function which takes as input a two-dimensional point and returns a scalar value ("height") as output. In other words, a function f takes x and y coordinates and returns a z coordinate.

Rendering a height field over arbitrary coordinates is somewhat tricky--we will simplify the problem by making our function piece-wise. Visually, the domain of our function is a two-dimensional grid of points, and a height value is defined at each point. We can render this data using only a point at each defined value, or use it to approximate a surface by connecting the points with triangles in 3D.

Your Implementation

You will be using image data from a grayscale JPEG file to create your height field, such that the two dimensions of the grid correspond to the two dimensions of the image and the height value is a function of the image grayscale level. Since you will be working with grayscale image, the bytes per pixel (i.e. Pic->bpp) is always 1 and you don't have to worry about the case where bpp is 3 (i.e. RGB images).

Starter code for Visual Studio can be obtained here: **[VS 2010]** **[VS 2012]** **[VS 2017]**.

For Linux users, the starter code is **[here]**, and you will also need the **[pic]** image library which is used to read/write JPEG images. The makefile of the starter code assumes that the pic library locates one level above (i.e. if the starter code is in /home/tom/code/assign1, then the pic library should be in /home/tom/code/pic).

You have to compile the pic library before the starter code. Please make sure you have libjpeg before compiling pic. The libjpeg can be obtained by "sudo apt-get install libjpeg62-dev". Here is a sample sequence of Ubuntu commands that get everything compiled:

```
> tar -xvf pic.tar.gz
> tar -xvf assign1_starterCode_linux.tar.gz
> cd pic
> make
> cd ..
> cd assign1
> make
> ./assign1 spiral.jpg
```

For Mac OS X, starter code is **[here]**, and you will also need the **[pic]** image library. Before you do any coding, you must install command-line utilities (make, gcc, etc.). On Mac OS X Lion or newer, install XCode from the Mac app store, then go to XCode, and use "Preferences/Download" to install the command line tools. In Mac OS X Snow Leopard or older, you must install Mac OS X Developer Tools from the Mac OS X DVD. The makefile of the starter code assumes that the pic library locates one level above (i.e. if starter code == /Users/tom/code/assign1, then pic library should be in /Users/tom/code/pic). Please compile the pic library before compiling the starter code. Here is a sample sequence of commands that get everything compiled:

```
> unzip pic_MacOS.zip
> unzip assign1_starterCode_macOS.zip
> cd pic
> make
> cd ..
> cd assign1
> make
> ./assign1 spiral.jpg
```

Note: Under Mac OS X Mountain Lion, the X11 libraries no longer ship by default with the OS. If you get an error like this, you need to install XQuartz:

```
xpic.c:21:47: error: X11/Xlib.h: No such file or directory
xpic.c:22:23: error: X11/Xutil.h: No such file or directory
xpic.c:23:42: error: X11/Xos.h: No such file or directory
```

See this Apple **[post]** for how to install XQuartz. You also need to set your path (you can put it into your .bashrc to avoid typing it each time you open the shell).

```
 export CPPFLAGS="-I/opt/X11/include" LDFLAGS="-L/opt/X11/lib"
```

Please email the TA if you have trouble compiling the starter code. Note that the starter code includes an "exit" function call in the middle of the main function. On MS Visual Studio, you will see a console window appear briefly and then close right after that. Regardless of what operating system and platform you use to solve the assignment, you need to remove the exit function call and replace it with proper OpenGL initialization.

Grading Criteria

Your program must:

- Handle at least a 256x256 image for your height field at interactive frame rates (window size of 640x480). Height field manipulations should run smoothly.
- Be able to render the height field as points, lines("wireframe"), or solid triangles (with keys for the user to switch between the three).
- Render as a perspective view, utilizing GL's depth buffer for hidden surface removal.
- Use input from the mouse to spin the heightfield around using glRotate.
- Use input from the mouse to move the heightfield around using glTranslate.
- Use input from the mouse to change the dimensions of the heightfield using glScale.

- Color the vertices using some smooth gradient.
- Be reasonably commented and written in an understandable manner--we will read your code.
- Be submitted along with JPEG frames for the required animation (see below).
- Be submitted along with a readme file documenting your program's features, describing any extra credit you have done, and anything else that you may want to bring to our attention.

Animation Requirement:

After finishing your program, you are required to submit a series of JPEG images which are screenshots from your program. Functionality to output a screenshot is included in the starter code, and assumes you are using a window size of 640x480--your JPEG images must be this size. Please name your JPEG frames 000.jpg, 001.jpg, and so on, where 000.jpg is the first frame of your animation, and please do not exceed 300 frames. Expect a frame rate of 15 frames per second, which corresponds to 20 seconds of animation running time maximum.

There is a large amount of room for creatitiy in terms of how you choose to show your results in the animation. You can use our provided input images, or modify them with any software you wish, or use your own input images. You may also use your animation to show off any extra features you choose to implement. Your animation will receive credit based on its artistic content, whether pretty, funny, or just interesting in some manner.

We will compile a video of all student submissions and show it in class. Optional: If you would like to convert your frames to a video by yourself, you can use **[Adobe Premiere]**, **[ffmpeg]**, **[QuickTime Pro]**, or **[Windows Movie Maker]**.

Submission

Please zip your code and JPEG images into a single file and submit it to Blackboard. After submission, please verify that your zip file has been successfully uploaded. You may submit as many times as you like. If you submit the assignment multiple times, we will grade your LAST submission only. Your submission time is the time of your LAST submission; if this time is after the deadline, late policy will apply to it.

Tips

- Make sure the starter code works on your machine.
- Familiarize yourself with GL's viewing transformations before attempting to render the height field itself. Try rendering a simple object first. If the camera is set up correctly, you should see an image like the **[following]**.
- Do not mix up GL_MODELVIEW and GL_PROJECTION.
- For glFustrum and gluPerspective, the near and far values for clipping plane have to be *positive*. You will see weird problems if they are zero or negative.
- Finish your program completely before worrying about the animation.
- One way to optimize your program is to minimize the number of calls to glBegin(GL_TRIANGLES) and glVertex3f. For example, you should only call glBegin(GL_TRIANGLES) once at the beginning of your triangles and glEnd() at the end instead of calling glBegin(GL_TRIANGLES) for every single triangle. To further optimize your program, you can use GL_TRIANGLE_STRIP or GL_TRIANGLE_FAN.
- Don't try to do this at the last minute. This assignment is supposed to be fun and relatively easy, but time pressure has a way of ruining that notion.
- Don't over-stretch the z-buffer. It has only finite precision. A good call to setup perspective is:

  ```
  gluPerspective(fovy, aspect, 0.01, 1000.0);
  ```

  A bad call would be:

  ```
  gluPerspective(fovy, aspect, 0.0001, 100000.0);
  ```

  or even worse:

  ```
  gluPerspective(fovy, aspect, 0.0, 100000.0);
  ```

- In the last two examples, the problem is that the ratio between the distance of the far clipping plane (=last parameter to gluPerspective), and the distance of the near clipping plane (=third parameter to gluPerspective) is way too large. Since the z-buffer has only finite precision (only a finite number of bits to store the z-value), it cannot represent such a large range. OpenGL will not warn you of this. Instead, you will get all sorts of strange artifacts on the screen and your scene will look nothing like what you intended it to be.

- **On JPEG Types:** pic.h supports JPEG images with one, three, or four bitplanes (i.e. Pic->bpp can be 1,3, or 4). In other words, a single pixel may have either one, three, or four bytes for its intensity values. All JPEG images given in this assignments have one byte per pixel, and you don't need to worry about images with three or four bitplanes. If you have assumed one byte per pixel and happen to give a three- or four-bitplane JPEG file to your program, you will get incorrect results, at best. For your information, to convert images from RGB (bpp=3) to grayscale (bpp=1) in Windows or Mac, you can use Photoshop (**[Image->Mode->Grayscale]**). In Linux, you can use GIMP (**[Image->Mode->Grayscale]**).

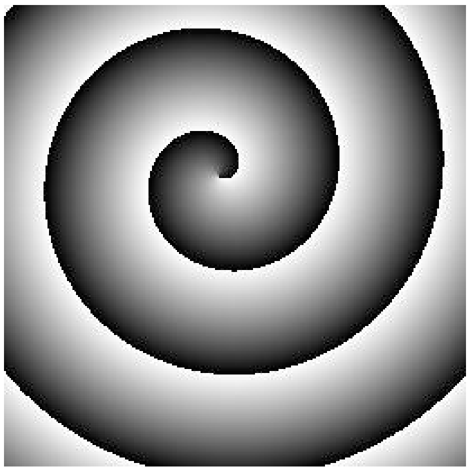
Extras

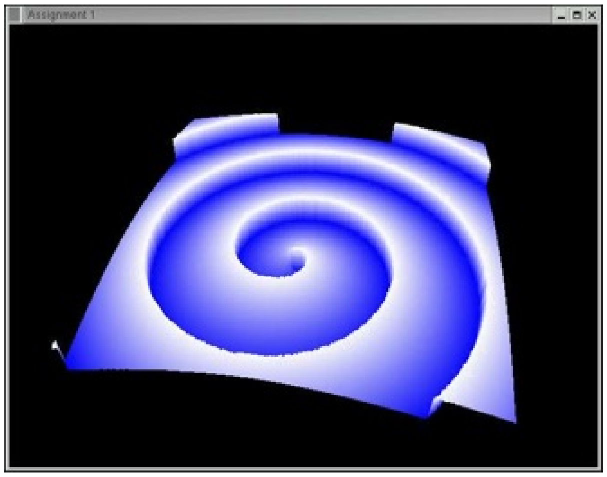
You may choose to implement any combination of the following for extra credit.

- Experiment with material and lighting properties.
- Support color (bpp=3) in input images.
- Render wireframe on top of solid triangles (use glPolygonOffset to avoid z-buffer fighting).
- Color the vertices based on color values taken from another image of equal size. However, your code should still support also smooth color gradients as per the core requirements.
- Texturemap the surface with an arbitrary image.
- Allow the user to interactively deform the landscape.

Please note that the amount of extra credit awarded will not exceed 10% of the assignment's total value.

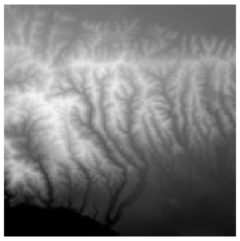
Examples and inputs


input (source image)



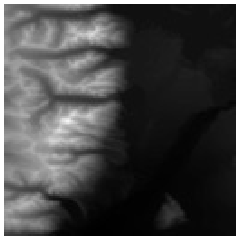
output (height field)

more inputs (real world data)



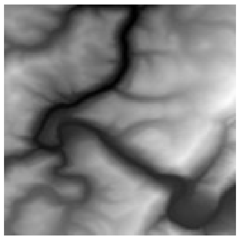
[128] [256] [512] [768]

Santa Monica Mountains
Min elevation: 0m / 0ft
Max elevation: 638m / 2093.17ft
Image size: 15.9km x 15.9km / 9.8miles x 9.8miles



[128] [256] [512] [768]

Grand Teton National Park
Min elevation: 1936m / 6351.71ft
Max elevation: 4200m / 13779.52ft
Image size: 27.8km x 27.8km / 17.2miles x 17.2miles



[128] [256] [512] [768]

Ohiopyle State Park
Min elevation: 326m / 1069.55ft
Max elevation: 712m / 2335.95ft
Image size: 7.1km x 7.1km / 4.4miles x 4.4miles

Data available from U.S. Geological Survey, Earth Resources, Observation and Science (EROS) Center, Sioux Falls, SD.