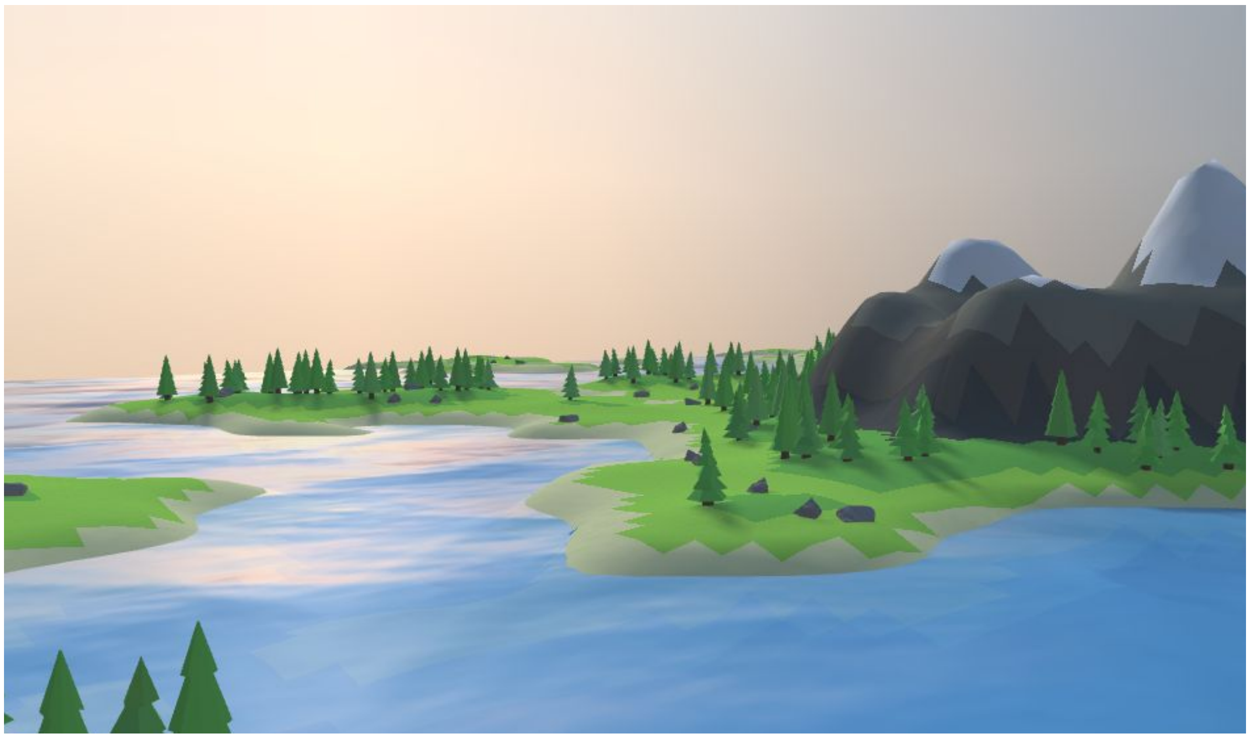

Placeholder image [1]

World-101110011

Implementation outline

COMP 371 - Team # 13

12.02.2017

—

Name(s): Tarik Abou-Saddik,

Alice Barkhouse, Sami Boubaker,

Vsevolod (Seva) Ivanov, Eric Morgan

Overview

The purpose of this application is to explore computer graphics using OpenGL with the C++ programming language. More precisely, we are interested in the creation process of a procedurally generated world. We decided to go with the idea of recreating the nature in a form of a forest. For the world, we will attempt to create tessellated aesthetics to underline the artificial nature of this computer generated world. Depending on the time constraints, this forest will have various “objects” on its terrain like trees, plants, bushes etc.

This world will present an artistic simulation of a forest to be explored by its user. It will be presented as an extendable base structure. World-101110011 will serve as an inspirational environment relying on a visual representation of the tessellation performed by our graphic cards.

The interaction of its user is based on an explorational gameplay of the game, Journey [2].

Goals

1. Create a procedural world with a terrain.
2. Load procedurally generated objects into that world.
3. Handle collisions and allow camera motion.
4. Add shadows, weather, etc.

Specifications

To enhance team collaboration, we work in a private GitHub repository : https://github.com/TarikAbou-Saddik/COMP371. Moreover, this would allow us to handle cross platform compatibility, contributions and issues. Afterwards, either this repository or a teammate fork will be made public on GitHub under a license the majority will agree on.

Each member will work in his classes to allows us modularity and to avoid recurrent merging conflicts. Moreover, this would allow us to keep a good level of transparency by using a simple interface located in the .hpp header C++ files.

The Milestones are not definitive. We will all attempt to implement each part that we have chosen. However, the complexity of the implementation tend to differ from previsions. Therefore, it is probable that we will help each other on complex subjects. The latter cross-contributions will be mentioned in the project README file.

We will create a planning with deadlines to ensure project progress and delivery in time.

Milestones

I. Research and present

Each one of us will research into a subject of interest in the below Milestones and we will present them to the team as well as the way we want to integrate them. We will focus on modularity (e.g. OOP) to create a simple interface that we can directly use without the necessity to merging various parts and increasing the risk of breaking things.

II. Make a modular SkyBox

It will create an illusion of depth in our world beyond the motion limits.

III. Procedurally generate the terrain

We will attempt to create a non planar terrain with height and shape variations. For this purpose, we will use the Perlin Noise. Afterwards, we would have to decide on textures variations depending on the "level from sea", height of the noise function.

IV. Define and find digital assets for non-repeating items

Objects such as trees should be dynamically generated with slightly random sizes and shapes. These object will be put in their own cpp and will extend a base class allowing the programmers to easily integrate them into the terrain. The object classes will also generate a hitbox for collision detection. We can find these assets from opengameart.org [3], or a similar site that allows free use of their models for non-commercial use.

V. Motions

Our world will be viewed through a first person perspective. Though we will essentially be sweeping the camera through our world, it might be necessary to make Camera.cpp the base class so that the view of our world can be set in motion and make a descendant class titled Character. Dividing the two allows basic functionality to be tested with Camera and more motion specific testing to be done with Character.

VI. Collisions

Motion within the world will be free and fluid, but we must also prevent our camera/character from falling through the world and from passing through forest objects, such as trees. Though a substantial amount of work still needs to be done, algorithms such as the sweep-and-prune algorithm [4] are being assessed and data structures, such as kd-trees are being looked into.

VII. Shadows

Objects in the world should project a shadow. Point shadow mapping should be sufficient for our purposes, since we are attempting to recreate the natural feel of a forest.

VIII. Weather (particles)

Rain (with reflection), snow, wind, clouds. Undecided between different biomes or universal weather patterns. Particles such as rain and snow will fall from cloud objects and be deleted when they contact the ground, accelerating appropriately in between. Wind will be represented by air particles being subtly stylized and moving in patterns reflecting air current.

Creating / Generating Assets

If at some point, a little further down the line of the development, we feel that we have to start assembling together our own assets, we will most likely do so using popular applications like Maya or Blender, which we will start looking into very soon. In the meantime, we have also talked about using royalty-free textures or assets [3].

Interaction schemes

Keyboard and mouse input will be used to manipulate the location of the character, as well as the camera angle. We will most likely use the built in GLFW functions to create user input functionality. Such as using the WASD keyboard keys for movement, or the ‘f’ key to change to fullscreen (as an example).

View manipulation

- Window.cpp (for generating our OpenGL context).
- Camera.cpp
- Character.cpp

Virtual world manipulation

- World.cpp

Difficulties

A primary one would be to synchronize successful builds between three operating systems namely : GNU / Linux, Mac OSX and Windows. We'll try to use macros in our code to alleviate some of these problems.

Another difficulty would be the handling of merge conflicts and interdependency conflicts between our milestones.

If the tessellated world becomes too complex a project for us (within our given timeframe), we might turn it into a simpler representation, using more trivial aesthetics, similar to those in *Minecraft*.

Team

Student ID	Name	Parts (tentative)
40004286	Vsevolod (Seva) Ivanov	Project architecture, documentation, Skybox, procedurally generated terrain
27486782	Alice Barkhouse	Shadows, locating and integration of models, interaction shema
26417404	Sami Boubaker	Procedurally generated Objects, documentation
27518722	Tarik Abou-Saddik	Character motions and collision detection, documentation.
26863426	Eric Morgan	Weather and reflections

References

[1] Placeholder cover image, online, https://playcanv.as/p/fSkXv2EP/

[2] Journey game, online, https://www.youtube.com/watch?v=bkL94nKSd2M

[3] Assets, online, http://opengameart.org/

[4] Sweep-and-Prune algorithm implementation, online, http://www.codercorner.com/SAP.pdf