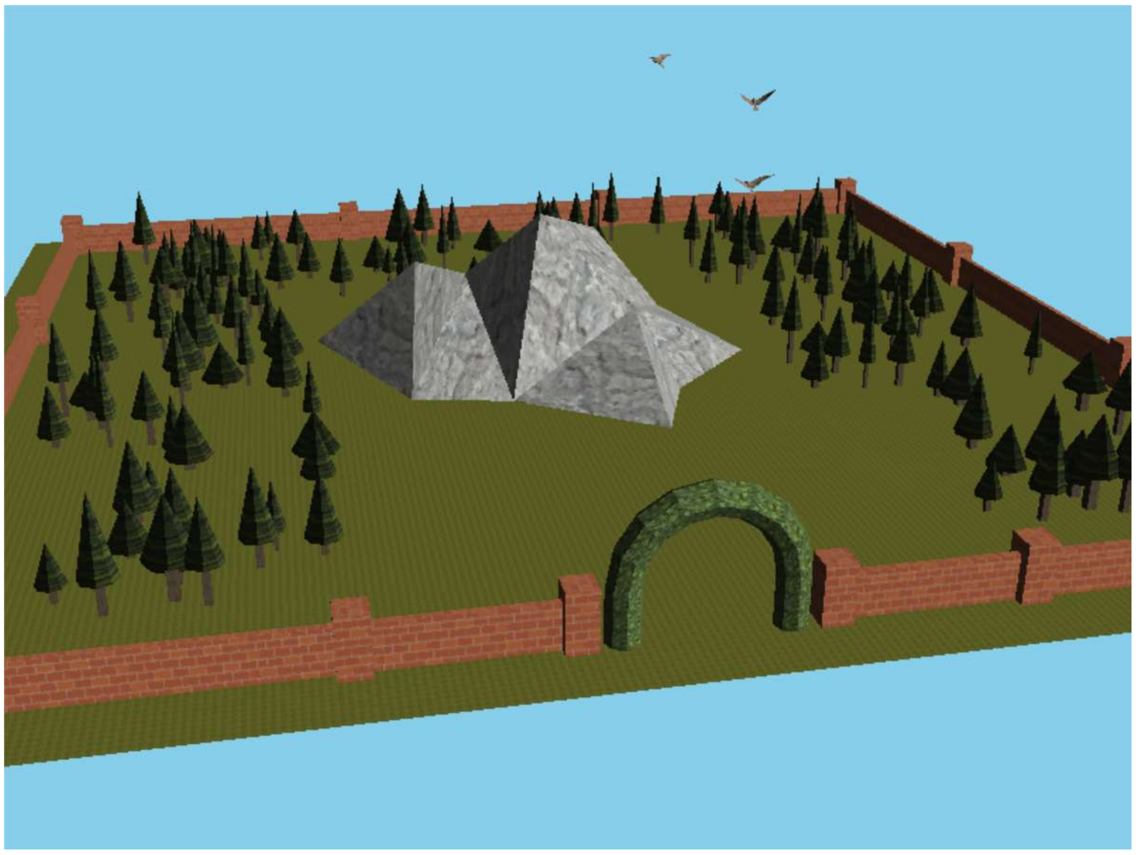
**Author:** Tayte Waterman
**Assignment:** Final Project Report
**Course:** CS547 – Computer Graphics
**Date:** Nov 29th, 2023

## Overview

The following report describes the CS547 final project 3D rendered scene implemented in OpenGL and C++. The scene shows a "Forest" park, containing multiple objects implemented via a collection of techniques covered within the course. This report describes the elements of the scene, its basic controls, and a general overview of techniques captured per object.

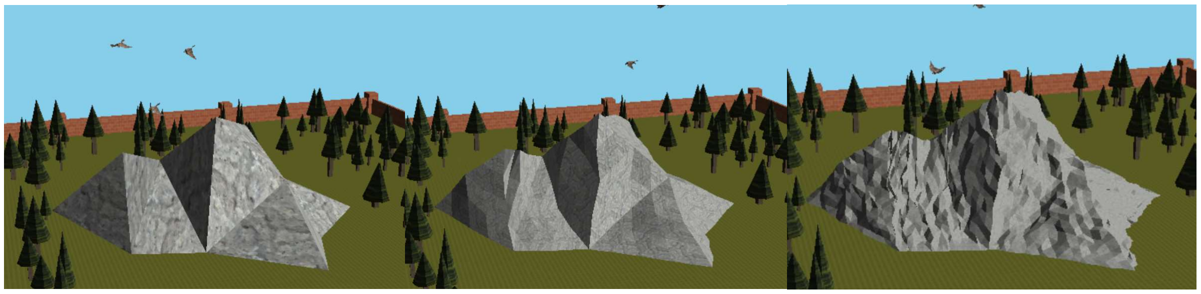
## Scene Summary

The implemented scene captures a "Forest" park, enclosed within a brick wall, and composed of a central mountain and surrounding forest of trees. The scene allows for user input controls to manipulate the user view, model attributes, and lighting.

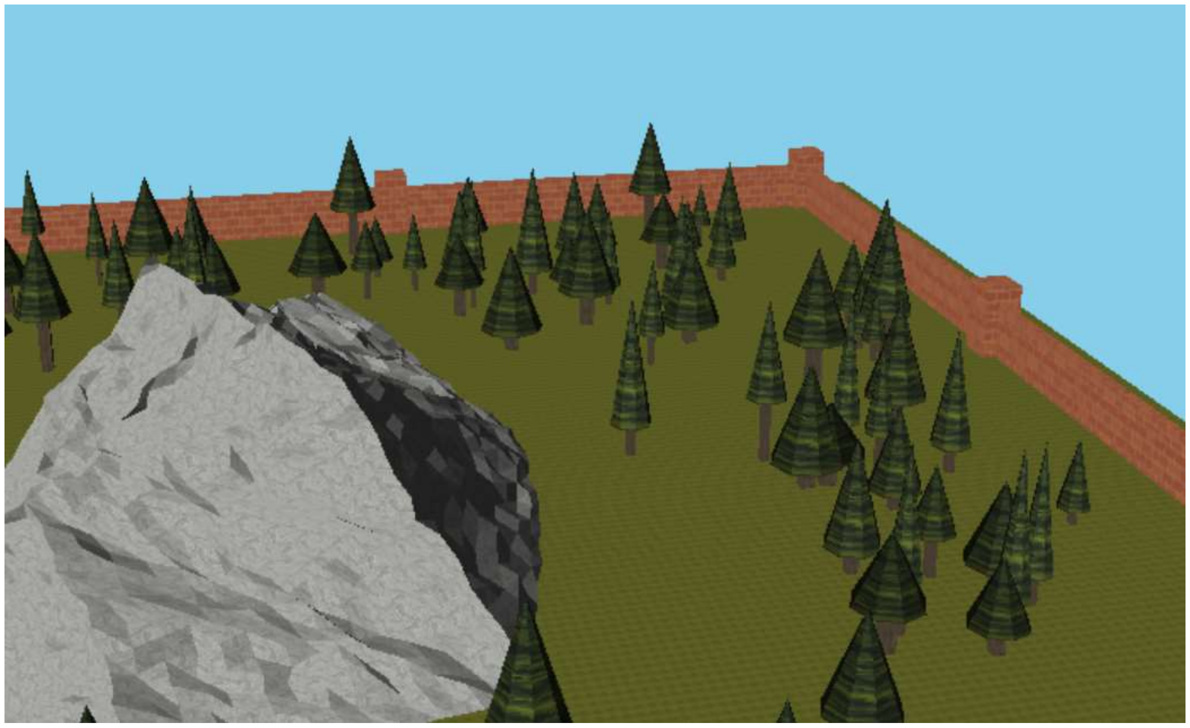


*Forest Park Scene*

The central focus of the scene is a mountain, implemented out of basic triangular geometry, but which can be increased in detail by increasing the subdivision of the faces through multiple levels.


*Increasing Mountain Subdivision*

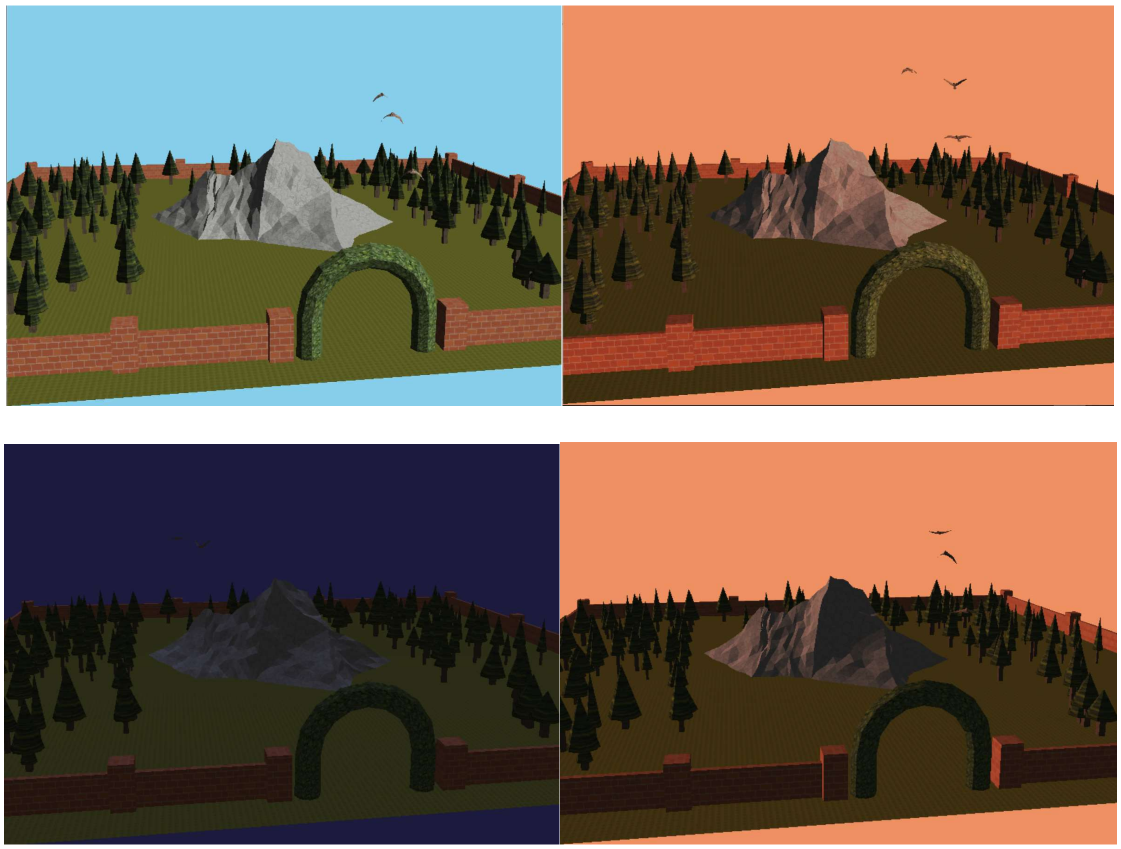
The surrounding forest is composed of parameterizable tree objects, randomly generated with different attributes to make each instance unique.


*Trees of Randomly Varying Attributes*

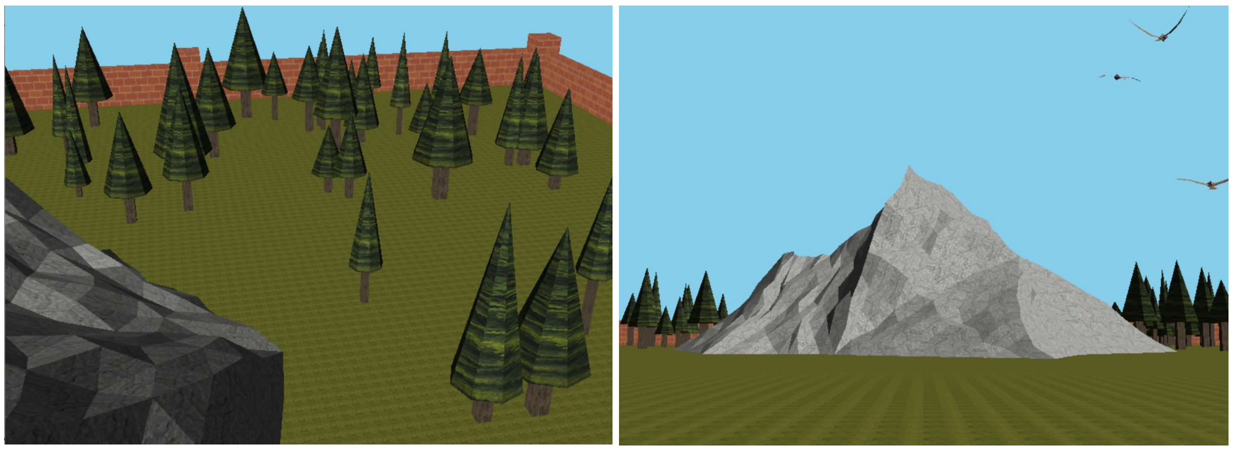
Above the mountain fly three animated birds which follow a pre-defined spline path around the mountain. The user may select to change the camera to the viewpoint of the leading bird.

*Animated Bird Fly Around the Mountain*

The scene can change the time of day, altering the direction of light, color, and default background color to switch between day, afternoon, night, and morning.


*Changing Lighting Reflecting Time of Day*

Last, the user may change their view perspective from the default view (rotation and pan) included from the project skeleton code, to a perspective from the lead flying birds perspective, and a low view looking up the mountain.


*Changing Camera View: Birds-Eye (left) and Low-View (right)*

**Controls**

| | |
|---|---|
| LEFT_MOUSE | Hold and drag (default view only) to rotate camera view |
| RIGHT_MOUSE | Hold and drag (default view only) to pan camera view |
| TAB | Change view: Default, birds-eye, low view |
| UP_ARROW | Increase mountain subdivision |
| DOWN_ARROW | Decrease mountain subdivision |
| RIGHT_ARROW | Advance time of day: Morning, day, afternoon, night |
| LEFT_ARROW | Reverse time of day: Night, afternoon, day, morning |

**Description of Objects and Contained Techniques**

The following section describes the contained objects and a high-level description of their implementation, highlighting the techniques used per the Final Project rubric.

*WorldWindow Modifications*

The WorldWindow class has been modified from the project skeleton code to contain the scene objects described below. In addition to adding the new objects, the code has also been modified to support additional user inputs (key presses), modify the scene lighting (time of day lighting), as well as change the view for the user.

The lighting changes are handled via a predefined set of arrays defining lighting color direction, and background color, which are accessed at advancing/reducing indices into the array.

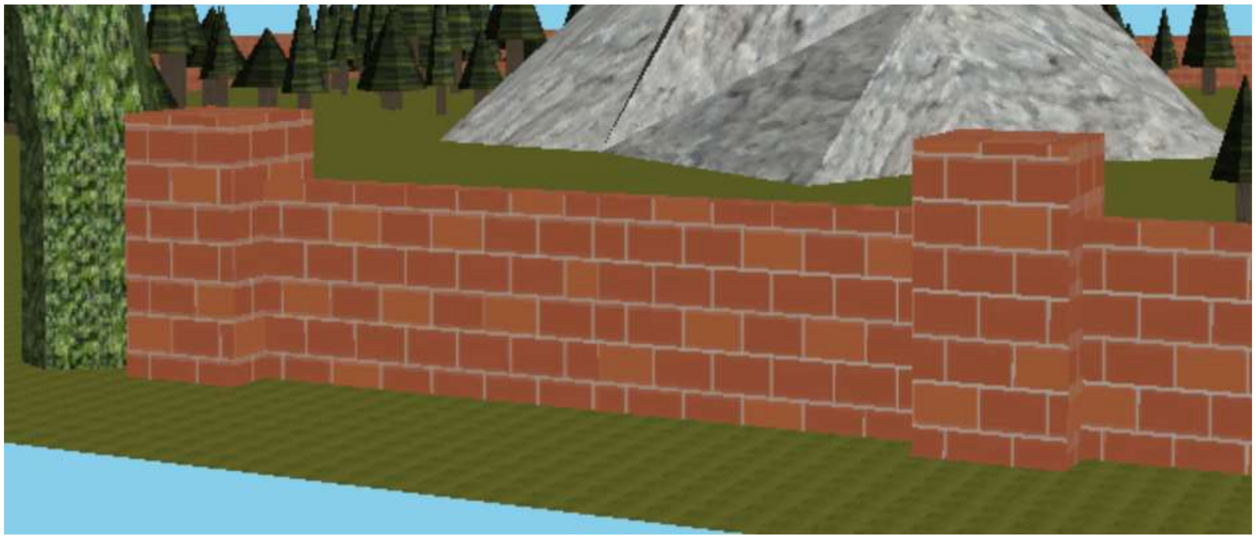
The view change is handled via a switch configuring the view parameters to the initial project skeleton default, a fixed view (hard-coded values), and the birds-eye view. For the birds-eye view, the view parameters are set via a pointer variable accessing the Track class internal bird position and orientation.

The model demonstrates the following techniques from the project rubric:

- Change the Navigation System
- Aesthetics (Lighting changes)

*Wall / Block*

The Wall model is handled via a Wall class containing and composed of Block class objects. The Block class objects themselves are simple cubes with a brick-texture applied to them. The Wall class, instead of defining individual vertices and polygons, instead defines the positional location and scale of multiple block instances to construct the park wall.

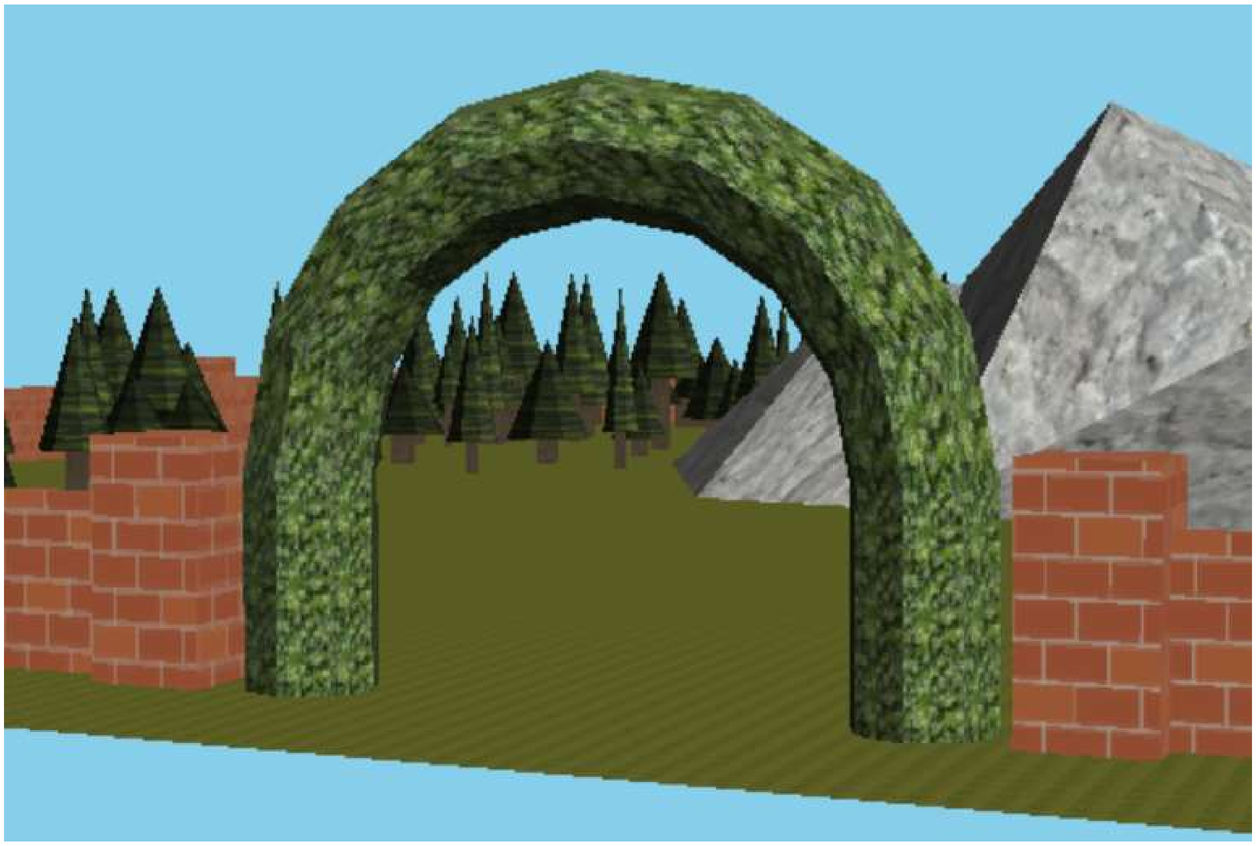

*Section of Wall Composed of Block Objects*

The models together demonstrate the following techniques from the project rubric:

- Texture Mapping

*Arch*

The Arch model is managed by the single Arch class which internally describes all vertices and polygons to construct it. The model is constructed as a half-torus, constructed of a double sweep pattern of quadrilateral polygons constructed off the circle extruded along the “path” of the arch. The arch itself is applied with a “jasmine” texture to make it appear as a hedge.


*Arch Object, As A Circular (Swept) Extrusion Across The Arch Path*

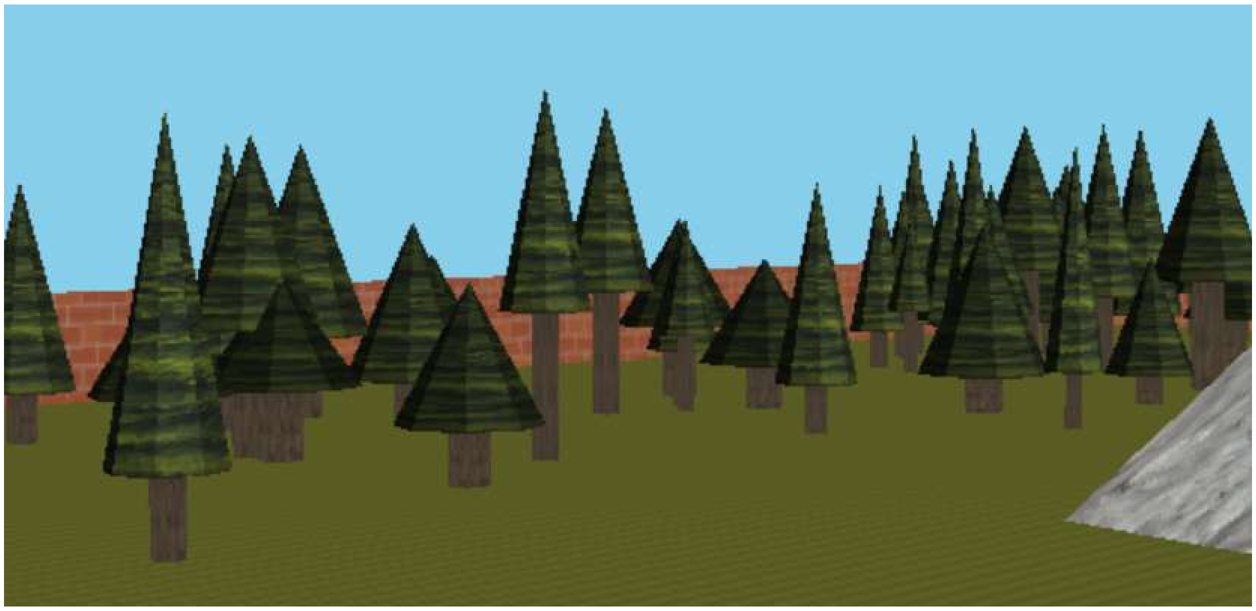
The model demonstrates the following techniques from the project rubric:

- Texture Mapping
- Sweep Objects

*Forest / Tree*

The Forest object is handled via a Forest class containing and composed of Tree class objects. The tree objects are constructed of simple geometry of a swept cone (the top) and a square prism (the trunk). Each tree has parameterizable attributes describing its structure (beyond simple scaling or position), which vary the ratio of the tree's limb width, its total height, and the ratio of the tree foliage to the tree trunk height. Using these attributes, the Forest class constructs a forest of randomly generated and positioned trees within a provided region, and number of trees. Each tree's attributes are randomly varied to make the forest appear to be constructed of unique trees.

To generate and render the forest in a reasonable amount of time, the tree class is defined to construct the tree components (top and truck) once across all classes as two display lists. At each tree instantiation, the same component display lists are called and transformed (per tree attributes) to prevent over calculating of the tree geometry.


*Forest Section Composed of Randomly Generated Trees With Differing Attributes*

The models together demonstrate the following techniques from the project rubric:

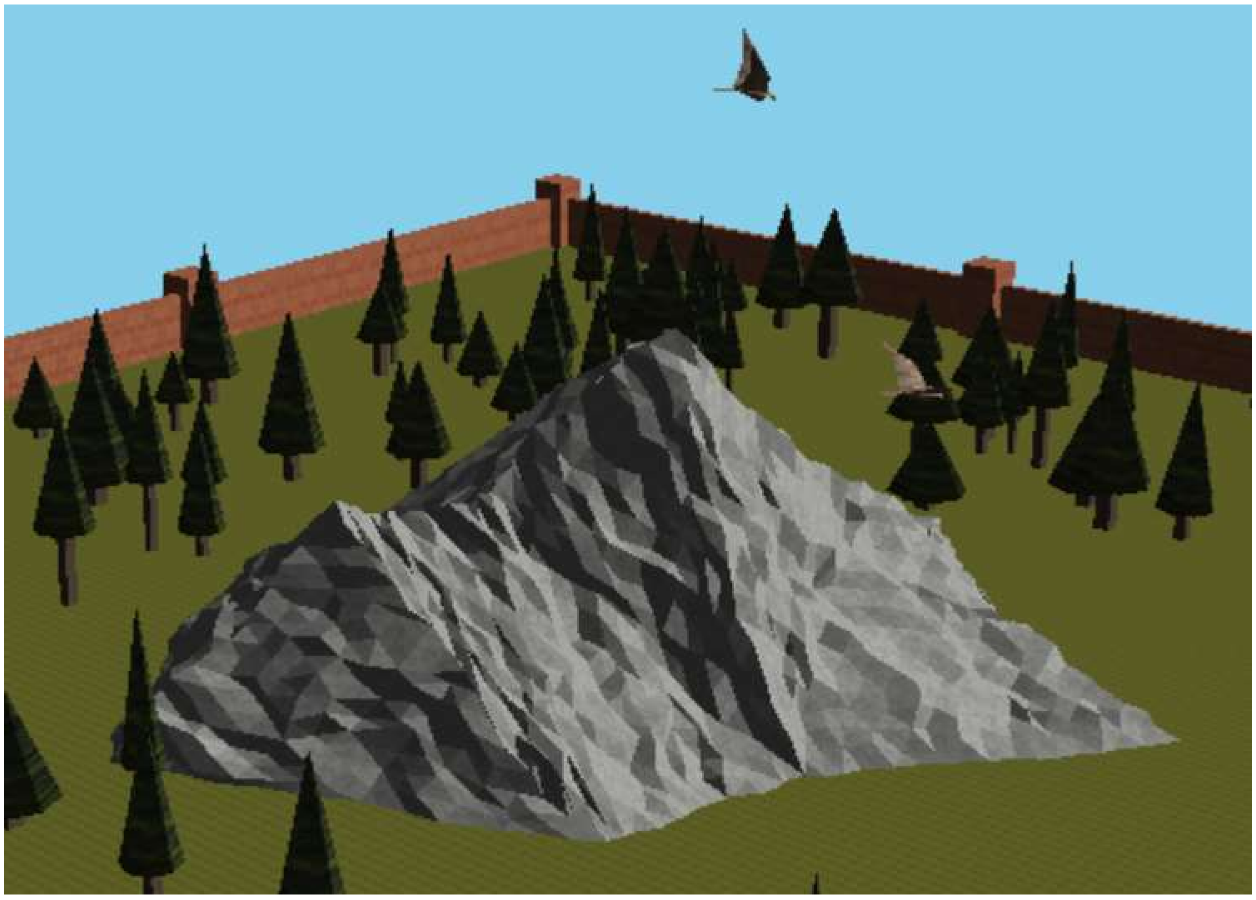
- Texture Mapping
- Swept Objects
- Parametric Instancing

*Mountain*

The Mountain object is handled via a Mountain class. The mountain class internally defines a “base” mountain geometry out of a small collection of vertices and triangular polygons. The “base” mountain does not appear very detailed, but the detail level can be increased by subdivision. At instantiation, the class pre-computes multiple meshes at multiple subdivision levels “seeded” by the base model. Although the base model is hard-coded into the class, the subdivisions are calculated from the “base” model.

At each subdivision, each triangular polygon edge is subdivided in two such that a new vertex is placed at the midpoint of each original vertex. Each new vertex is then randomly varied by a limited value, and 4 new triangular polygons are generated between the 3 new and 3 old vertices. At each subdivision, the magnitude that each new vertex is allowed to vary decreases to prevent the variation from overwhelming the surface. In this manner, the mountain procedurally produces a “rocky” shape.

The subdivision algorithm is not implemented recursively, but instead iteratively over all faces of the model. It is supported by an vertex tracking data-structure to prevent double-accounting of shared vertices across adjacent edges.


*Mountain Object At Maximum Subdivision*

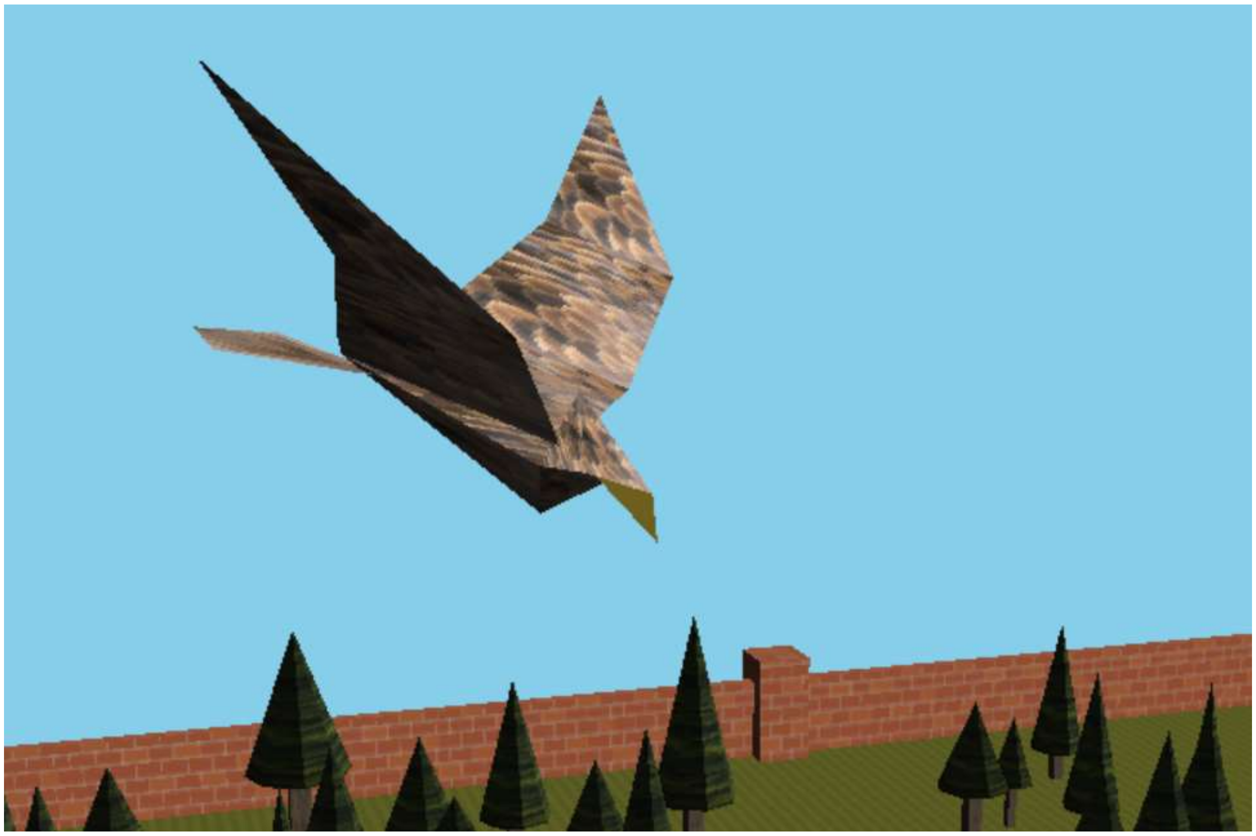
The model demonstrates the following techniques from the project rubric:

- Texture Mapping
- Subdivision

*Track / Bird*

The Track object handled via the Track class defines an animated train moving along a spline path. This class was directly taken from the project skeleton code, with modification to replace the “train” with an animated bird, flying around the mountain. The track class was also modified with wrapper functions to the bird class to allow position and orientation data from the bird to be extracted to allow for the “birds-eye” camera view.

The contained Bird object handled via the Bird class is a hierarchical model, constructed of several bird components, each defined as their own model and subsequent display lists. Upon drawing the model, the components are drawn hierarchically. In example, the wing tip, is rotated and placed onto the wing middle, and the wing middle rotated, and placed onto the wing base, etc. In this manner the bird flaps its wings in a “natural” way, and elevates and drops its tail as it flies.


*Animated Bird Object Attached to Track, Mid “Dive”*

The model demonstrates the following techniques from the project rubric:

- Texture Mapping
- Hierarchical Animated Model
- Change the Navigation System (Supports)