CS 371 – Assignment 4 – Game Project – Weighted as 2 "regular" assignments

In the D2L dropbox for Assignment 4, submit by 11:59pm, Saturday, December 9, one zip file with your *the-game.html* file and all the accompanying JavaScript source code files, mesh model files, and texture files that are necessary to run your game. I will put all of the files in that zip in a directory *http://csf11.acs.uwosh.edu/cs371/xxxxxx* where *xxxxxx* (all lower case) is the last name of the person who submitted the zip archive to the dropbox. When you point a browser to *http://csf11.acs.uwosh.edu/cs371/xxxxxx/the-game.html*, everything should run as you expect. You should check this well in advance of your presentation on December 13 or 15.

The premise for this assignment is to develop that game you've always dreamed about – complete with a hero(es) pursuing a coveted object(s), all the while battling a villain(s) who is (are) also pursuing the object(s). What do you or the villain do with the object when you reach it? That's for you to decide. Perhaps you want to send it careening toward a goal of some sort? Or perhaps capture it, hold it in your stash of objects, and take off after another object. Come to think of it – why am I giving you ideas about what a creative game should be? You all know much more about what makes a good game than I do.

Your starting point is the generic game program (and associated game objects) that we discussed in class on October 25 and 27. Remember that you are the hero. Consequently your eye point is always the center of the sphere that currently represents the hero, and your look-at point is always directly ahead, focused in the direction in which you are heading. Below are the minimal requirements for the assignment. Completely meeting these minimal requirements will yield a score of 80 out of 100. Your game manual/directions on the web page from which your game runs is worth 10. And, as usual, golly-gee-whiz points can raise you to the 100 level (or better), provided that the features you add for these GGW points are clearly explained and enumerated in the opening documentation block and/or the game manual/directions that accompany your program.

Minimal requirements:

- Add two goals or stashes for the objects of pursuit at appropriate places in the arena. Upon reaching the goal/stash the object of pursuit should do something visually appropriate for the rules of your game.
- The heroes, objects of pursuit, and villains should rebound off walls using the "true" reflection model described in class on Friday, October 27.
- Make at least one villain that demonstrates "intelligent motion". Minimally this intelligent motion means:
  - The villainous player should always be seeking the object of pursuit or yourself (i.e., the hero)
- Interaction between a player (that is, hero or villain) and object of pursuit when they "collide". Be sure something interesting happens here. Moreover that interesting effect should be clearly explained in the game manual/directions on the web page. Here are some options you could consider.
  - Option 1:
    - Object of pursuit's new direction is player's old direction, plus or minus a small random perturbation.
    - Player's new direction is tweaked by a different small random perturbation
  - Option 2: View the player-object collision as the object hitting a wall whose normal vector is the player's direction vector
- Interaction between two players, that is, hero and villain, when they "collide". Again be sure something interesting happens here – and that it is clearly explained in your manual/directions.
- Add another light to create some interesting visual effect.
- Keep score using the scoring system you define in your manual/directions. Display the score in real-time on your web page.
- Make the players and objects of pursuit look much more visually interesting than they presently do. Minimally your doing this means that you must employ at least two objects coming from mesh files. One of these can be "Phong's Volkswagen", which is in the framework you receive as your starting point, but you are encouraged to change this to something more appropriate for the game you are envisioning.
- Make use of textures to:
  - Apply a texture to each of your mesh file objects. You must use either a cylindrical, spherical, or box map shape when doing this. The map shape you use should be specified in your introductory documentation block. Moreover the code you use for doing this should be done in a separate function that you call for each of your two mesh file objects. In other words, don't just cut-and-paste nearly identical code into two separate places to achieve the mapping between the mesh file object and texture space. Do this once in a clearly written function, and then call that function twice.
  - Apply at least two additional textures to objects or surfaces other than your mesh file objects.

Each GGW feature that you add beyond the minimal specifications above should be clearly noted and defined in the introductory documentation block at the beginning of your *the-game.js* program or in the web page directions that accompany your game. If that documentation on your added features is missing from either place, you well may not receive credit for it!

Start your work by picking up from D2L the *the-game.zip* archive, which contains all the code, textures, and meshes we described in class.