CSC 476 – Program 2 - due April 29th 1 minute after 11:59pm
Set up for a more complex environment for gaming & simple game play

Please note that as with all programming assignments this program should be done **individually**! You may talk to one another about the program, but you may not look at someone's working code!

This is a somewhat large project, so please get started with as soon as possible.
To start with please create a **3D fractal terrain** surface using BinTriTrees. The terrain surface must use **level of detail modeling and display** based upon distance from the viewpoint. You will also need to implement view frustum culling for this assignment. In addition, for the sake of debugging aids, the game control must include modes for displaying the scene in **wireframe mode** as well as filled polygon mode, and there must be a control for switching to **sky view mode** for observing the level of detail and VFC effects from a viewpoint high above the terrain surface (e.g., like an airplane or satellite view). The terrain should be colored in a visually pleasing way (i.e. at least 4 different materials depending on height as shown in class or at least two different texture maps). Please note the generating non-uniform terrain with view-frustum culling will be worth 80% of the points for this program. At the start of the game, you should be able to switch to sky view mode and see that the terrain is non-uniform.

The rest of this program is comprised of a simple object collection game, similar to program 1. You have a number of choices of how to implement the game. However, you are required to have a player (which can either be the camera or a mesh with 3rd person camera views) with can traverse your terrain (and not intersect it) which looks for power-ups. Your player will start with an initial amount of energy (which decreases as game play continues), which can be replenished by collecting power-ups. The player's view should float above the terrain at a constant height about the current terrain. Power-ups may be represented by something fairly complex such as another mesh (or you may create your own shape with a decent number of polygons > 100). They should appear on the terrain (their alignment with the terrain does not need to be perfect but must be reasonable – please ask me if you are unclear on this requirement). The power-ups must be randomly generated at the start of the game (you can also have them re-generate) and there should be a fair number of them. The power-ups do not need to move. You will collect the power-ups using simple collision detection with the object (or your player may use a laser with finite range to collect the power-ups). Please keep track and display the player's current energy which increases and decreases during the game depending on time (decreases) or power-ups gathered (increases). Please also implement view-frustum culling for all power-ups (i.e. compute a bounding sphere and do not draw any power-ups that are not in the current view frustum). Reasonable traversal of the terrain and interaction with the power-ups (i.e. collision detection) comprises the last 20% for this programming assignment. Please note that you may be as creative as you'd like with the game aspects of this program – if you would prefer to have "good" and "bad" objects dropping from the sky which you need to collect and avoid, that would be great also. (Your game just must include reasonable player movement across the terrain and collision detection with target items).

- ## Problem Specifications

Here are the **specifications for the** 3D game scene, built on a **terrain surface.**

1. The terrain model must be implemented with a **binTri tree data structure** (preferably using a height map to ease player interaction with the terrain).
1. The algorithm must apply pseudo-random numbers and **fractal mathematics** to **subdivide a small number of initial triangles** into a **hierarchy of many triangles**. The **subdivision depth** of the fractal terrain subdivision is a **game parameter that must be set at game initialization**.
2. The **dimensions** of the terrain must be measured in **meters**. The terrain should be **at least 1 km by 1 km**. The terrain's height will be determined by the fractal subdivision depth and initial seed values for the terrain. Please do try to generate reasonable looking terrain which includes at least one mountain.
3. The terrain model must be **initialized** with a small number of **"seed points"** that specify **elevation values** at initial subdivision vertices so that a specific, real world terrain surface can be approximated. The number and value of these seed points are **game parameters that must be set at game initialization**.
4. Terrain triangles must be stored in the **binTri tree data structure**. The terrain surface must be displayed with different **levels of detail** depending up the **distance** from the game player's (camera) **viewpoint** by selecting the appropriate level from the binTri tree. We will be doing this in an imperfect way in order to learn about the concepts but save on implementation time. That is you do not need to implement a merge operator to reduce the level of detail in an area where the user has already been. This means that you will see the benefits of L.O.D. at the beginning of your game, when you are rendering triangles not in view (i.e. far away enough) as a lower resolution (this must be able to be checked with a sky view mode). As you move through-out your world though and increase the resolution of the local terrain, you will notice that your fps will decrease – this is acceptable for this program (if your final project uses a terrain model you should invest the time to do true L.O.D. with a split and merge priority queue). In addition, please note that using the distance to the camera is not the best metric to use for computing L.O.D., (we will be discussing other methods in lecture), however, it is a fine metric for this programming assignment in order to learn about the benefits of L.O.D.
5. Implement view frustum culling (vfc) to remove any binTrinodes and power-ups that are not in the scene. Include a toggle to turn vfc on and off. Please display the frames per second in both modes to show the gain for using vfc.

Please see the lab hand-out for this week for an initial specification of how to get started on the BinTriTree. In addition, a number of useful links have been included on the class webpage for fractal terrain generation, BinTriTrees and view frustum culling.