July 2023 CSE 410 - Computer Graphics Sessional
Offline - 1
Assignment on OpenGL

Deadline: 08 December 2023, 11:45 PM

Task 1 : Movement

Controlling Camera

You need to code a fully controllable camera as follows:

Translation

`up arrow`	–move forward
`down arrow`	–move backward
`right arrow`	–move right
`left arrow`	–move left
`page up`	–move up
`page down`	–move down

Rotation

`1`	–rotate/look left
`2`	–rotate/look right
`3`	–look up
`4`	–look down
`5`	–tilt counterclockwise
`6`	–tilt clockwise

Please play with the samples for further clarification.

Bonus

Implement the following two camera movements.

`w`	– move up without changing reference point
`s`	– move down without changing reference point

Task 2: Rolling Ball

- Draw a ball on the xy plane. Each ball has a direction (in the xy plane) and a movement speed.
- Draw the ball by dividing it into sectors and stacks. You can follow this link. Neighbouring sectors will have alternating colours. See the sample.
- Show an arrow in the current direction of the ball.
- Draw the floor as a checkerboard.
- There should be two modes: manual control and simulation. To shift between modes, press space.
- The ball has to rotate on its own axis (rolling effect).

Manual Control

A user can control the ball. The controls are:

Controls

`j` –direction will rotate counterclockwise
`l` –direction will rotate clockwise
`i` –go forward
`k` –go backward

Simulation mode

In this mode, we will implement a billiard board with one ball. A ball will move in a region bounded by planes (not necessary 4). The normals of the planes will be coplanar with the balls' direction. When a collision occurs, the direction of the ball should be updated using the reflection formula.

Time is continuous, but we have to consider time as a discrete unit. There are two natural approaches:

1. Time-driven simulation.
2. Event-driven simulation.

You may give a read. We will **use event-driven simulation** for smooth and correct simulation. You may start initially with a time-driven simulation. A change of direction of ball should also work in simulation mode.

Controls

`j` –direction will rotate counterclockwise
`l` –direction will rotate clockwise

Please see the sample `rolling_ball.exe` (windows) / `rolling_ball.out` (linux) file.

Task 3 : Magic Cube

- The magic cube makes a transition between a sphere and an octahedron.
- Pressing the following keys will change the shape:
  `.` (dot) – sphere to octahedron
  `,` (comma) – octahedron to sphere
- You should draw only one triangle, one cylinder segment and one sphere segment, and then use transformations (translation, rotation, scaling) to put them in the right places.
- Try to make sure the surface stays smooth where any triangle, cylinder segment or sphere segment touches.
- The fully sphere state of the magic cube can be inscribed into the fully octahedron state so that the sphere touches the faces of the octahedron as shown in Figure-1.
- *You need to do a fair bit of geometry for this task.*

Please play with the sample `magic_cube.exe` (windows) / `magic_cube.out` (linux) for clarification.

Controlling Object

You need to control the magic cube as follows:
`a` – rotate the object in the clockwise direction about its own axis
`d` – rotate the object in the counter-clockwise direction about its own axis.

Hint

- An octahedron has 8 triangular faces, 12 edges and 6 vertices. You need to draw cylinder segments in place of the edges and sphere segments in place of vertices.
- The faces are equilateral triangles. The base triangle that you draw should have its vertices at $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$.
- The dihedral angle (ϕ) of an octahedron (the angle at which two adjacent faces of the octahedron are joined to each other) is $\phi = \arccos(-\frac{1}{3}) = 109.47°$. Therefore, each cylinder segment will make an angle of $180° - \phi = 70.5287794°$ in its centre (as shown in Figure-2).

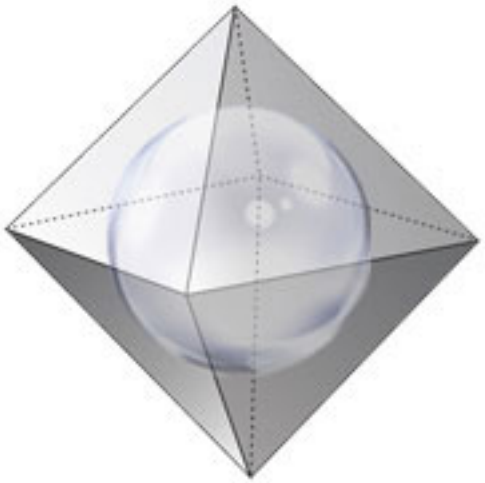

Figure 1: The sphere state can be inscribed inside the octahedron state

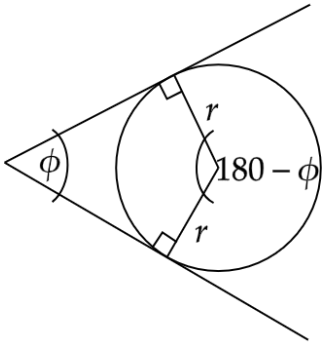

Figure 2: Angle of each cylinder segment.

- The sphere should be divided into six identical pieces. It is better to use subdivision methods to draw the sphere segments. You can follow this link.

Submission

1. Create a directory with your 7-digit student id as its name.
2. Put all the source files only into the directory created in step 1. The source files should be named `rolling_ball.cpp` and `magic_cube.cpp`
3. Zip the directory (compress in .zip format. Any other format like .rar, .7z etc. are not acceptable).
4. Upload the .zip file in Moodle.

Special Instructions

1. Please DO NOT COPY solutions from anywhere (e.g., your friends, seniors, internet). Any form of plagiarism, irrespective of source or destination, will result in -100% marks int online/offline.
2. Do not hardcode. Use variables whenever possible. This would be helpful for you during **online**.

Mark Distribution

Task	Mark
1	20
2	40
3	40