IS71021A Mathematics and Graphics for Computer Games 1

Coursework 2

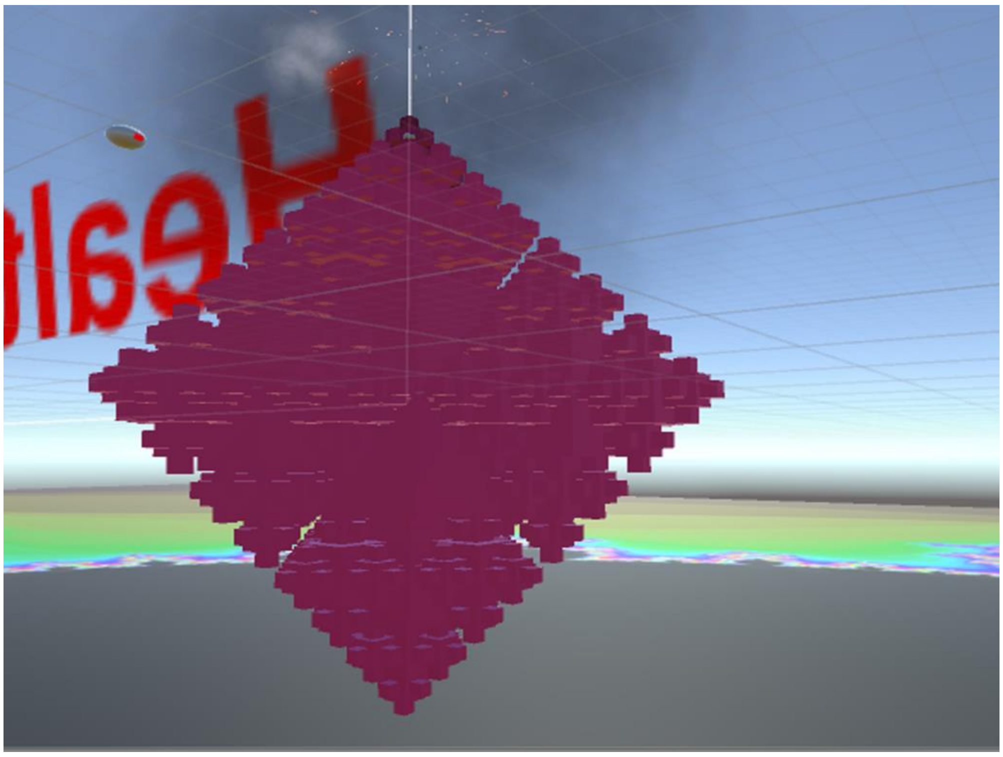
Stefanos Katsaros

Youtube link: https://www.youtube.com/watch?v=uXTmBWFPYRA

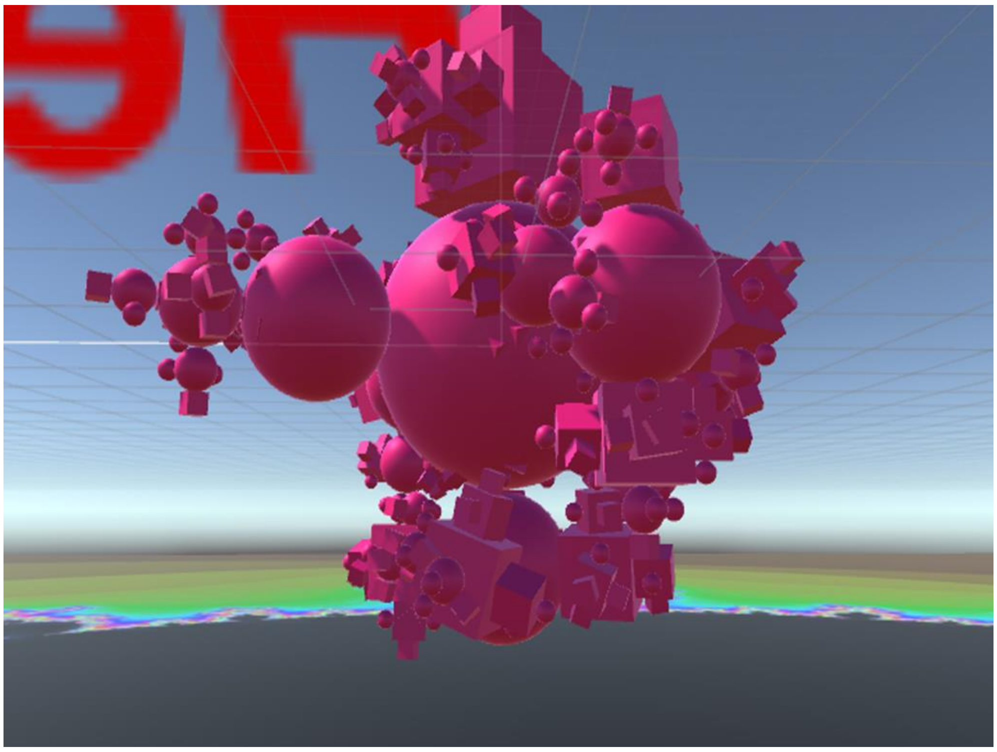
The project I have conducted is based on the study of fractals and the Mandelbrot set. It is composed of 2 parts, the unity part where the game takes place and live fractal generation is demonstrated, and a java program which creates a Mandelbrot image which is later used as a texture for a material in the unity game.

The game: The game is a first person shooter, where the player has to defend his fractal generated structure from the enemies that spawn constantly and try to destroy it. In the meantime he must be carefully to not fall from his elevated structure, because he will end up dead in the Mandelbrot Set. The game is as simple as it sounds, with the player having a score and a health system and wielding a weapon that generates physic structured large bullets. The enemies roam around the xyz axis shooting and destroying the entire structure while trying to take out the player. There are two options for level difficulty which are easy mode and hard mode. In the following section I will not be mentioning the gameplay, the enemy logic and such, but will instead be discussing the fractal generated structure and later on the production of the Mandelbrot Set.

The Fractal class I have created refers to a repeating process that generates an array of objects, on all six sides of a generated root object, and repeats the same process for all of those children, and all of the children's children and so on, until I specify otherwise. Additional parameters are added to each child that is generated, such as size and rotation, and impressive structures appear. There are two modes, easy mode and hard mode, and each one refers to the parameterization of the fractal generation. In easy mode, everything is much simpler. There is a set number of children generating from a root object which is a large cube, and the objects simply shrink and shrink to a maximum depth of 4 (meaning that children will be generated up to 4 times (the structure must have a depth size or the process will go on endlessly until the PC is unable to cope)).



On the other hand hard mode shows the power of using fractals to generate an impressive structure. Messing with each child's parameter, I ended up adding a randomness to the spawn, a constant rotation to every game object and two different primitive objects (cube and sphere) that randomly take the place of the root and the next child object. The gameplay thus becomes much harder.



For the Mandelbrot Set image, which is used as the ground that the player falls into accidentally or when his structure is destroyed, I used java to generate an image which is then stored as a texture in a material. The Mandelbrot Set is the set of complex numbers c for which the function f(z) = z^2 + c does not diverge when iterated. More specifically it can be found found by picking a starting point c and iterating the formula $z_{k+1} = z^2 + c$. This gives a sequence of numbers that may either stay in bounds or goes out and far from the starting point. The complex number c belongs to the Mandelbrot set if the sequence stays within a radius of 2 from the origin. To plot the Mandelbrot set I had to match each pixel on the screen to a complex number, see if it belongs to the set by iterating the formula and later colour differently each iteration if it does belong, and if it doesn't colour it black. Furthermore since java does not support complex numbers we represent the complex number z = x +iy as the pair of real numbers (x,y). Therefore, the logic of the code is centering the image to be mapped as (0,0) and so given a pixel we subtract half the image height from the vertical coordinate system and half the width form the horizontal coordinate system. Then since the Mandelbrot set belongs to a set radius of 2, the entire width of the image should have a length of 4. Furthermore we implement a set number of iterations so that the program won't loop endlessly and we set a resolution for the image that will support further detail. The resulting image is the following and I have used it on the mandelMat material in unity.