**Investigating a real-time Hydraulic Erosion Simulation to be used for terrain generation in games**

1 Introduction

This portfolio project will contain an investigation into real-time hydraulic erosion when creating realistic terrain in video games and a program that will use techniques found in the investigation to create a real-time hydraulic erosion simulation. Terrain is one of the most important aspects of emulating a realistic virtual experience whether it’s in a computer game that requires a large-scale terrain for an open world experience, a movie that requires a fantasy style location that might be too dangerous to film in or a training simulation that requires the most realistic environment to help train people.

1.1 Aims and Goals

What needs to be achieved

* Aims – quite broad
* Objectives – very specific

2 Literature Review

General tips:

* Use diagrams, but be aware that for important stuff for your project you really need to explain in your own words

Extra sections here:

* Real world stuff
  + Include diagrams, pictures etc (all fully referenced)
* How have scientists modelled this?

2.1 Terrain Generation

Could have an intro bit here to introduce your subheadings

Generating realistic terrain for games is a long and time-consuming process. To solve this problem developers have found quicker and more effective ways for generating realistic terrain. When talking about terrain generation the first step is how that data is represented. Terrain data can be represented with two main models. These being a volumetric model and an elevation model.

Subheading: ways to represent terrain

The elevation model can be described with either an elevation function or a discrete heightfield. The most commonly used method is the discrete heightfield. The discrete heightfield uses a two-dimensional grid to represent the altitude of each position. This means that it is unable to recreate suspended materials like overhangs, caves, and arches. But it is a lot less data heavy which allows a simulation to be ran a lot faster with more data to work with. The elevation function uses a formula that can generate a point of altitude of any point in terrain. This method is mostly used when procedurally generating terrain. The most used elevation functions are Perlin noise or Simplex noise which are procedural generation methods that allow for infinite terrain generation without any user input needed.

The volumetric model is similar to the elevation model but instead of using two-dimensional space it uses voxels which allow for a three-dimensional space where each cell represents a material at a particular position. This allows for the use of features like overhangs, caves, and arches. Volumetric models are very data heavy which makes them slower to run. There are ways to optimize the data structure by using compression techniques like Sparse Voxel Octrees to reduce the memory cost. This is done using an octree. An octree is where a three-dimensional space gets recursively divided into subspaces of children nodes until each voxel only contains one point or multiple point of similar data. This technique can be used in open world games which require real time interaction.

Subheading: ways to generate terrain

* Make a bullet point list of the main topics that go into this section, e.g.:
* Data sets
  + Geographical
  + artist generated data
* Procedurally generated data
  + Fractal
  + Noise
  + etc

The next step is the talk about how the terrain data can be acquired. This can be done user inputting premade data or by a program generating its own data also known as procedurally generated data. What we are going to talk about is different method for procedurally generated data. These methods include, subdivision based and fractal noise methods such as Perlin noise. The subdivision method takes in a piece of data like a plane or a cube and iteratively refines the shape by subdividing it into smaller and smaller pieces. (MIGHT NEED TO GO INTO MORE DETAIL ABOUT SUBDIVISION). One of the most used procedural generation methods is the noise method. Noise is essentially a random number generator for computer graphics. By adding different scales and amplitudes to the noise it is possible the create procedurally generated terrain that can be endlessly generated allowing for landscapes to be easily generated at any size a user might need. (COULD ADD MORE ON THE CALCULATIONS MAYBE)

2.2 Water/Fluid Simulation

(INTRO TO WATER NEEDED). 🡨 make sure to set the context, fluid sim is a vast topic. Write this intro once youv’e done the rest

There are two main methods for simulating water. These are grid based and particle based. Both grid-based and particle-based methods have their own advantages and disadvantages that make them better suited for different scenarios.

What are your main subheadings:

* Grid based
  + Full volume vs shallow water (height maps)
  + Identify the main topics, then think about how to organise them
* Particle based

The most efficient way for simulating water with a grid based method would be by using a height field which uses a two-dimensional grid with each cell holding a height rather than using a three-dimensional grid with multiple data point to consider for each simulation step. Some methods that use height fields are the shallow water equations which only simulate a thin layer on the surface of the water and the pipe model which uses pipes that connect each cell. (TALK ABOUT SHALLOW WATER). (TALK ABOUT PIPE MODEL). (TALK ABOUT METHODS THAT USE 3D GRIDS).

The most used particle method is the Smooth Particle Hydrodynamics which is based on the Navier Stokes Equations. This method tracks each particles location instead of using a fixed grid. Each particle represents a quantity and mass. The quantity is calculated by a weighted sum of the neighbouring particles.

(TALK ABOUT METHOD THAT COMBINES SOME PARTICLE AND GRID BASED)

2.3 Erosion Algorithms

O

**Heading: games implemented**

* Talk about specific games and how they implement their methods (where known)

3 Design

o