In order to expand your 3D animation experience beyond a stationary model, this programming assignment extends the “Jumping Brain” model to place several of those characters in a simple 3D setting and to have them move independently. You may use your version of the previous assignment (assuming that it was complete) or the instructor’s source code, which, along with an executable version of this new assignment, is provided on the course Web site (http://www.cs.siue.edu/~wwhite/CS482/Syllabus.htm).

Several changes will be needed to extend the “Jumping Brain” program into the new “Jumping Brains” program. Specifically:

- The Jumping Brain class will need some new data members in order to place several instances within the scene. These include a position (for locating the character), and a scale factor (for resizing so that several characters can fit in the scene). Mechanisms for moving the character in its current orientation direction when it hops, as well as for elevating and lowering the character during the course of its hop, must also be employed.

- The jumping Brain class must also have the following two visual improvements. First, the color of each character should be variable. The seven Jumping Brain characters in the scene that you will be implementing should have seven different colors (red, green, blue, yellow, magenta, cyan, and white). Second, in an effort to make the brain look more brain-like, the large egg-shaped “brain” will be replaced with a series of small spheres that coat the surface where the old ovoid once was. (Spherical coordinates are a convenient way to implement this feature.)

- The seven Jumping Brain characters will be scattered on the surface of a large disk (use gluDisk to implement this, similar to gluSphere and gluCylinder from the previous assignment). Each character should be oriented differently, and animation will begin as soon as the application starts (i.e., no keyboard controls for starting, stopping, or altering the speed of the characters).

- No collision detection or avoidance between characters is required in this assignment, but when any character hops beyond the boundary of the disk, it should plummet into the abyss below.

- Graduate students are required to include a “reset” keyboard control that will reposition all characters in their original positions and orientations when the user presses the “R” key.

All old keyboard controls (and their associated routines) should be removed, but new controls for the arrow keys are needed to implement user control over the viewer’s left-right viewing angle and up-down viewing angle. There should be appropriate boundaries placed on these angles, as illustrated in the executable.
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 “Jumping Brain” code to reflect the changes that you make in your “Jumping Brains” 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, September 28, 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.