Will Richards- *IMD 3002 - Prof. Chris Joslin*

Term Project - Part A (outline)

Objectives

I fairly recently finished The Black Swan by Nicholas Nassim Taleb and have been inspired to create a Mandlebrot set representation in Maya. There are various levels of complexity in achieving an effect like this. I will do my best to complete all of them but there are of course **great lengths a project like this can be taken to.** time permitting, I'd like to:

0. *Make extrude work on complex objects. (item #0 as this would not be considered to be in line with project definition but still complicated to achieve)*
1. *Create a "mock" Mandlebrot generation on objects. Something that approximates a Mandlebrot set to a recognizable degree to ensure extrusion in either direction is possible*
2. *An "art" effect on the extruded surfaces. likely a RG gradient to stretch from either side to better fulfill the 3D art theme for this year*
3. *Begin experimentation with LoD in an "infinite" zoom / "infinite" Mandlebrot set generation based on params on Maya GUI.*

To recap, the designed script should allow the user to embed / extrude a Mandelbrot fractal into the face(s) of an object and optionally create an effect on the surface that has been affected. The mandelbrot fractal will scale to a certain (not actually infinite) point depending on user's input in the window. It is difficult to quantify to "utility" of this application given the nature of 3D art, however I believe the repetitive nature of calculating any level of depth in a fractal lends itself well to Maya's API in scripting for 3D modelling. I also think this will be an interesting area to explore given the many different ways a fractal can be loaded into software.

Requirements / Specifications

To better outline how I will adhere to the aforementioned objectives, I am providing a more specific set of rules to follow in design this script:

- *Extrusion/Embedding in a mesh will be toggleable in the Gui along with an "onion-sketch layer" silhouette that displays how the effect will look.*
- *For complexity reasons. The user will only be able to extrude/embed along the face's normal. The z-value for this will be available to change from the GUI*
- *Extrusion should work on complex objects without destroying the mesh or creating non-manifold geometry. May need to clamp extrusions similar to how Blender's bevel tool works.*
- *Mandlebrot approximation needs to appear as such. The "fractal" created should appear like the real thing, minus the ability to scale infinitely, at least at first. The lowest LoD should be able to run no problem on low-end computers (or least computers capable of running Maya Haha).*
- *Given the specified LoD, the user should be able to scale to a point that appears correct on their mesh. Essentially LoD should be high enough fidelity that the user can utilize the script on a low-res mesh.*
- *The gradient effect will appear like a standard RG uv map and the two colours will customizable from the GUI.*
- *While I have not experimented much with camera clipping I'd like to try implementing dynamic LoD loading as the user scrolls further in to view the mesh. As a formal requirement for the project however, I'd like to ensure that LoD is Statically decided upon by the user within the GUI*

Coding Layout

`getBounds()` *- get bounds of selected faces Area. Ensure the faces are continuous so that the fractal continues throughout the selected mesh and that the faces share the same normal. (the same normal ensures that there are no non-manifold shapes created through collisions), then return the area to be rendered upon.*

`validSelection()` - Helper function to assist getBounds in ensuring the faces are uniform and with parallel normals. The previous function getBounds() will return the faces that will be rendered upon after being checked by validSelection()

`renderPath()` - This function will draw the Mandlebrot fractal outline along valid faces after being checked. This render path is what will be referenced to create the "onion-sketch" overlay and the final render. renderPath() will take inputs like width, height, LoD, and other modifiers

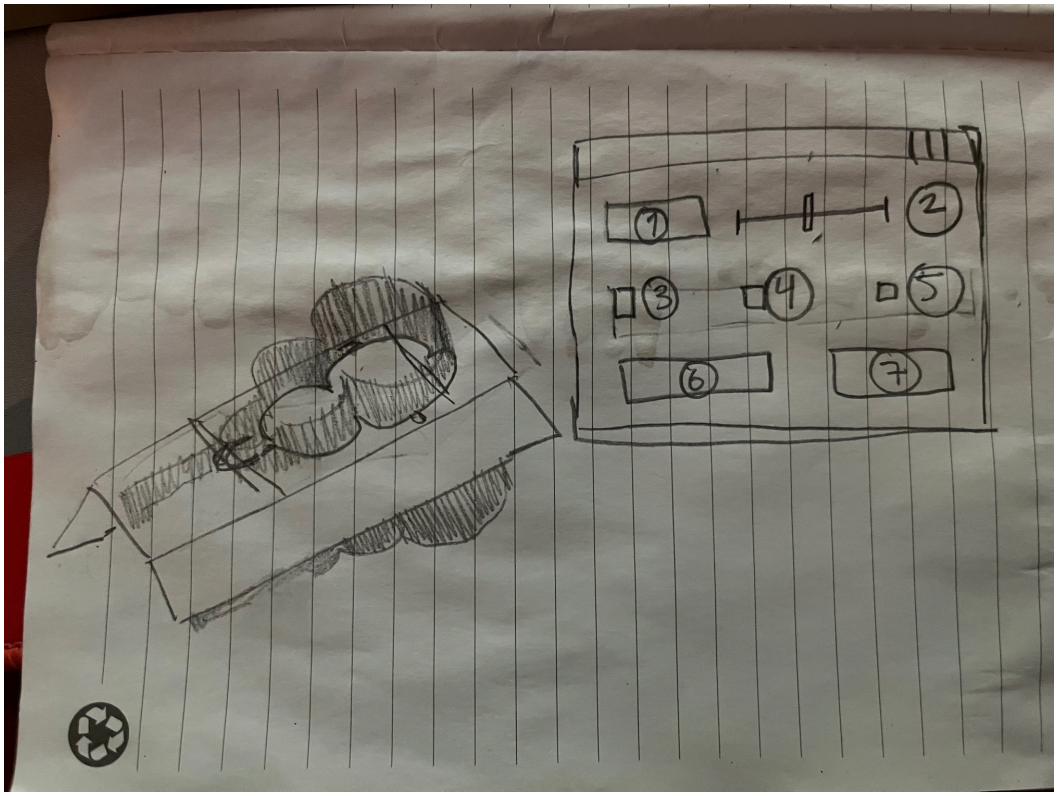
`createOnion()` - This function will take in the vertices specified by renderPath() and create a translucent overlay to visualize the effect. The overlay will change with the z-value entered, and will not account for proper geometry checking as it is temporary

`extrudePath()` - Take an input of renderPath() vertices, subdivide the face to support the added vectors and extrude by the input z-value. Varying levels of cleanup may have to be done but the user depending on how far I get in this project. This function returns the faces after extrusion to be used for color changes

`embedPath()` - Take an input of renderPath() vertices, subdivide the face to support the added vectors and embed by the input z-value. Varying levels of cleanup may have to be done but the user depending on how far I get in this project. This function returns the faces after embedding to be used for color changes

`colorFractal` - Applies a customizable lambert material to color extrusions/embeddings based on a Mandlebrot-esque pattern

Design Sketches



img GUI has ordered labels, explained below:

1. *Check Selection - button to verify that the selected faces are valid before user attempts to perform action*
2. *Z-Value Slider - This slider will affect the "onion-sketch" overlay when user presses the "Check Selection" button. This is the value used for the extrusion and embedding*
3. *Color? - button will toggle wether the user wants to color the shape after extrusion/embedding. This makes the color process more robust code-wise - off by default*

4. *Col 1 - color picker for gradient - grey by default*
5. *Col 2 - color picker for gradient - grey by default*
6. *Extrude - button uses the params provided to extrude the fractal (up in this diagram)*
7. *Embed - button uses the params provided to extrude the fractal (down in this diagram)*

I may just combine extrude and embed into one button and allow negative z-value input

The onion sketch makes things easier to comprehend for the user while posing in interesting challenge to implement, which will be fun. My hope is that it will be easy fast on the user side as it won't require geometry checking, though I will see after spending more time in Maya. This input scheme seems fairly intuitive and logical given the code needed to implement, though of course subject to change if I find a better way to lay things out.