In this final extension of the “Jumping Brains” program, you’ll make two substantial modifications to the animation. First, using quaternions, you will change the surface on which the “Jumping Brain” characters are hopping from a flat disk to a spherical world. Second, you will add a compositied particle system to make it appear as if sparks (or “neurons”) are being emitted from each character’s brain whenever it lands on the sphere at the end of a hop. You may use a modified version of one of your previous assignments (or the instructor’s version of the third programming assignment, which is available on the course web site, http://www.cs.siue.edu/~wwhite/CS482/Syllabus.htm). The executable for one version of this new assignment, along with a texture map for the spherical world, is also available on the course website.

Several changes are needed to extend the “Jumping Brain” program for this assignment. Specifically:

- The arrow key controls must be altered to change the viewer’s camera perspective to maintain distance from the spherical platform, with that platform always in view. The left-right arrow keys will control the rotation of the camera around the y-axis (the azimuth angle, ranging from 0 to $2\pi$ radians), while the up-down arrow keys will control the altitude angle, from above the north pole (0 radians) to the south pole ($\pi$ radians).
Positioning the characters on the surface of the spherical platform so that they are hopping in the direction that they are facing is actually quite a challenge. Quaternions make this problem somewhat easier by affording a straightforward means of implementing spherical interpolation (known as “slerp”). Specifically, if P and Q are two points on the surface of the sphere, with vectors CP and CQ from the center of the sphere to these respective points, then let’s define \( \phi \) as the angle between these two vectors (i.e., \( \mathbf{CP} \cdot \mathbf{CQ} = |\mathbf{CP}| |\mathbf{CQ}| \cos \phi = R^2 \cos \phi \), where \( R \) is the radius of the sphere). Quaternion theory tells us that the slerp equation for all of the points on the great circle of the sphere joining P and Q is:

\[
P(t) = P \frac{\sin[(1-t)\phi]}{\sin \phi} + Q \frac{\sin(t\phi)}{\sin \phi}
\]

Following this arc should keep the character hopping in the right direction. The slope between the difference in azimuth and the difference in altitude of the two points P and Q provide a reasonable means for approximating the correct orientation of the character.

The particle system representing the set of “neurons” being emitted from each brain at the end of a hop should be composed of numerous line segments flowing from the center of the character for a few frames (about half as many frames as it takes to complete a hop). Color-code the particles to enable the viewer to associate particular neurons with particular brains. Other attributes should include position, position increment (i.e., velocity), size, transparency, and lifetime. These attributes should be randomly set for each particle, with the range of random values appropriately set up. A modified version of the particle class from the previous assignment may be used for this purpose.

Keep your code modular and readable, with an extensive explanation (including your name) at the top of each program file, explanatory sentences preceding each function, and in-line comments every place within the code where your logic is particularly complicated. Modify all comments in the provided code to reflect the changes that you make in your revised version. Avoid code redundancy by foregoing cut-and-paste in favor of placing any code that is needed repeatedly into its own module (function, class, structure, etc.).

Place all of your program files (just the .cpp and .h files, not your project file) into a single folder named with your last name. When the instructor creates a new Visual C++ project and properly attaches the OpenGL libraries and your code, it must compile and execute in order to be graded. Zip-compress this folder and copy it to your Moodle drop-box by **Thursday, November 30, 2017, at 12:00 Noon**. Late assignments are not accepted without verifiable medical documentation. You must write your own code (with the exception of clearly annotated code that you receive from the instructor or as part of the material available from the textbook publishers), and no one but the instructor may see your code.