Project 1 - Transformation with Trackball and STL file

CS 1566 — Introduction to Computer Graphics

Check the Due Date on the Canvas

The purpose of this project is for you to transform (rotate and scale) two different sources of objects in three dimensional space using a mouse or a track pad.

Part I: 3D Objects (40 Points)

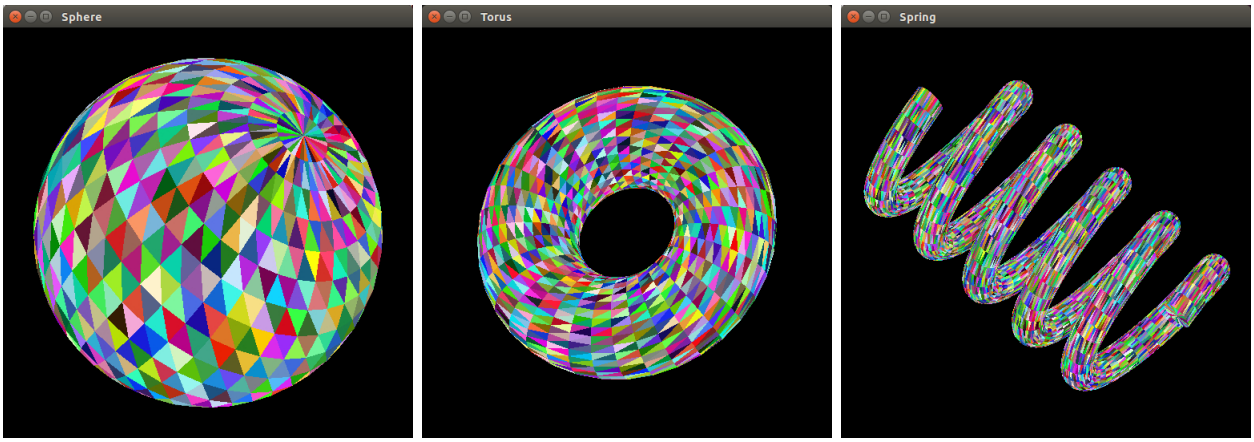
For this project, you need to create the following objects:

- a computer generated object, and
- an object where its vertex positions are stored in a file.

When you program starts, your program should ask the user whether he/she wants to view a computer generated or a file.

Computer Generated Object (20 Points)

For a computer generated three-dimensional object, its surface must consist of a number of triangles. Each triangle of the object should have a random color. For best result, the center of these objects must be at the origin. You must create either a sphere, a torus, or a spring as shown below:

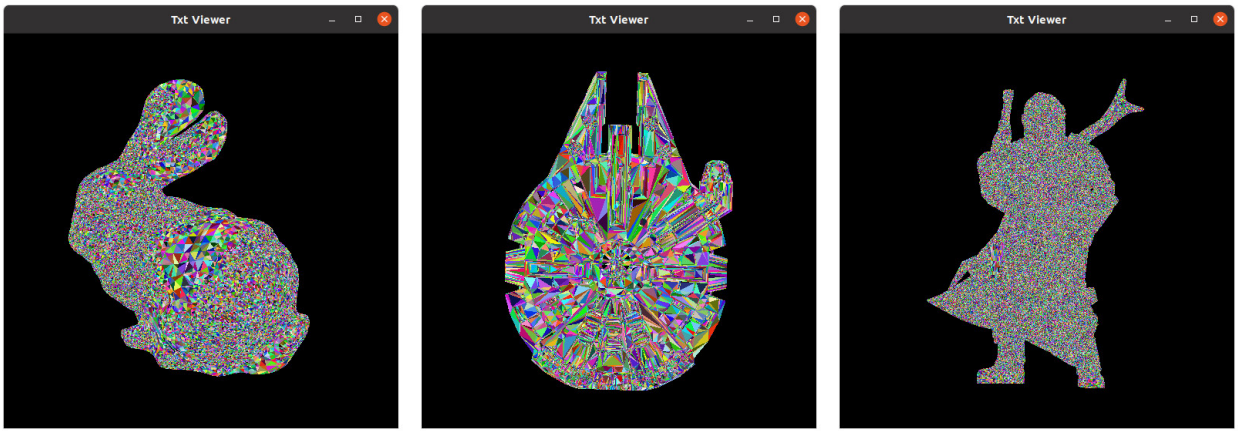


Vertices Data from a File (20 Points)

A number of `zip` files are provided. Inside each `zip` file, there is a plain-text file. The first line of the text file is an integer representing the number of vertices in the file. Each line after that is a vertex position x, y, z, and w. For example, here is the beginning of the text file `cube.txt`:

```
36
-35.000000,60.000000,20.000000,1.000000
-55.000000,60.000000,20.000000,1.000000
-35.000000,40.000000,20.000000,1.000000
-35.000000,40.000000,20.000000,1.000000
:
```

It tells you that there are 36 vertices. The position of the first vertex is $(-35.0, 60.0, 20.0, 1.0)$. The position of the second vertex is $(-55.0, 60.0, 20.0, 1.0)$ and so on. Note that the data are in homogeneous coordinate and the fourth element is always 1.0 for a vertex position. Use either `bunny.txt` (337,206 vertices0, `falcon.txt` (444,984 vertices), or `mandalorian.txt` (6,474,768 vertices) for this part. `cube.txt` (36 vertices) and `menger_sponge.txt` (are given but you should use them for testing since they contain a small number of vertices. There is no color information in any given file. Simply give every three vertices (a triangle) a random color as shown below:



Note that it may be a little hard to see the detail since we did not have a lighting effect yet.

Zoom In and Zoom Out (10 Points)

For this project, we will use the scroll wheel of a mouse to scale an object. Scroll one way is an equivalent of enlarge an object in all direction about the origin by the factor of 1.02. Similarly, to shrink the object by the factor of 1/1.02 can be done by scrolling the other way.

Scrolling events are the same as mouse event. So, you need to call the `glutMouseFunc()` function to register your callback function:

```
glutMouseFunc(mouse);
```

where your `mouse()` function should look like the following:

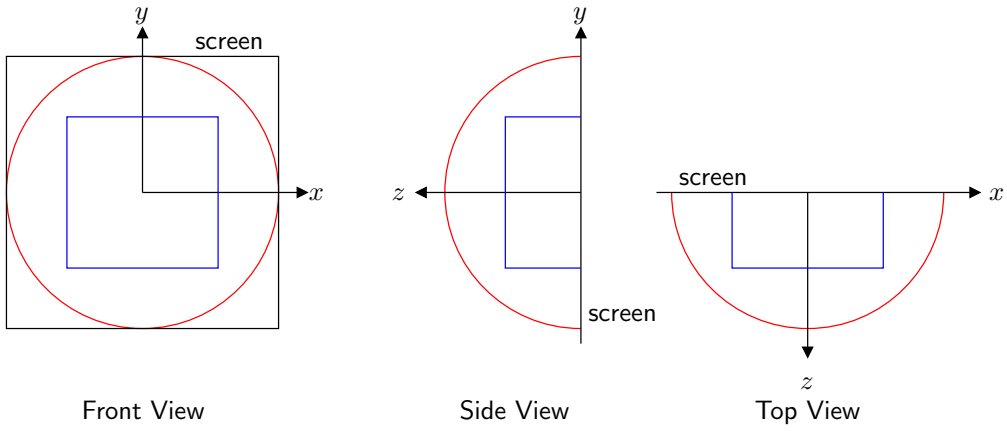
```
void mouse(int button, int state, int x, int y)
{
   :
}
```

If you scroll up, the `mouse()` function will be call with the variable `button` initialized to 3. Similarly, if you scroll down, the variable `button` will be initialized to 4. Simply apply the scaling matrix and call the `glutPostRedisplay()` function.

Note that if you use the track pad of your laptop, it may not registered as a scroll wheel. In this case, use a couple keys on your keyboard to perform zoom in and zoom out instead.

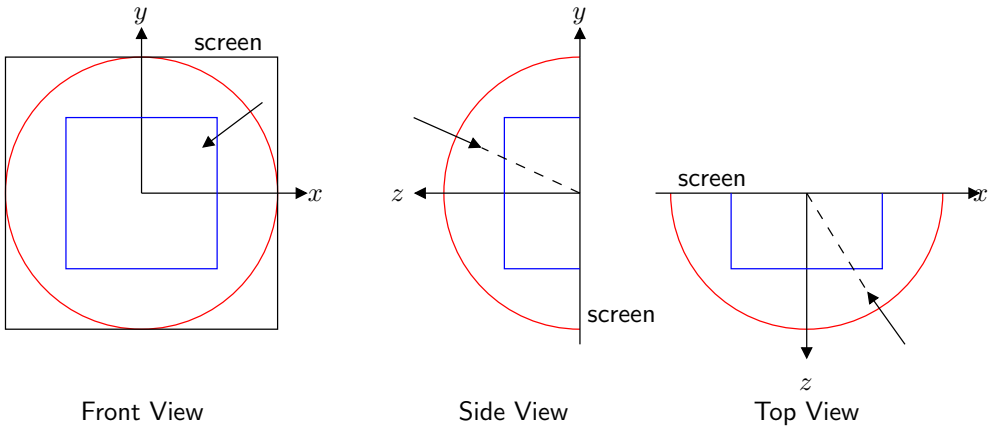
Trackball Style Rotation (50 Points)

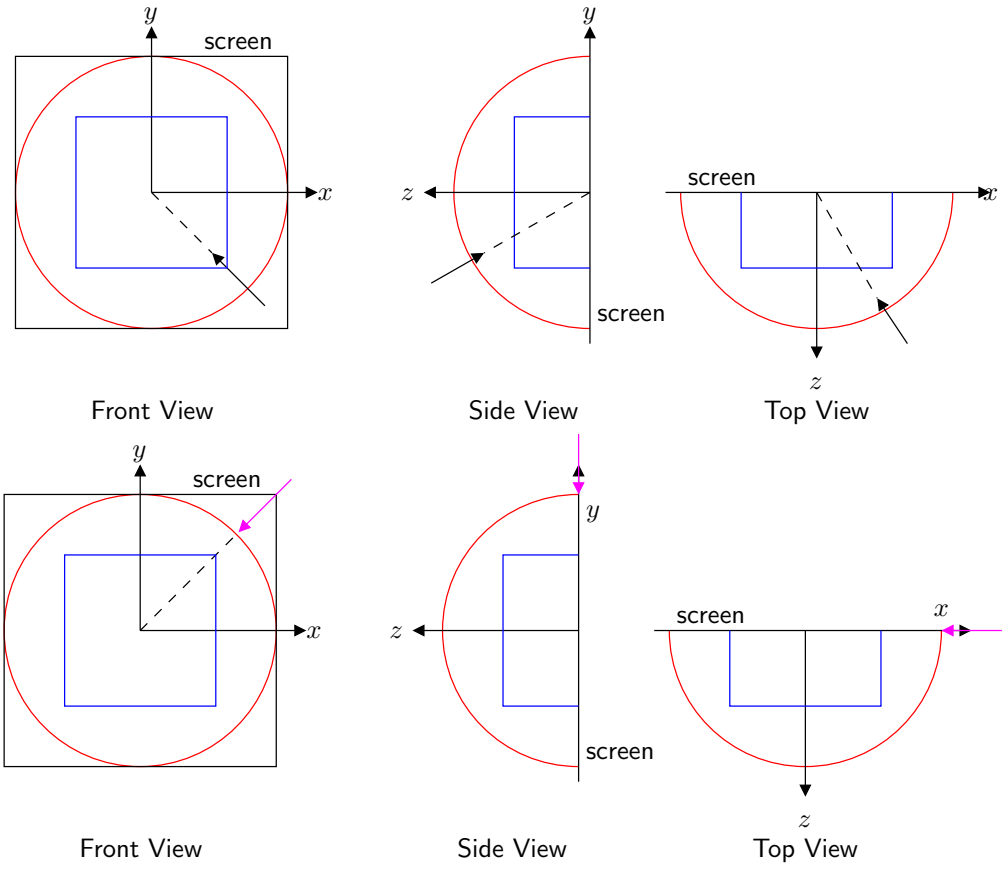
To rotate an object in 3D, simply imagine that your object is located in the middle of a glass ball. This glass ball can be spun in any direction. Now, imagine that half of this glass ball pops out of your screen as shown below:



From the above picture, there is a blue cube sitting inside this glass ball. If the glass ball is rotated, this cube is rotated as well.

Note that when a user click a mouse on the screen which is a two-dimensional surface, you have to imagine that the mouse pointer is a finger that touch the glass and point directly to the center of the glass ball in three-dimension. Mouse function only provide your x and y position (**screen coordinate**) but you have to come up with your imaginary z since it is in three-dimensional space. Here are some examples:





To rotate the object inside this class ball, a user needs to simply move his/her finger while touching the ball. For this project, assume that a user's finger is always point directly to the origin while it is moving. Ideally, a user can twist his/her finger to rotate the glass ball. But since we cannot twist the mouse pointer, we assume that twisting the finger is not allow for this project. We will use left button of a mouse to simulate a user touches the glass ball. If the left button is down, user touches the ball at the current pointer position. If the left button is up, user released his/her finger from the ball. To capture the left button event, we use the same callback as in previous section. The variable `button` will be initialized to `GLUT_LEFT_BUTTON` and the variable `state` will be initialized to either `GLUT_UP` or `GLUT_DOWN`. The variables `x` and `y` will be set to the pointer position. **Note** that the pointer position is the **screen position**. The top-left corner of the screen is at $(0, 0)$ and the bottom right is $(511, 511)$ for a 512×512 window.

If the mouse pointer is moving while the left button is down, it simulates a user turning the glass ball. When the glass ball rotates, it rotates about a vector and the fixed point of rotation is at the origin. Your job is to come up with the vector so that you can apply rotation matrices correctly. A method of calculating this vector will be discussed in class.

To capture mouse motion events, use the `glutMotionFunc()` function as shown below:

```
glutMotionFunc(motion);
```

and the `motion()` function should look like the following:

```
void motion(int x, int y)
{
```

```
    :
}
```

The variables `x` and `y` will be set to the current pointer position.

Spinning an Object (Just for fun)

One special feature of this glass ball is that it can rotate indefinitely (no friction). If a user touches the glass ball, drags his/her finger, and releases the finger, the glass ball should spin in the same direction of the user's finger indefinitely. **Note** that the zoom-in/zoom-out feature must work while the object is spinning indefinitely.

Submission

The due date of this project is stated on the Canvas. Late submissions will not be accepted. Zip all files related to this project into the file named `project1.zip` and submit it to the Canvas. Do not forget that you must demo this project to your TA on the due date during your recitation.