The University of Melbourne
School of Computing and Information Systems
COMP30019 Graphics and Interaction

Project 1, 2020

Set: 20th August

Electronic submission: 4pm, 20th September

Marks: This project counts towards 30% of the marks for this subject

This project should be completed in groups of **three** or **four**.

Aim

The purpose of this project is to get you acquainted with basic Unity development, as well as collaborative programming using Git/Github.

Task

In this project, you will use fractals to automatically generate a 3D landscape, and enable a user to navigate around the landscape in a 'flight simulator' style.

Diamond-square algorithm

You must use the ***Diamond-Square algorithm***, which is a de-facto standard in fractal landscape generation. Your fractal will generate a ***heightmap*** which can then be used to construct the 3D landscape geometry.

There is information on the diamond-square algorithm on Lecture 5, and you will receive support during the tutorials.

Git/Github

An important part of this project is not just the submission itself, but the collaborative process. You will use the version control software Git (via Github) to track changes to the project and allow each group member to easily make remote contributions. **It is important that all group members commit their changes to Github from early on in the process.**

Specifications and marking criteria

A project that satisfies all the criteria listed below will receive 30 marks.

- **Modelling of fractal landscape (7 marks):**

  - You must automatically generate a randomly seeded fractal landscape at each invocation of the program, via a correct implementation of the diamond-square algorithm.
  - You must use Unity's architecture appropriately to generate and render the landscape.
  - There must be no notable problems or artifacts with the polygonal representation.
  - The default parameters of the Diamond-Square algorithm should be set appropriately to model a *realistic* looking landscape.
  - The landscape should be generated in a timely manner (i.e. several seconds maximum on any computer environment).
  - Pressing the 'space' key should result in the generation of a new random terrain according to the same criteria above.

- **Camera motion (5 marks):**

  - Your camera controls should be implemented in a 'flight simulator' style, with the following specifications:
    * Moving the mouse should control the relative pitch and yaw of the camera
    * The 'w' and 's' keys should cause the camera to move forwards and backwards respectively, relative the camera's current orientation
    * The 'a' and 'd' keys should cause the camera to move left and right respectively, relative to the camera's current orientation
  - You must allow the user to move anywhere in the world (including up into the sky), and prohibit the user from ever moving or seeing "underground" or outside the bounds of the landscape.
  - The camera must not become 'stuck' upon nearing or impacting the terrain, i.e. reversing and continuing to move must always be possible.
  - You must utilise perspective projection, and choose a suitable default perspective, so that the landscape is always clearly visible from the first frame of the simulation (irrespective of the initial mouse position). This should be reset every time the terrain is re-generated.

- **Surface properties (10 marks):**

  - The colour of the terrain must correspond in a sensible way with the height of the terrain at any particular point (for example rocky outcrops or snow on top of mountains and grass or soil in valleys).
  - Realistic lighting must be present based on the Phong illumination model (diffuse, specular and ambient components). You should use a custom Cg/HLSL shader for this.
  - The direction of the lighting must change with time, to simulate the effect of a sun rising and setting.
  - The sun itself must also be rendered, in order to help verify the correctness of your lighting implementation. You may use any simple geometric shape such as a pyramid, cube or sphere to represent the sun.
  - Water sections must be present. You may use a plane passing through the terrain to achieve this. A custom Cg/HLSL vertex shader should be used to create realistic waves via displacement of vertices within the plane. Ensure the plane has enough vertices for the effect to be sufficiently detailed.
  - A constant and reasonable frame refresh rate must be maintained during program execution (i.e., 30 frames per second or more)

- **Project organization and documentation (8 marks):**

  - Github must be correctly used to track changes to the assignment code. The project should be easily downloadable via the given commit ID (see below for details).
  - Each group member should make at least one commit representing a personal contribution to the project.
  - A README.md file should be present in the root directory of the project for the given commit ID. This should concisely describe your implementation, including:
    * A description of your Diamond-Square algorithm implementation.
    * How the geometry of the terrain (including normals) is generated and represented within Unity.
    * The transformations used to facilitate the camera motion.
    * A justification of how you parameterised the Phong illumination model to achieve realistic surface properties for both the terrain and water.
    * How you implemented a vertex shader to achieve a realistic wave effect.
  - The README.md should be well-written and formatted appropriately. It should be easily readable within Github.

Electronic submission

Your submission must open and run in Unity 2019.4.x without modification via a direct download from Github. It is highly recommended that you test this regularly while working on the project to ensure that your work is being committed correctly. **One member** of your group must submit a text file to the LMS (Canvas) by the due date which includes the repository URL, the commit ID of your final submission, and the names of all group members. All group members should commit changes to Github from the very start of the project. You will not be marked on the content of the commits before the final submission commit, but these may assist in investigating disputes over unequal contributions to the project.

Plagiarism and attribution

If your code contains code from other sources, in particular from other web sites, you have to clearly indicate this both within the code (as a comment) and README.md. Remember that copying code from external sources without attribution is considered academic misconduct. Additionally, we expect that a majority of your submission is your own regardless, in particular your Diamond-Square implementation. You may copy code from the labs, but full credit will not be awarded where there is an over-reliance on external code, even if it is attributed. We will be checking for similarity between submissions and with code available over the Internet.

Delays

Make sure you deliver your work on time using Canvas. Overdue delivery will result in a reduction of 10% of the marks (3 point) for each day of delay.