Project 1 - Transformation with Trackball JavaScript

CS 1566 — Introduction to Computer Graphics

Check the Due Date on the Canvas

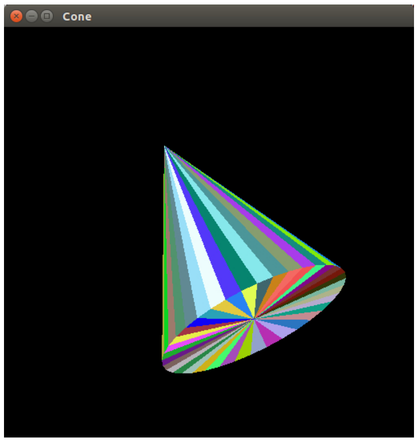
The purpose of this project is for you to transform (rotate and scale) an object in three dimensional space using a mouse or a track pad.

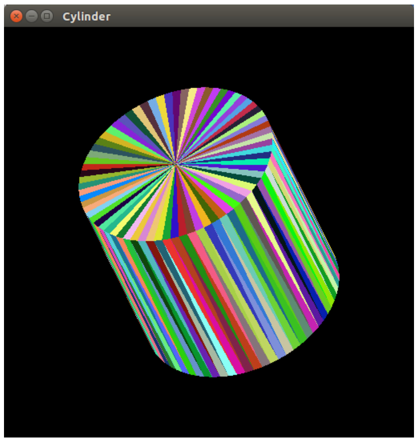
Part I: 3D Objects (40 Points)

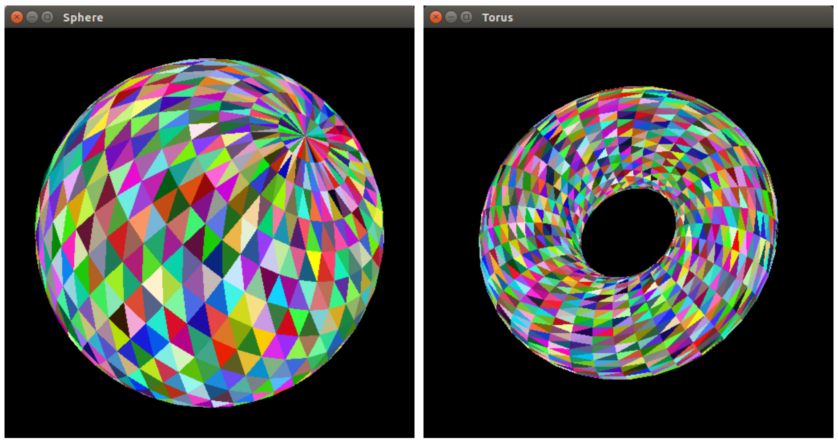
For this project, you need to create a computer generated three-dimensional objects where each surface (triangle) of the object has a random color. For best result, the center of these objects should be at the origin (but not necessary). The maximum score of this part is 40 points. The score of this part will be based on the difficulty of the object you decide to create as follows:

- Cube: 5
- Cone: 5 Points
- Cylinder: 10 Points
- Sphere: 10 Points
- Torus: 10 Points

Here are some examples:







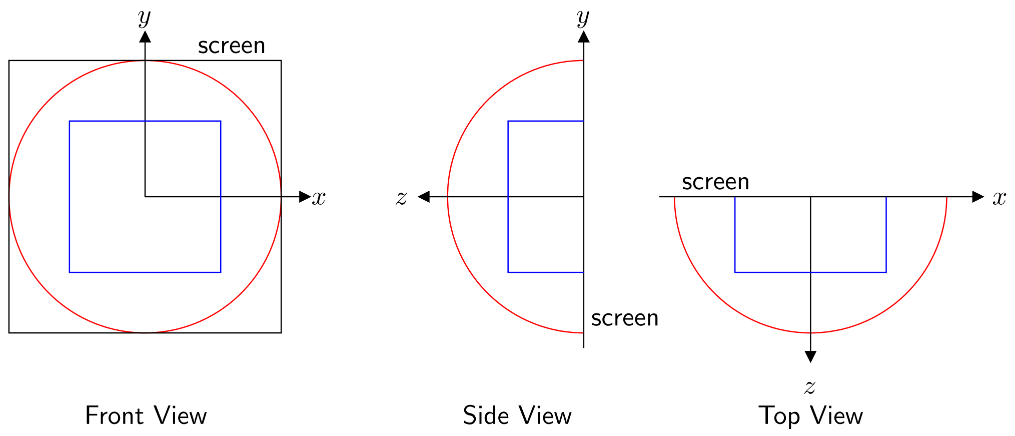
Scaling (10 Points)

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.

A touchpad can also be used to scale object but you need to set the callback function for scrolling on the touchpad correctly. Touchscreen may be used as well. But for simplicity, you can simply use two keys of your keyboard to enlarge or shrink objects.

Trackball Style Rotation (40 Points)

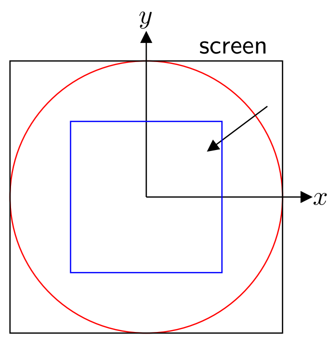
To rotate an object in 3D, simply imagine that your object is located in the middle of a glass ball. This glass ball can be spun in any direction. Now, imagine that half of this glass ball pops out of your screen as shown below:



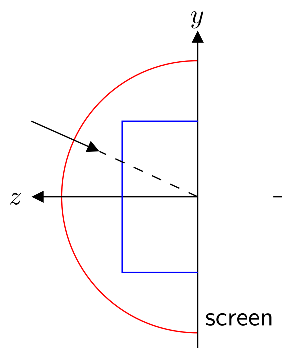
From the above picture, there is a blue cube sitting inside this glass ball. If the glass ball is rotated, this cube is rotated as well.

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 you x and y positions (**screen**

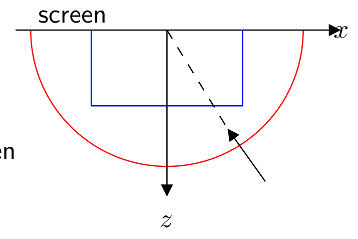
coordinate) but you have to come up with your imaginary z since it is in three-dimensional space. Here are some examples:



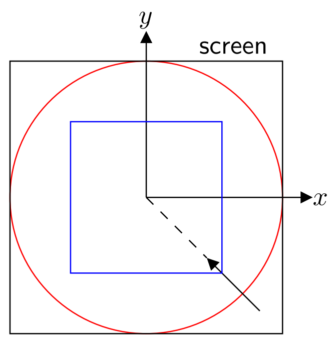
Front View



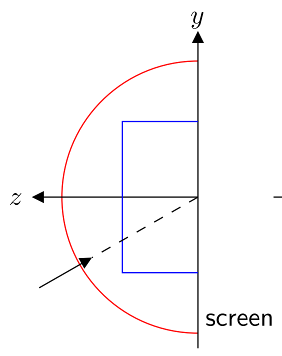
Side View



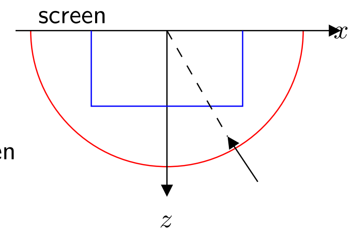
Top View



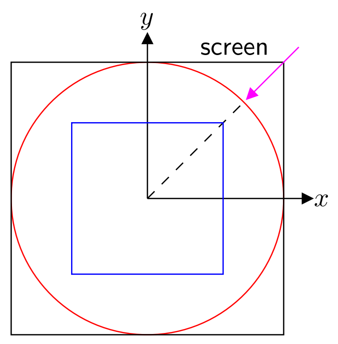
Front View



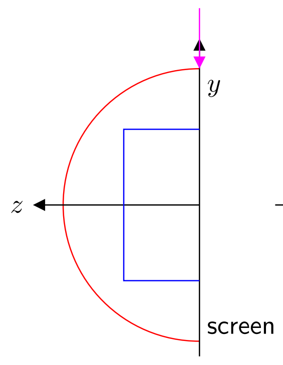
Side View



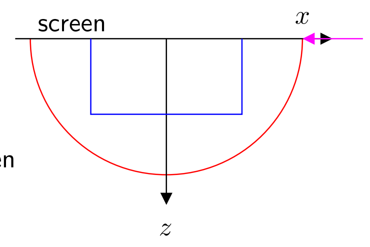
Top View



Front View



Side View



Top View

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 events of mouse buttons being down or up, use use `onmousedown` and `onmouseup` of the `canvas`. Here is the `main()` function of a `starter` code:

```
function main()
{
    canvas = document.getElementById("gl-canvas");
    if(initGL(canvas) == -1)
        return -1;
    if(init() == -1)
        return -1;

    // Register callback functions
    // Comment out those that are not used.
    canvas.onmousedown = mouseDownCallback;
    canvas.onmouseup = mouseUpCallback;
    canvas.onmousemove = mouseMoveCallback;
    document.onkeydown = keyDownCallback;
    :
```

The `mouseDownCallback`, `mouseUpCallback`, and `mouseMoveCallback` are functions that will be called when its associate event occurs. Here are those functions:

```
// This function will be called when a mouse button is down inside the canvas.
function mouseDownCallback(event)
{
    console.log("mouseDownCallback(): " +
                "event.which = " + event.which +
                ", x = " + (event.clientX - canvas.offsetLeft) +
                ", y = " + (event.clientY - canvas.offsetTop));
}

// This function will be called when a mouse button is up inside the canvas
function mouseUpCallback(event)
{
    console.log("mouseUpCallback(): " +
                "event.which = " + event.which +
                ", x = " + (event.clientX - canvas.offsetLeft) +
                ", y = " + (event.clientY - canvas.offsetTop));
}

// This function will be called when a mouse pointer moves over the canvas.
function mouseMoveCallback(event)
{
    console.log("mouseMoveCallback(): " +
                "event.which = " + event.which +
                ", x = " + (event.clientX - canvas.offsetLeft) +
                ", y = " + (event.clientY - canvas.offsetTop));
}
```

Note that the pointer position is the **canvas coordinate**. The top-left corner of the canvas is at $(0, 0)$ and the bottom right is $(511, 511)$ for a 512×512 canvas.

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.

Spinning an Object (10 Points)

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. The speed of spinning should be associated with the speed of the user dragging the glass ball right before he/she lefts his/her finger. **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 including directory structure 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.