There are four tasks each carrying the same weight.

1. Fully Controllable Camera (1.exe)
2. Gun (1.exe)
3. Bubbles (2.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 left
2 - rotate/look right
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);
```

Marks: 10

2. Gun (1.exe)

Color pattern of the gun should be same as the one modeled in 1.exe. You can't use any OpenGL library function to draw the parts of the gun.

Press the keys q, w, e, r, a, s, d, and f to find out how the gun rotates.
Also observe that after a certain amount, each joint ceases to rotate.

Left click the mouse to fire the gun.
Draw the gunshots (red squares) slightly in front of the wall to avoid glitches. Be careful to ensure that your model is located well within the far distance (assigned in `gluPerspective`) from the camera.

Right click the mouse to toggle viewing the axis. (not a mandatory requirement)

Marks: Model the gun (7) + Rotate the gun (7) + Fire the gun (6) = 20

3. Bubbles (2.exe)

left arrow – steer the yellow bubble to the left
right arrow – steer the yellow bubble to the right

The Camera is fixed looking at the center of the boundary circle from a perpendicular position. The radius of the boundary circle is 10 times the radius of the bubbles. The bubbles have the same velocity.

Hint:

Maintain 2 global variables for each bubble to indicate its position and speed. A 2d point `p1` indicates the position of the center of the yellow bubble on the XY plane, and a 2d vector `v1` indicates the forward direction of the yellow bubble. Similarly `p2` and `v2` indicate respectively the position and forward direction of the green bubble.

In the display function, write codes to draw the bubbles in appropriate position (as indicated by p1 and p2) and arrows in the appropriate direction (as indicated by v1 and v2). The `atan2` function may come handy while drawing the arrows in the appropriate direction.

In the idle function, update the position of the bubbles (`p1` and `p2`) by adding to them the respective direction vectors (`v1` and `v2`). Check whether a bubble (or, both bubbles) intersects the boundary circle. If yes, reflect its direction vector on the boundary circle [vector.ppt slide#14]. Also check if the bubbles intersect themselves. If yes, determine the unit vector `n` perpendicular to the colliding line. Use `n` to reflect the direction vectors of both bubbles.

When left or right arrow key is pressed, rotate `v1` by 3 (or by any amount you find appropriate) degrees on the XY plane counterclockwise or clockwise respectively [vector.ppt slide#11].

Note that, you can use an angle instead of the vectors `v1` and `v2` to indicate the forward direction of the bubbles. But it is recommended (not mandatory) that you use the vector form.

Marks: 10