Praktikum: Echtzeit Computergrafik

Assignment 3 – *"Texture Generation"*

10 Points

In this assignment, we will replace the texture and normal generation in your existing TerrainGenerator with your own code.

All code created in this assignment will replace the GEDUtils::TextureGenerator call from the last assignment. We highly recommend to structure your code into several small functions or classes for better overview!

Normal Calculation (3P)

- As described in the slides, use central differences to calculate a normal field from the height field (**-1P** if you forget to normalize!)
- Don't forget special treatment for the border values (**-1P** otherwise)
- Save the created normal map using the SimpleImage class.
  - Use the filename given as a command line argument as described in the last assignment!
  - Note that SimpleImage now saves .tiff files as noted in the slides!

Texture Blending (5P)

To generate a color texture for our terrain, multiple predefined textures will be added together depending on the underlying terrain.

- Create a function that calculates the blend weights as described in the slides. The weights should be calculated for each pixel using a given terrain height and slope for a fixed number of textures (4 in our example). (**2P**)
- Create the final color texture by blending the four seamless material textures into a single one using the blend weight function. (**3P**)
  - If you use the textures from "`external`", **do not** copy them to your own repository. Instead use paths relative to the current working directory; e.g. "`../../../../external/textures/gras15.jpg`".
  - Use seamless texturing to create higher resolution outputs by tiling the input textures. **-1P** if you just use the texture without any tiling!
  - If you start your project from within Visual Studio, the default working directory is set to where your .vcxproj file resides (`$ProjectDir`). Change this in the project properties by setting the Working Directory in Debugging to

`$(TargetDir)`. Hint: This is stored in your .user file, add this file to your Git repository.

- Save the color texture to the file given by the command line arguments.

Heightfield Downsizing (1P)

Since the small terrain details are now given by the normal map, we don't need the original high-resolution terrain any more.

- As a last step, generate a heightfield of 1/16th the resolution of the original heightfield (e.g. from a 4096² input heightfield create a 1024² output heightfield).
  - The new height value of each pixel is given by averaging the height values of 4x4 pixels
- Save the newly created, smaller heightfield instead of the full resolution one. The old heightfield can be discarded afterwards as it is no longer needed.

Simple Content Pipeline (1P)

To reduce the loading time for our final game, all resources are converted into a custom format beforehand. The texture converter is part of your solution and will be built into "$(OutDir)\texconv.exe". To automate the conversion process, we will call all conversion tools during the build step with a custom makefile project.

- Add a new makefile project ResourceGenerator to your existing solution. To do this, select "General -> Makefile project" instead of "Win32 Console Application" in the project creation wizard. Create the project in the "Projects" subdirectory! You can skip all the settings (just click "finish").
- In the properties of the newly created project, select "NMake"
- Set the Build Command Line (don't forget to do this for **all** build configurations) to call your terrain generator and the texture converter. You can find an example in `external/templates/nmake_build.txt` to copy & paste.
  Hint: You will need to explicitly click on "Edit" in the drop down box for the Build Command Line to paste multiply text lines!

  Background information: The provided build command line will call the TerrainGenerator. The temporary TIFF files for colors and normals are created in the "intermediate" directory $(IntDir) (which by default resides in the $(Configuration) subdirectory in the project directory). The heightfield is directly created in the resources directory "$(OutDir)resources". It will then call the texture converter to convert the color and normal textures to DDS and store them in the resources directory.
- Set the Clean Command Line to delete all temporary and final resources. You can find an example in `external/templates/nmake_clean.txt` to copy & paste.
- Set the makefile project to depend on the TerrainGenerator and the texture converter (texconv): Right-Click your solution -> Properties -> Project Dependencies, select

"ResourceGenerator" and check "TerrainGenerator" and "texconv" in the dependency list

- Build the newly created project. As a result, the terrain heightmap should be created as a TIFF file and the color- and normal-map should be created as DDS files in `$(OutDir)resources`, where `$(OutDir)` is either `<solution_directory>\Debug\` or `<solution_directory>\Release\`. Test this for both configurations! Also try this with a new Git Clone!