Code abstract

Jonathan Granier

July 12, 2018

This is a short abstract of the code that explain how the code work in general.

1 Prototype abstract

The prototype take as input *DEM* files (.asc) and make a 3D render of this. The main goal is generate the shading and the shadow of the mountains and display them on a mesh. For generate the shading , the main idea is generate a multi scale with a laplacien pyramid and on each scale , locally directs the light with the slant.

2 Explanation of the code

It's a OpenGL code with two parts. The C++ code (in the *cpp* folders) and the shader (in *shaders* folder).

2.1 C++

This code is split in 2 parts. The UI in the *MainWidnow* folder and the engine in the *OpenGL* and *LightCamera* folder.

2.1.1 MainWindow

The UI of the prototype. All QT code are in this folder.

- main.cpp : Just start the app and run mainwindow.
- mainwindow.cpp : Setup the UI and start viewer. Do the connection between the UI and the viewer.
- mainwindow.ui : A UI files generate by QTdesigner.
- viewer.cpp : The real main class of the project. Make the link between the UI and the engine. Setup the opengl context. Controle the light , the camera and the scene. Select wich part of the pipeline will be display on the screen (Drawmode).

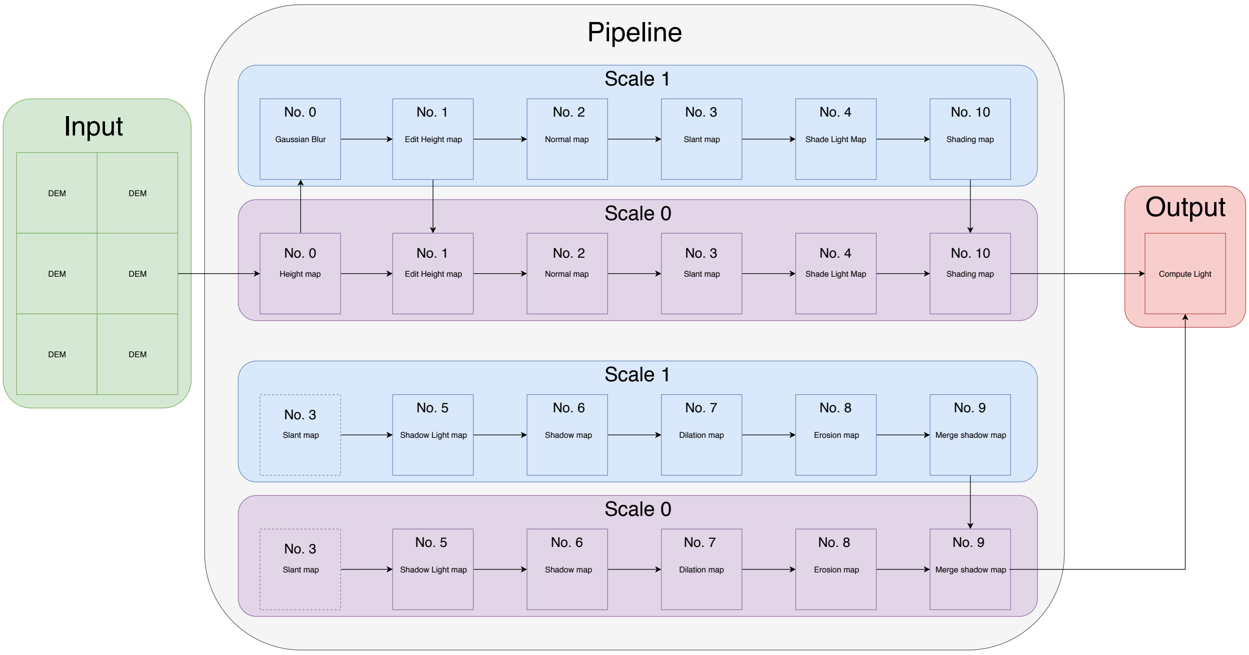
2.1.2 Light Camera

Just vector and matrix computation for manage a camera and a light

- Light : Define by the 2 Euler's angle yaw and pitch.
- Camera and Trackball : A Camera that move around a central point.

2.1.3 OpenGL

- Scene : Manage the multi scale, and the computation order .
- Scale : Manage a scale.
- mesh,vertex : Manage the mesh.
- meshloader : translate a DEM to a heightMap (texture) and a heightMap to a mesh.
- Textures : Load or generate a texture, send the texture to GPU.



2.1.4 Pipeline

- MainWindow : UI
- Viewer : Central point, select the texture to display
- Scene : Manage the computation order of the multi-scale and the shared settings.
- Scale : Manage a unique scale, uniform to send to the shaders.

2.2 Shaders

All the computation are done in the image space.

The shaders

- No. 0. gaussBlur : Do a Gaussian blur on a height map. First part of the Laplacien pyramid.
- No. 1. editheightmap : Do the difference beetween two heightMap. Second part of the Laplacien pyramid.
- No. 2. normalmap : Compute the normals of a height map.
- No. 3. slantmap : Compute the slant map of a height Map.
- No. 4. shadelight : Directs localy the light with the slant. Compute the light for the shading only.
- No. 5. shadowlight : Directs localy the light with the slant. Compute the light for the shadows only.
- No. 6. shadowmap : Compute the shadow map with a raymarching.
- No. 7,8. morpho : Make a mathematical morphology on a shadow map.
- No. 9. mergeshadow : Merge two shadows maps with a lineare interpolation.
- No. 10. shading : Compute the shading from the normalMap and the local light.
- No. 11. ComputeLight : Only shader with the mesh. Mix the mesh with the shading , the shadows and the color

- genheightMap : Out of the pipeline. Generate a height map in level of gray
- drawTexture : Out of the pipeline. Shader for draw a unique texture.

The textures :

- No. 0. HeightMap : level of gray between 0 and 4809 (height of the *Mont blanc*).
- No. 1. EditHeightMap : level of gray between 0 and 4809(height of the *Mont blanc*).
- No. 2. NormalMap : 3D vector , y up.
- No. 3. SlantMap : 2D vector + size of the vector in blue chanel.
- No. 4. ShadelightMap : 3D vector.
- No. 5. ShadowlightMap : 3D vector.
- No. 6. ShadowMap : Boolean value in the red chanel. 0–> shadow , 1–> no shadow.
- No. 7. DilationMap : Boolean value in the red chanel. 0–> shadow , 1–> no shadow.
- No. 8. ErosionMap : Boolean value in the red chanel. 0–> shadow , 1–> no shadow.
- No. 9. mergeshadowMap : Value between 0 and 1 in red chanel.
- No. 10. shadingMap : level of gray between 0 and 1.