Batch: B2 Roll No.: 16010121110

Experiment No. 07

TITLE: Write an OpenGL program to implement Shadow Mapping.

AIM:

Write an OpenGL program to implement Shadow Mapping.

Create 3D object and demonstrate the shadow of same object.

Expected OUTCOME of Experiment:

shadow mapping

Books/ Journals/ Websites referred:

http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/

Algorithm/ Pseudocode for each process:

1) **Make a 3d object**
2) **Make a point light source**
3) **Check shadow of the object**

Implementation details:

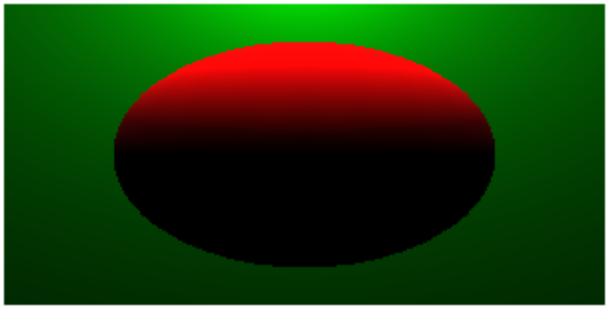
Shadow of an sphere on itself

```
<html>
 <head>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script>
 </head>
 <body>
  <script>
   const scene = new THREE.Scene();

   // Create a WebGLRenderer and turn on shadows in the renderer
   const renderer = new THREE.WebGLRenderer();
   renderer.shadowMap.enabled = true;
   renderer.setClearColor(0xffffff); // Set the background color to white
```

```
    document.body.appendChild(renderer.domElement);

    // Create a DirectionalLight and turn on shadows for the light
    const light = new THREE.DirectionalLight(0xffffff, 1);
    light.position.set(0, 1, 0); // default; light shining from the top
    light.castShadow = true; // default false
    scene.add(light);



    // Create a sphere that casts shadows (but does not receive them)
    const sphereGeometry = new THREE.SphereGeometry(5, 32, 32);
    const sphereMaterial = new THREE.MeshStandardMaterial({ color: 0xff0000
});
    const sphere = new THREE.Mesh(sphereGeometry, sphereMaterial);
    sphere.castShadow = false; // default is false
    sphere.receiveShadow = false; // default
    scene.add(sphere);

    // Create a plane that receives shadows (but does not cast them)
    const planeGeometry = new THREE.PlaneGeometry(50, 50, 32, 132);
    const planeMaterial = new THREE.MeshStandardMaterial({ color: 0x00ff00
});
    const plane = new THREE.Mesh(planeGeometry, planeMaterial);
    plane.receiveShadow = true;
    scene.add(plane);


     //Create a PointLight and turn on shadows for the light
const light2 = new THREE.PointLight( 0xffffff, 1, 100 );
light2.position.set( 0, 10, 4 );
light2.castShadow = true; // default false
scene.add( light2 );

    // Set the camera position and look at the scene
    const camera = new THREE.PerspectiveCamera(75, window.innerWidth /
window.innerHeight, 0.1, 1000);
    camera.position.z = 10;
    camera.lookAt(0, 0, 10);

    // Render the scene
    const animate = () => {
      requestAnimationFrame(animate);
      renderer.render(scene, camera);
    };

    animate();
  </script>
```

```
 </body>
</html>
```

Output(s) (Screen Shot):



Conclusion and discussion:

Three. js is a cross-browser JavaScript library and application programming interface (API) used to create and display animated 3D computer graphics in a web browser using WebGL. By using ThreeJS, we used OpenGL in backend to render shadows.

Date: 13 Oct 2023

Signature of faculty in-charge

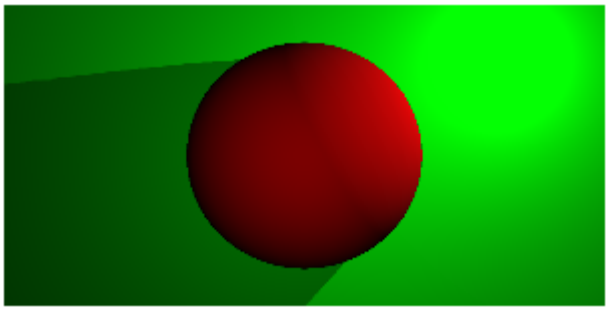
Post lab question

Write a program to demonstrate shadow for two objects.

```
<html>
 <head>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script>
 </head>
 <body>
  <script>
    const scene = new THREE.Scene();

    // Create a WebGLRenderer and turn on shadows in the renderer
    const renderer = new THREE.WebGLRenderer();
    renderer.shadowMap.enabled = true;
    renderer.setClearColor(0xffffff); // Set the background color to white
```

```
    document.body.appendChild(renderer.domElement);

    // Create a DirectionalLight and turn on shadows for the light
    const light = new THREE.PointLight(0xffffff, 0.5,100);
    light.position.set(0, 0, 10); // default; light shining from the top
    light.castShadow = true; // default false
    scene.add(light);



    // Create a sphere that casts shadows (but does not receive them)
    const sphereGeometry = new THREE.SphereGeometry(5, 32, 32);
    const sphereMaterial = new THREE.MeshStandardMaterial({ color: 0xff0000
});
    const sphere = new THREE.Mesh(sphereGeometry, sphereMaterial);
    sphere.castShadow = true; // default is false
    sphere.receiveShadow = false; // default
    scene.add(sphere);

    // Create a plane that receives shadows (but does not cast them)
    const planeGeometry = new THREE.PlaneGeometry(50, 50, 32, 132);
    const planeMaterial = new THREE.MeshStandardMaterial({ color: 0x00ff00
});
    const plane = new THREE.Mesh(planeGeometry, planeMaterial);
    plane.receiveShadow = true;
    scene.add(plane);


     //Create a PointLight and turn on shadows for the light
const light2 = new THREE.PointLight( 0xffffff, 1, 100 );
light2.position.set( 10, 5, 4 );
light2.castShadow = true; // default false
scene.add( light2 );

    // Set the camera position and look at the scene
    const camera = new THREE.PerspectiveCamera(75, window.innerWidth /
window.innerHeight, 0.1, 1000);
    camera.position.z = 10;
    camera.lookAt(0, 0, 10);

    // Render the scene
    const animate = () => {
      requestAnimationFrame(animate);
      renderer.render(scene, camera);
    };

    animate();
  </script>
 </body>
```

</html>



Explanation -
Shadow number 1 - shadow of front part of sphere on to itself
Shadow number 2 - shadow of sphere on the plane

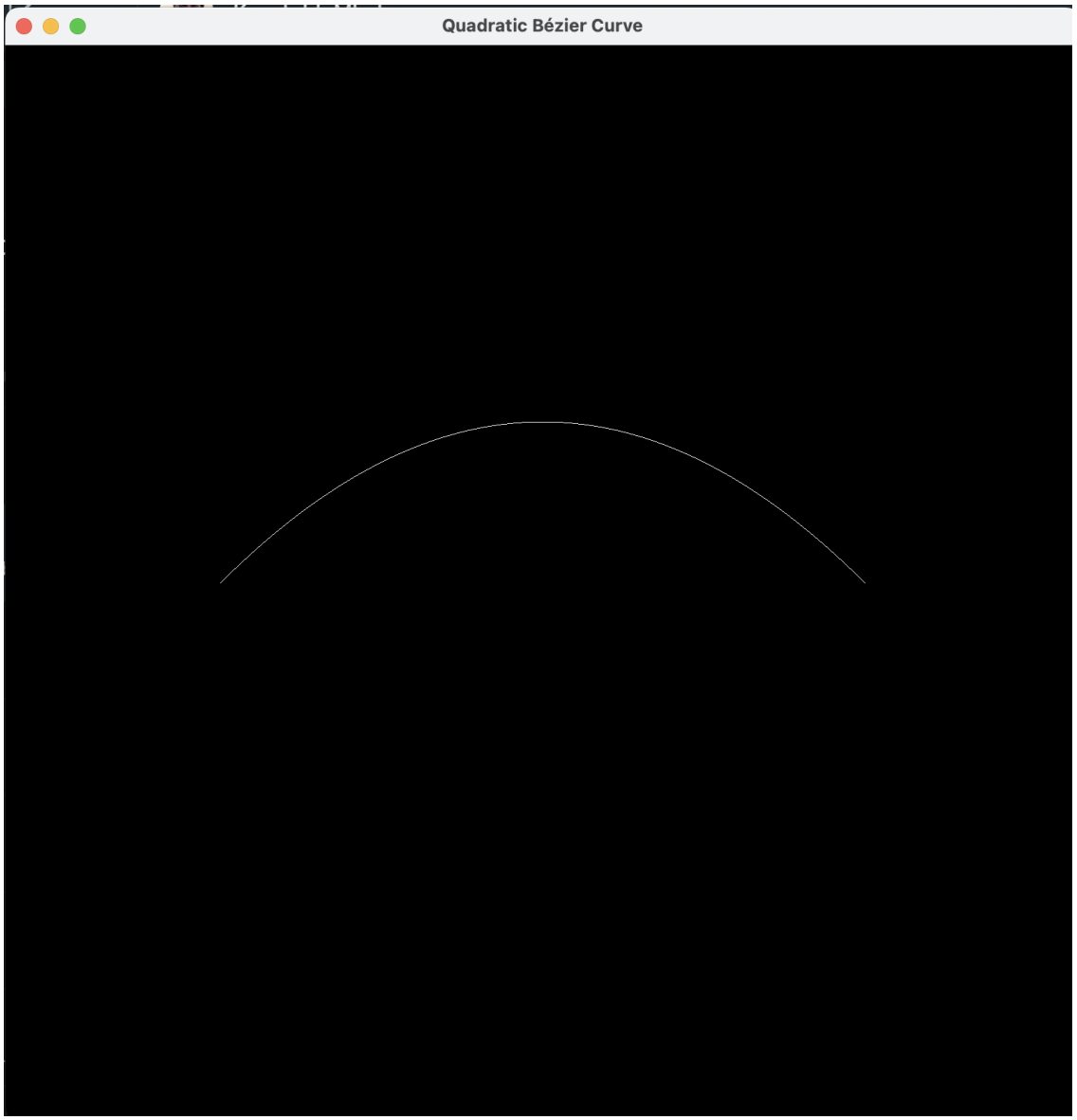
Write to program to implement various curves (at least two - three types of curve)

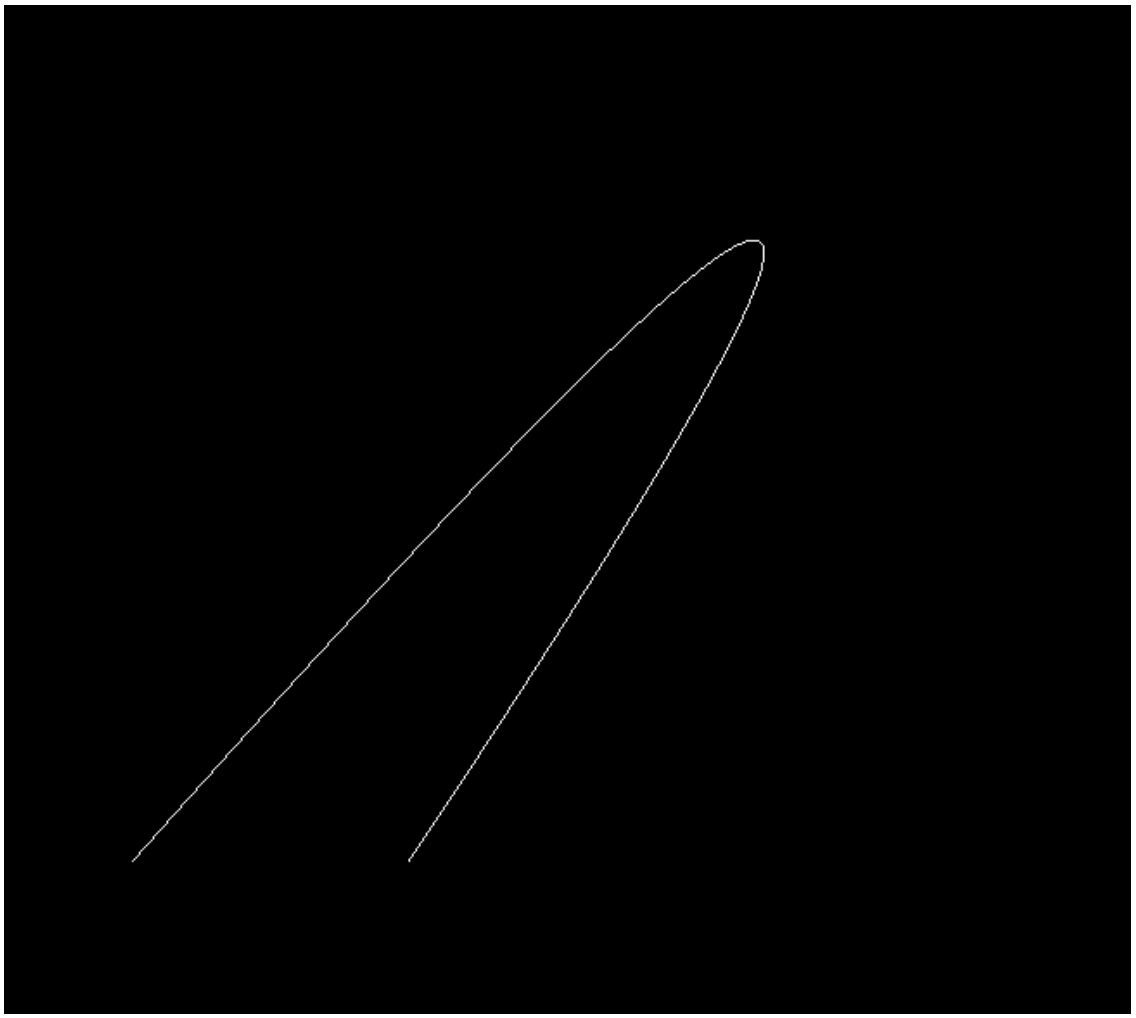
Beizer curves

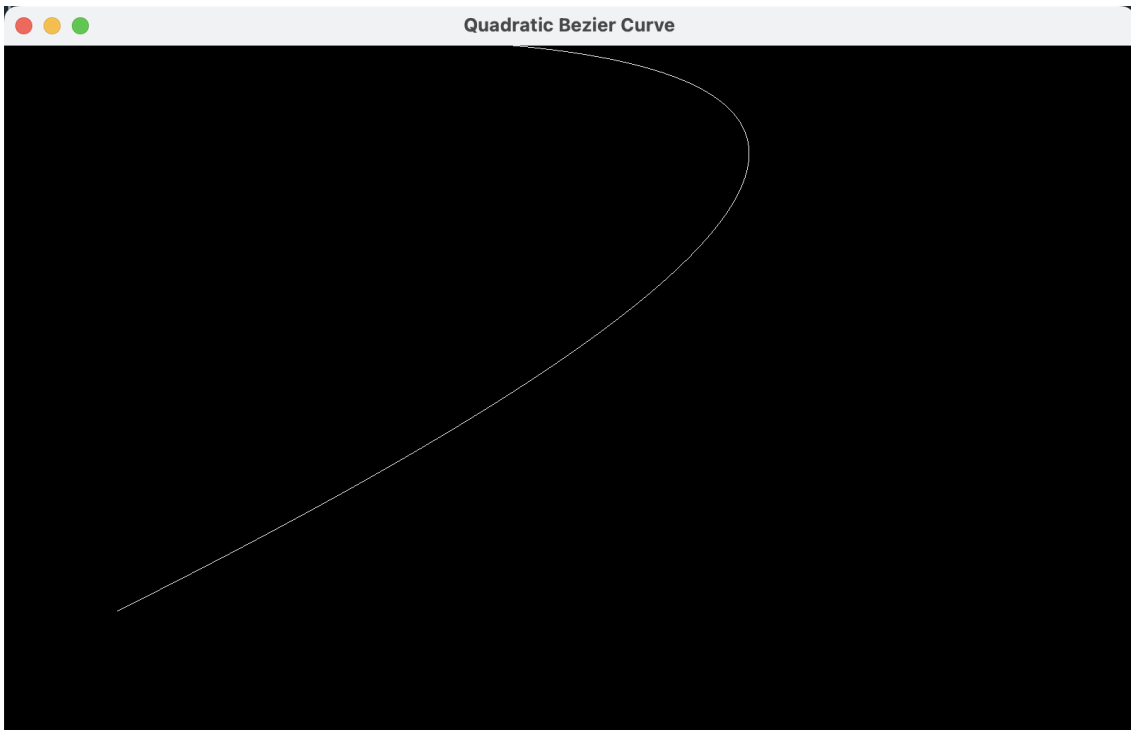
```
import glfw
from OpenGL.GL import *
from OpenGL.GLUT import *
import numpy as np

# Define the control points for the quadratic Bezier
curve
control_points = np.array([
[-0.1, 1.0],
[1.0, 0.9],
[-0.8, 0.0]
])

# Variable to control the number of segments in the
curve
num_segments = 100
```

```
def draw_quadratic_bezier_curve(control_points,
num_segments):
glColor3f(1.0, 1.0, 1.0)
glBegin(GL_LINE_STRIP)
for i in range(num_segments + 1):
t = i / float(num_segments)
p = (1 - t) ** 2 * control_points[0] + 2 * (1 - t) * t
* control_points[1] + t ** 2 * control_points[2] #
formula of beizer
glVertex2f(p[0], p[1])
glEnd()

def display():
glClear(GL_COLOR_BUFFER_BIT)
draw_quadratic_bezier_curve(control_points,
num_segments)
glFlush()

def main():
if not glfw.init():
return

window = glfw.create_window(800, 800, "Quadratic
Bezier Curve", None, None)

if not window:
glfw.terminate()
return

glfw.make_context_current(window)
```

```
glOrtho(-1, 1, -1, 1, -1, 1)
glClearColor(0.0, 0.0, 0.0, 1.0)

while not glfw.window_should_close(window):
glfw.poll_events()
display()
glfw.swap_buffers(window)

glfw.terminate()

if __name__ == "__main__":
main()
```


Quadratic Bézier Curve




Quadratic Bezier Curve