There are four tasks each carrying the same weight.

1. Fully Controllable Camera (1.exe)
2. Sphere to/from Cube (1.exe)
3. Hand Simulation (2.exe)
4. Wheel Simulation (3.exe)

1. Fully Controllable Camera (1.exe)

up arrow - move forward
down arrow - move backward
right arrow - move right
left arrow - move left
PgUp - move up
PgDn - move down
1 - rotate/look right
2 - rotate/look left
3 - look up
4 - look down
5 - tilt clockwise
6 - tilt counterclockwise

Hint:

Maintain 4 global variables: 1 3d point `pos` to indicate the position of the camera and 3 3d **unit** vectors `u`, `r`, and `l` to indicate the up, right, and look directions respectively. `u`, `r`, and `l` must be perpendicular to each other, i.e., `u.r = r.l = l.u = 0`, `u` = `r` X `l`, `l` = `u` X `r`, and `r` = `l` X `u`. You should initialize and maintain the values of `u`, `r`, and `l` such that the above property holds throughout the run of the program. For example, you can initialize them as follows: `u = (0, 0, 1)`, `r = (-1/√2, 1/√2, 0)`, `l = (-1/√2, -1/√2, 0)`, and `pos = (100, 100, 0)`. And while changing `u`, `r`, and `l`, make sure that they remain unit vectors perpendicular to each other.

The first 6 operations listed above are move operations, where the position of the camera changes but the up, right, and look directions do not. The last 6 operations are rotate operations, where the camera position does not change, but the direction vectors do.

In case of a move operation, move `pos` a certain amount along the appropriate direction, but leave the direction vectors unchanged. For example, in the move right operation, move `pos` along `r` by 2 (or by any amount you find appropriate) units.

In case of a rotate operation, rotate two appropriate direction vectors a certain amount around the other direction vector, but leave the position of the camera unchanged. For example, in the look up operation, rotate `l` and `u` counterclockwise with respect to `r` by 3 (or by any amount you find appropriate) degrees [vector.ppt slide#12].

If you maintain `pos`, `u`, `r`, and `l` in this way, your `gluLookAt` statement will look as follows:

```
gluLookAt(pos.x, pos.y, pos.z,
          pos.x + l.x, pos.y + l.y, pos.z + l.z,
          u.x, u.y, u.z);
```

2. Sphere to/from Cube (1.exe)

   Home - cube to sphere
   End - sphere to cube

   Draw one eighth of a sphere, one fourth of a cylinder and a square once.

   Use transformations (translation, rotation etc.) to put them in the right places.

   It is not required to imitate the color patterns, but the shape of the object must be discernible.

3. Hand Simulation (2.exe)

   Press the keys 1, 2, 3, ..., 9, 0, q, and w to find out how they work.
   Also observe that after a certain amount, each joint ceases to rotate.
   Use arrow keys to move the camera.

   You can use the OpenGL library function `glutWireSphere` and scale it to draw the parts of the arm.

4. Wheel Simulation (3.exe)

   w - move forward
   s - move backward
   a - rotate left
   d - rotate right
   Use arrow keys to move the camera.

Keep in mind that a full (360 degree) rotation of the wheel moves the wheel linearly by a length equal to the perimeter of the wheel.[1]

It is not required to imitate the color patterns, but the shape of the wheel must be discernible.

Hint:

Maintain 3 global variables: a 2d point `pos` to indicate the position of the center of the wheel on the XY plane (note that you do not need to store the z coordinate of the center of the wheel, because it is always equal to the radius of the wheel), a 2d vector `v` to indicate the forward direction (the direction the wheel moves when w is pressed) and an angle `theta` to indicate the rotation of the wheel with respect to its axis of rotation.

When a or d is pressed, rotate `v` by 3 (or by any amount you find appropriate) degrees on the XY plane counterclockwise or clockwise respectively [vector.ppt slide#11]. When w is pressed, change `pos` along `v` (or `–v,` if s is pressed) by 2 (or by any amount you find appropriate) units and change `theta` in accordance with [1].

You should draw the wheel in a standard location and use appropriate transformations to put the wheel in the right position and angle as determined by `pos`, `v`, and `theta`.

Note that, you can use an angle instead of the vector `v` to indicate the forward direction of the wheel. But it is recommended (not mandatory) that you use the vector form.