Pandora

COMS 557 OpenGL Project

Supriya Raul | UID - 401341969
Iowa State University | Computer Graphics and Geometric Modeling| Fall 2017

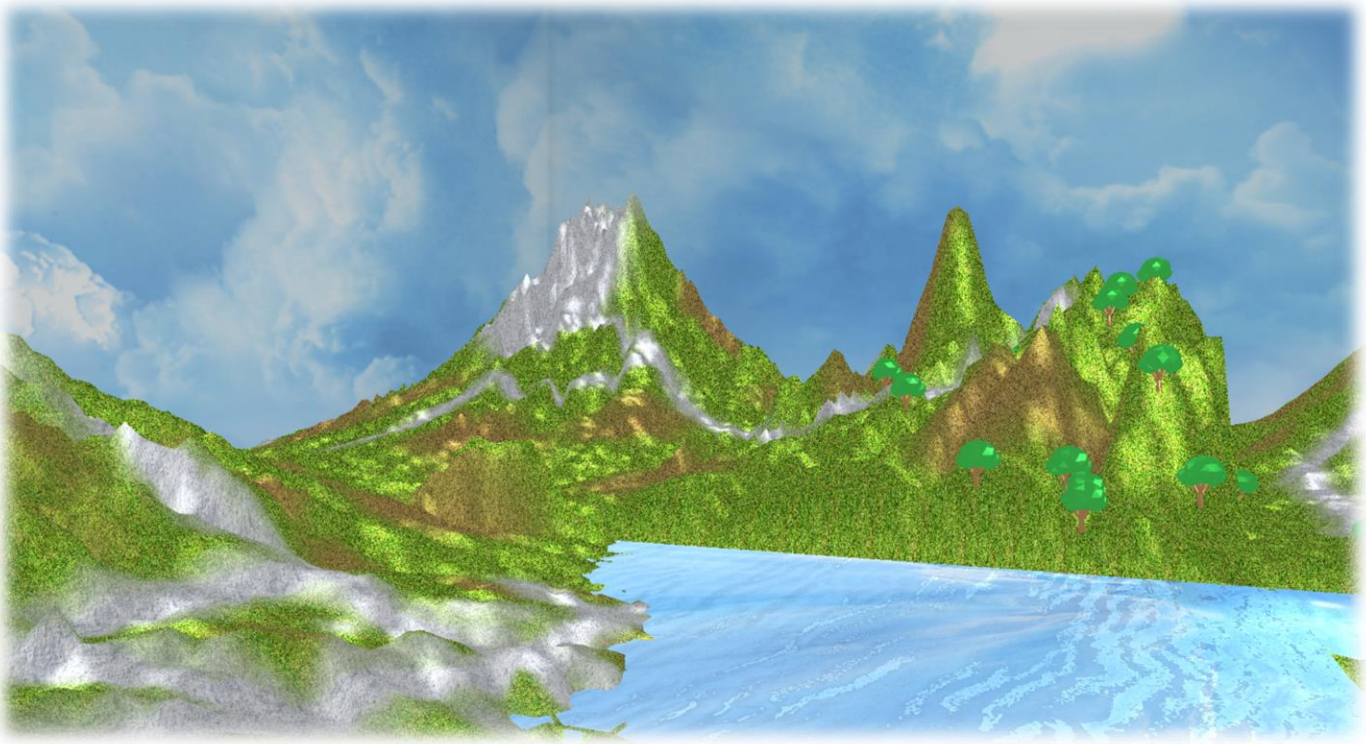


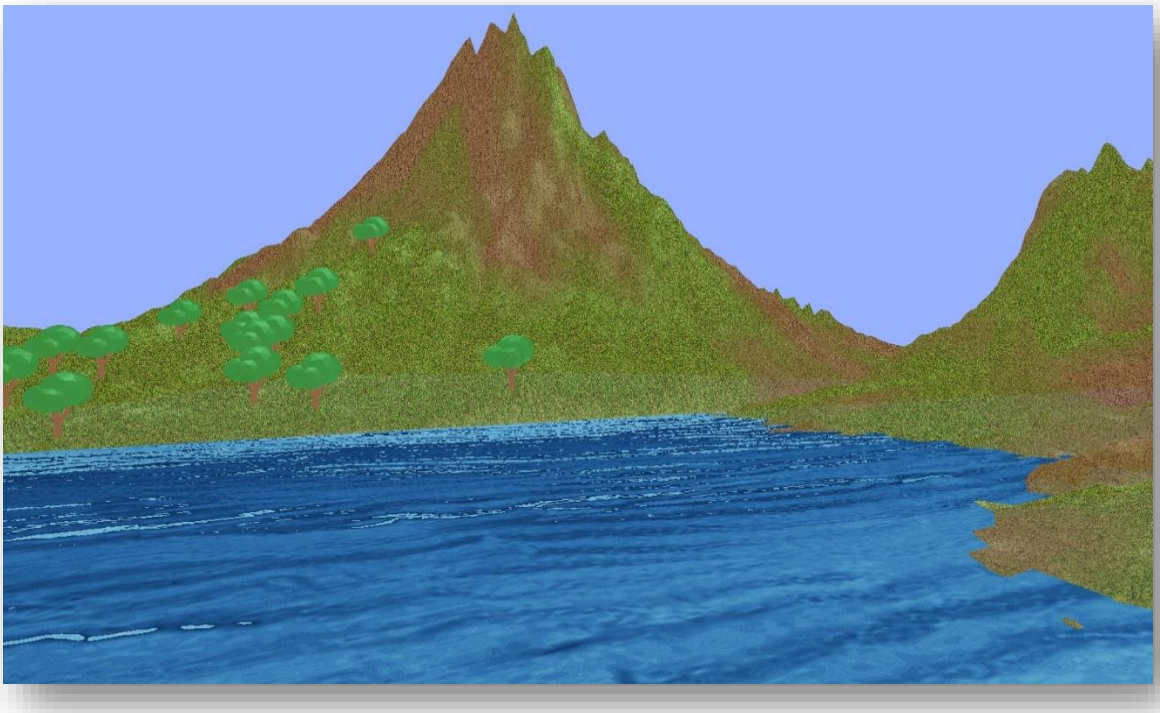
Table of Contents

Table of Screenshots

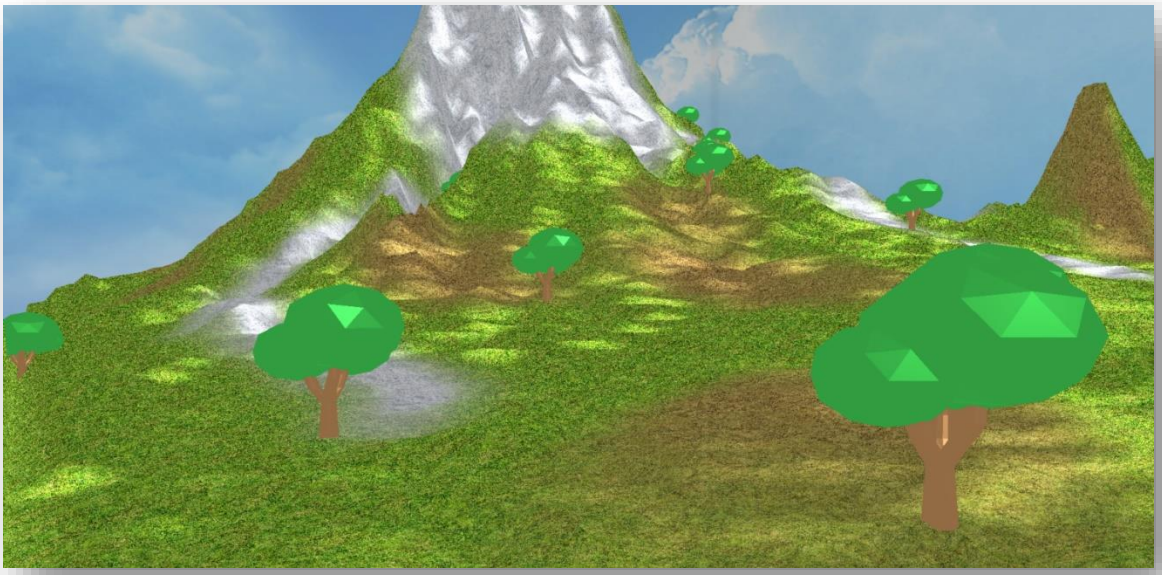
Screenshots

Terrain Height Mapping + Multitexture + No light:

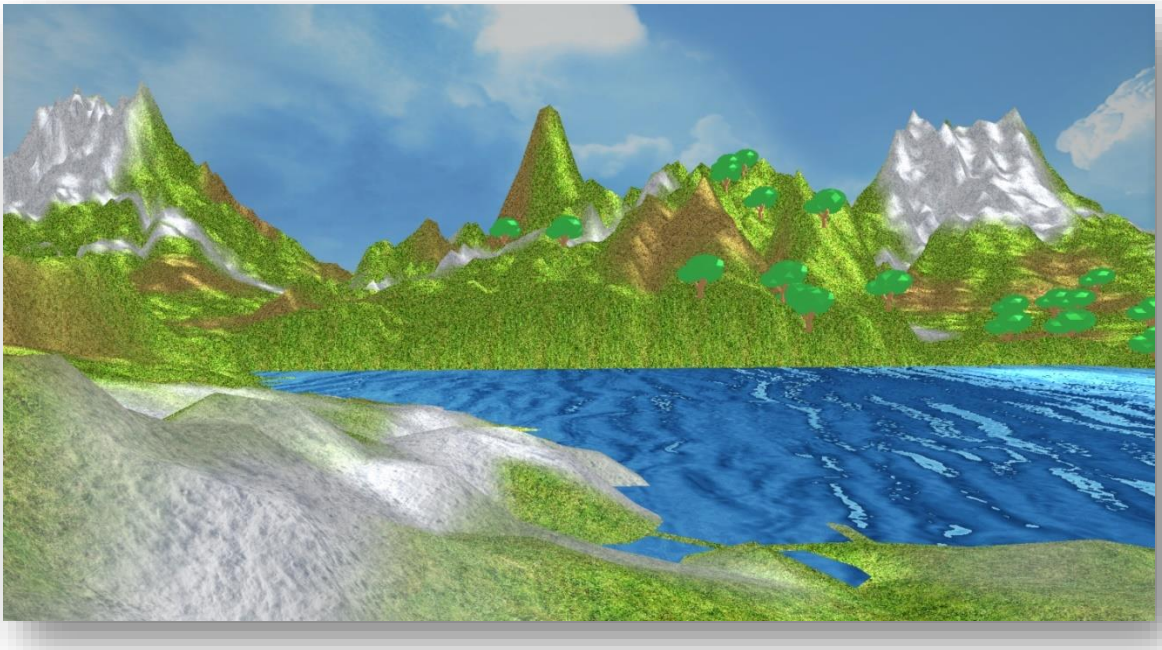
Displacement Mapping + Multitexture + No light:



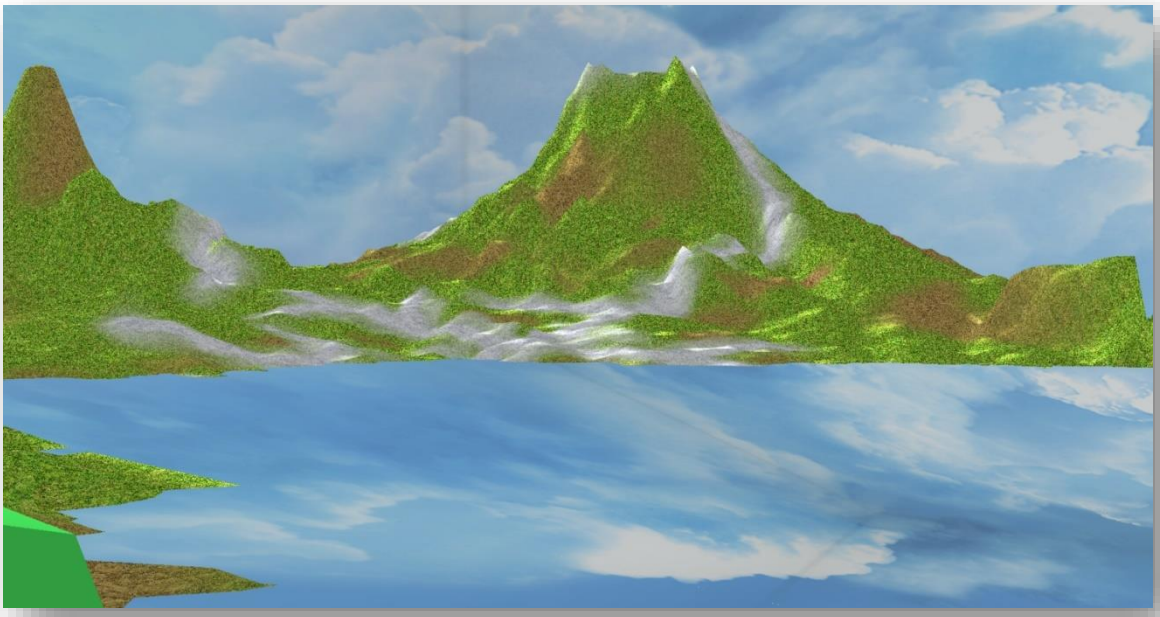
Model Loading(Tree) + Tree placement based on terrain height + Directional Light:



Light Effect + Skybox:

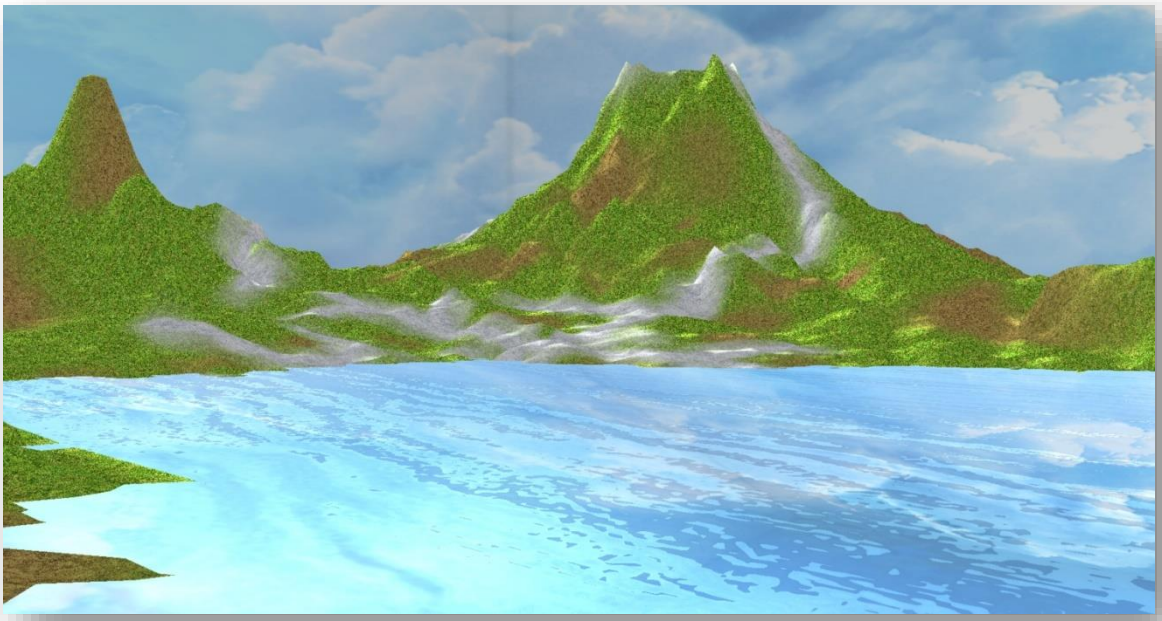


Environment Mapping:



Environment Mapping + Displacement Mapping:

Environment Mapping + Displacement Mapping + Light Effect:



1. Introduction

As stated goal of the project was to develop a model highlighting the concepts taught in the classroom lectures. A terrain with some mountains, trees, a water body and an environment were targeted to incorporate some eye-catching attributes. Concepts such as multi-texturing, object loading, height mapping, displacement mapping, environment mapping and lighting are applied to achieve the results.

2. Methodology

2.1. Implementation Details

The project is designed following object-oriented approach. Every object in the scene is implemented as a stand-alone object and it is added to the scene at runtime. This has allowed me to implement effects with great flexibility. Once the objects are created, the rendering is performed in the right order to obtain the desired effect.

First, in the main function a window is created by utilizing the functions provided by GLFW and our viewport dimensions are set. Next, light, material and texture objects are created to be used to set the appearance of visible objects in the scene. The terrain, river, tree and skybox objects are built next and their appearances are set. Each of these objects contain a draw function which renders them as the program is executed.

2.1.1. External Libraries

OPENCV: OpenCV image processing and image transformation functions are used to read height map image. Version opencv-3.3.1-vc14 has been used as it only is compatible with Visual Studio community 2017. It needs to be included manually in the project properties. An environment variable has been created for it. Variable name: OPENCV_DIR , variable value: S:\opencv\build\

Commands used to link this library in visual studio 2017:

- Configuration properties/VC++ directories/ Include directories --- **$(OPENCV_DIR) include**
- C/C++/General/Additional Include directories --- **$(OPENCV_DIR)include**
- VC++ directories/ Library directories --- **$(OPENCV_DIR)x64**
- Linker/Input/additional dependencies --- **$(OPENCV_DIR)x64\vc14\lib\opencv_world331.lib** and

$(OPENCV_DIR)x64\vc14\lib\opencv_world331d.lib

SOIL2: SOIL (Simple OpenGL Image Library) is used to create a cube map texture out of the six images.

Commands used to link this library in visual studio 2017:

- Linker->General->Additional Library Directories--- **$(SolutionDir)\External Libraries\SOIL2\lib**
- Linker->Input->additional dependencies --- **soil2-debug.lib**

2.2. Navigation

The user can move around the scene as though "walking" through it using keyboard and mouse. The up and down arrows move the camera forward and backward in the direction it is facing. The left and right arrows pan the view sideways. The U and D keys are used to move up and down. The X key can be used to change the angle of view. The mouse is used for trackball navigation. The mouse is used to look around and keyboard to navigate across the terrain.

Key Stroke	Description
"↑" **(up arrow)**	Move the camera forward
"↓" **(down arrow)**	Move the camera backward
"→" **(right arrow)**	Pan the view to the right
"←" **(left arrow)**	Pan the view to the left
U	Move the camera up
D	Move the camera down
X	Change the angle of view

These keys can be held down to get a smooth navigation. When just pressed and released they make a slight change in the camera position.

2.3. Elements Included

The modelling of the terrain started off as four simple flat squares one in each coordinate section. The blend map texture shown in the figure is used to implement multi-texturing on the squares. Based on different colors in the blend map, different textures are applied to the terrain.

Black: Grass texture

Green: Mud texture

Blue, Red: Snow texture

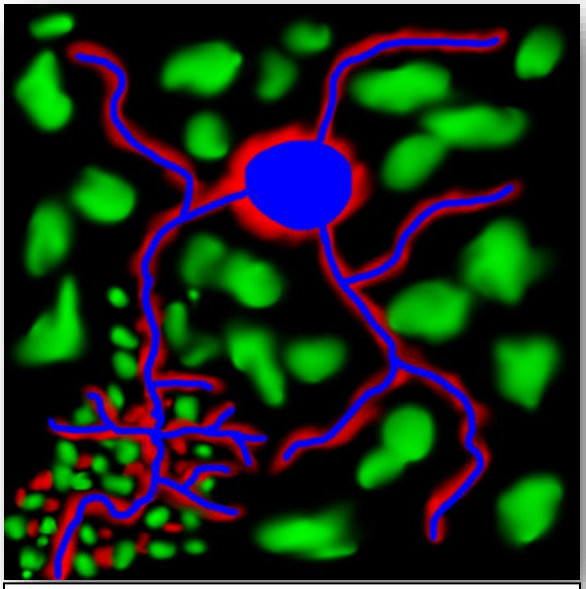


Figure: Blend Map for Multi-texturing

The height map shown in the figure is used to make the mountains and valleys. OpenCV image processing and image transformation functions are used to read this image as a series of float values. A function calculates the height values for each vertex based on these values. These values are used to calculate the heights and normal at all the vertices.

Figure: Height Map for Mountains

The tree object is created in Blender and imported here. Then, its vertices, normal, indices and textures are then loaded through OpenGL script. The UV map shown in the figure is created using unwrapping in Blender and used as a texture for trees.

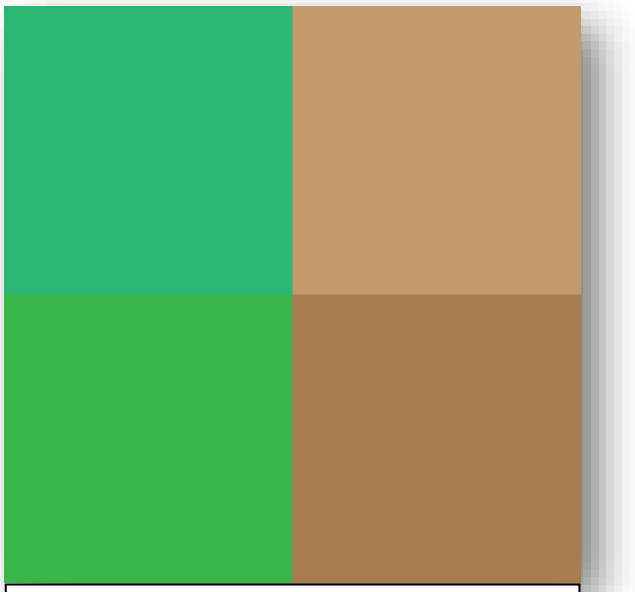


Figure: UV Map for Trees

The displacement/noise map shown in the figure, along with time component, is used to create ripples in the river. This functionality is implemented in the object's fragment shader.

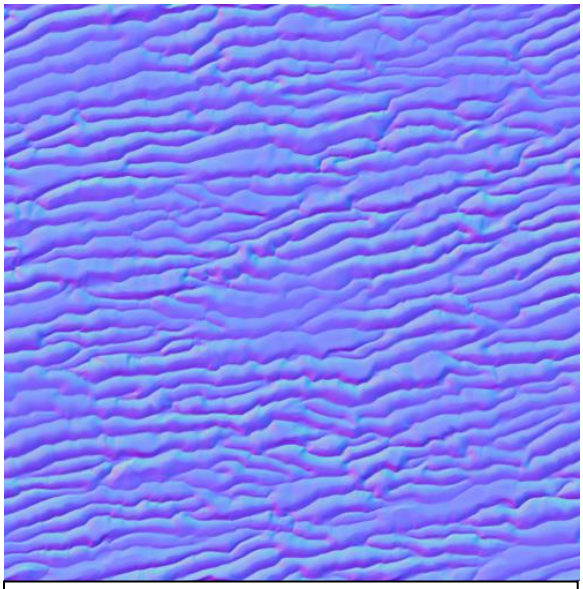


Figure: Displacement Map for River

The Cube map images are shown the figure below. These are used to create the skybox. SOIL library is used to create a cube map out of these images. Cube sampler is used in the fragment shader to apply these textures.

Figure: back	Figure: front	Figure: left

Figure: top	Figure: right	Figure: bottom

Sky's reflection is generated on the surface of the river through passing this cube map to its fragment shader as well. The reflection vector is calculated in the shader and the texture is fetched for this value. This effect is added to the already calculated texture. This resulted in a realistic look where the reflection is blended with the river's ripples and light reflections.

2.4. Positioning of individual trees

A function based on Barycentric Interpolation was created to find the position of the trees on the y axis. Barycentric coordinates can be used to express the value of any point located on the triangle with three scalar values at the corners.

Based on the randomly generated x and z coordinates the primitive is found in which the tree needs to be positioned. The height of this primitive is calculated using our function. The heights at the 3 corners of the triangle is known and can be used to calculate the interpolated height value for random position of the tree within the triangle.

3. Result

A scenic terrain with snowy mountains, trees, a rippling and shining water pond with bright yet cloudy surrounding has been achieved.

Following goals are met:

1) Implementing as many concepts taught in the classroom lectures as possible.
2) Minimal visible visual artifacts hindering the end use experience.
3) Keeping the architecture simple and extensible for future work.

	Modeling	**Light**	**Appearance (Material)**	**Navigation / interaction**	**Keyframe Animation**
1	Single primitive or loaded objects ✔	*0* ✔ *you need a single light*	Basic specular + diffuse reflect code ✔	Trackball navigation ✔	Single linear transformation
2	Multiple primitives ✔	Combined Spot + direct + point light	Texturing	Navigation in 3D ✔	Animation with rotation and translation
3	Complex object surface	Combined light on surface	Multi-Texturing ✔	Navigation with keyboard + mouse ✔	Triggered (keyboard,etc.) animations
4	Hierarchical model	Combined **colored** light on surface	Environment Mapping ✔	Navigation along a surface	Animation with collision detection
5	Hierarchical model, h>2	Triggered light sources	Bump or Displacement Mapping ✔	Navigation with multiple cameras	Multiple animation paths per object

4. Key areas for future enhancements

As a next step it is intended to build the mountains of "Pandora" with a waterfall from the World of "Avatar". This can be an extension to the currently fetched result.

More possible improvements:

1) Navigation along the surface could be implemented in an improved version of the project. To traverse along the varying height, the function used to place the trees at different terrain heights could be used.
2) The skybox could be made seamless. The black borders at the edges could be removed.
3) The skybox could have a slow movement along an axis to create a visual effect of clouds floating in the sky.
4) The concept of refraction could also be applied to see the bottom of the river through clear water.
5) A Sun could be created as an emissive spherical object to move around the scene to create the visual effect of sunset and sunrise. Simultaneously, day and night effect could be created with fading/enhancing lights on surface and using a night sky cube map as well.
6) More realistic rendering of the landscape could be achieved by adding fog simulation and giving a sense of depth in the scene.