"La Sapienza" University of Rome
Faculty of information engineering, information technology and statistics
Department of informatics, automation and control engineering
"ANTONIO RUBERTI"
Degree program: Artificial Intelligence and Robotics

SAPIENZA
UNIVERSITÀ DI ROMA

PROJECT REPORT
ON
BUBBLE BUSTER GAME

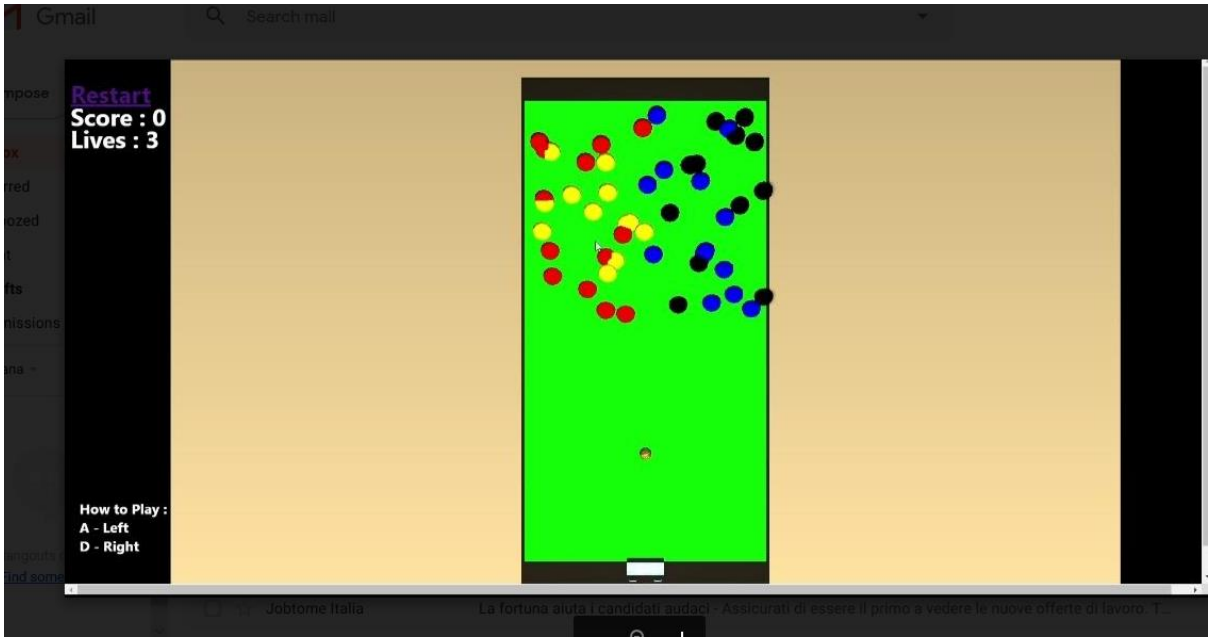
Student: 1
Nagothu Jeevana Bhagyasri
Matricola: 1847507
Medasana Poonam
Matricola:1874750
SUBJECT:INTERACTIVE GRAPHICS((2018-19)

Teacher: PROF.MARCO SCHAREF

INDEX

1.INTRODUCTION ABOUT THE PROJECT:

We are developing a game called **bubble buster** where our aim is to burst the bubbles with the ball by moving the paddle. Here, we have 3 lives and live score recording and once ball fails to hit the paddle it means the game is over! once the ball hits the paddle ,it automatically gets enlarged



2.REQUIREMENTS:

The player needs a web browser to run the game,the game has been tested on google chrome ,Microsoft edge and have been found to be successfully running without any errors.
The player can run the game in two ways:

- Open CMD and traversed in the path of the game folder.
- Start “python-m SimpleHTTPServer” in the shell(for python 3.0 and above type “python -m http.server” in your shell)
- Open browser and type:localhost:8080/

3.HOW TO PERFORM THE GAME:

Inorder to play the game, we have been developed two keys A and B where A is used to move the paddle LEFT and B is used to move the paddle to RIGHT. After paddle starts moving then ball starts its moving by passing through the center of the bubble .uless and until,it passes trhough the center of bubble,the ball doesn't starts its movement.it is nothing but based on physics law.

LIVES AND SCORECARD**:**

The simplicity of the game makes the scoring simple as well.the score will be increased as it blasts the each ball.and score will be updated at every level successfully completed.Every intial game has 3 lives and on loosing all 3 lives,the game is over and the score is published with a choice to start all over again.

4.DESCRIPTION OF THE ENVIRONMENT AND LIBRARIES USED IN THE PROJECT:

The following libraries which we have used to design the project:

THREE.MIIN.JS:-it is used to create and display animated 3D computer graphics in the web browser.

KEYBOARD.JS:-It is used to bind the keys to perform actions on the scene.

BUBBLEBLASTER.JS:-This is the 2D rendering game which we have created with the simulation of gravity, collision.

5.TECHNICAL ASPECTS OF THE PROJECT:

To buildup this game we used Three.js and bubbleblaster.js,The step by step implementation to start the game is described as follows:

CREATING THE TEXTURES OF THE OBJECTS:

```
var ballTexture = new THREE.ImageUtils.loadTexture("./img/ball/ball1.jpg")
var paddleTexture = new THREE.ImageUtils.loadTexture("./img/paddle/paddle1.jpg")
//var bubbleMaterial = [0xf6ff9f,0xffce96,0xf78686,0xf739a6,0xc031b5]
var bubbleMaterial = ["red" , "blue","yellow","black"]
var planeColor = 0x00FF00
var bubbleLocation = undefined

var bubbleLimit = 50,ballBurst = 50
```

CREATING THE RENDERING WORLD:

```
function createScene() {

    renderer = new THREE.WebGLRenderer()
    camera = new THREE.PerspectiveCamera(
        VIEW_ANGLE,
        ASPECT,
        NEAR,
        FAR
    )
```

//creating scene

```
 scene = new THREE.Scene()
//adding scene to the camera
scene.add(camera)
//camera position and rotation

camera.position.z = 480
camera.rotation.z = -Math.PI/2

//creating paddle material

var paddleMaterial = new THREE.MeshPhongMaterial({map:paddleTexture})

//creating planematerial

var planeMaterial = new THREE.MeshLambertMaterial({
color: planeColor
})

//creating tablematerial

var tableMaterial = new THREE.MeshLambertMaterial({
 color: 0x111111
})

//creating ground material

var groundMaterial = new THREE.MeshLambertMaterial({
color: 0x888888
})

var sphereMaterial = new THREE.MeshPhongMaterial({
 map: ballTexture
})
```

ADDING THE BALL

```
ball = new THREE.Mesh(
    new THREE.SphereGeometry(
        radius,
        segments,
        rings),
        sphereMaterial
)

scene.add(ball)
```

ADDING THE PADDLE

```
paddle = new THREE.Mesh(
    new THREE.CubeGeometry(
        paddleWidth,
        paddleHeight,
        paddleDepth,
        paddleQuality,
        paddleQuality,
        paddleQuality),
        paddleMaterial
)

// scene.add(paddle)
```

ADDING BUBBLES

```
addBubbles()

paddle.position.x = -fieldWidth
paddle.position.z = paddleDepth


var ground = new THREE.Mesh(
    new THREE.CubeGeometry(
        1000,
        1000,
        3,
        1,
        1,
        1),
        groundMaterial
)
```

CREATING THE WHEELS

//add wheels

```
function addwheels()
{

    var wheelRadius = 3.5;
    var wheelThickness = 0.8625;
    var wheelSegments = 10;
```

//wheel geometry

```
  var wheelGeometry = new THREE.CylinderGeometry(
      wheelRadius,     // top radius
      wheelRadius,     // bottom radius
      wheelThickness,  // height of cylinder
      wheelSegments);
```

//wheel material

```
var wheelMaterial = new THREE.MeshPhongMaterial({map:paddleTexture});
```

```
//creating wheel positions
```

```
 var wheelPositions = [
   [ -paddleHeight/3 ,(paddleWidth) ,  2 ],
   [ paddleHeight/3,-(paddleWidth),  2 ],
   [ paddleHeight/3,(paddleWidth) , 2],
   [ -paddleHeight/3,-(paddleWidth), 2],
  ];
```

//creating wheelmesh

```
  var wheelMeshes = wheelPositions.map((position) => {
    var mesh = new THREE.Mesh(wheelGeometry, wheelMaterial);
    mesh.position.set(...position);
    mesh.rotation.z = Math.PI * .5;
    mesh.castShadow = true;
    mesh.name="wheel"+count
    count+=1
    paddle.add(mesh);
    return mesh;
  });
```

//adding paddle to the scene

```
scene.add(paddle)
```

CREATING THE ANIMATION

```
function draw() {
    renderer.render(scene, camera)

    requestAnimationFrame(draw)
    playerPaddleMovement()
    paddlePhysics()
    console.log("Pause : ",pauseGame)
    if (pauseGame)
    {
    ballPhysics()
    ballHittingBubbles()
    }
}
```

6. DESCRIPTION OF INTERACTIVE OBJECTS:

This section describes the interaction of objects with the environment in the form of animation.
HIRARCHICAL MODELS:

We have created certain hierarchical models.the following parts are:

- Camera
- light
- Plane
- Ball
- Paddle
- Bubbles

Lights and Textures:

We have initialized pointlight and spotlight.The light is fixed at one position.
//adding pointlight to the screne

```
pointLight = new THREE.PointLight(0xF8D898)


//adding spotlight to the screne

spotLight = new THREE.SpotLight(0xF8D898)
```

we have used two textures named balltexture and paddle texture.
//Balltexture
we have loaded them from THREE.ImageUtils.loadTexture(function) and it takes image as its argument.
//Paddletexture

we have loaded them from THREE.ImageUtils.loadTexture(function) and it takes image as its argument.

PLAYER INTERACTION:
//Player Paddle movement

For the paddle movement we are using two keys A(LEFT) and B(RIGHT)

//Ball Movement

Ball starts its movement when it hits the paddle then it passes through the center of radius of bubbles

```
function playerPaddleMovement() {
    if(Key.isDown(Key.A) || Key.isDown(Key.R)) {
        if(paddle.position.y < fieldHeight * 0.45) {
            paddleDirY = paddleSpeed * 0.5
        } else {
            paddleDirY = 0
            paddle.scale.z += (10 - paddle.scale.z) * 0.2
        }
    pauseGame=true
    }
    else if(Key.isDown(Key.D) || Key.isDown(Key.L)) {
        if (paddle.position.y > -fieldHeight * 0.45) {
            paddleDirY = -paddleSpeed * 0.5
        } else {
            paddleDirY = 0
            paddle.scale.z += (10 - paddle.scale.z) * 0.2
        }
            pauseGame=true
    }
     else {
        paddleDirY = 0
    }

    paddle.scale.y += (1 - paddle.scale.y) * 0.2
    paddle.scale.z += (1 - paddle.scale.z) * 0.2
    paddle.position.y += paddleDirY


    rotateWheel(paddle)
```

//ballmovement:

```
var rot=0.1;
function ballPhysics() {
    if(ball.position.x <= -fieldWidth / 2) {
        score--
        document.getElementById("scores").innerHTML = "Scores: "+ score
        resetBall()
    }


    if(ball.position.y <= -fieldHeight / 2) {
        ballDirY = -ballDirY
    }

    if(ball.position.y >= fieldHeight / 2) {
        ballDirY = -ballDirY
    }

  if(ball.position.y >= fieldHeight / 2) {
      ballDirY = -ballDirY
  }
  if(ball.position.x >=fieldWidth/2)
      {
          ballDirX = -ballDirX
          translateBubble()
      }
  ball.position.x += ballDirX * ballSpeed
  ball.position.y += ballDirY * ballSpeed

  if(ballDirY > ballSpeed * 2) {
      ballDirY = ballSpeed * 2
  } else if(ballDirY < -ballSpeed * 2) {
      ballDirY = -ballSpeed * 2
  }



  if (rot>100)
      rot=0.01
  rot+=0.5
   ball.rotation.y = rot
   ball.rotation.z = rot

  //console.log(rot)
  }
```

Ball hits the paddle and due to gravity ,it bounces back and tries to blast the bubbles.

```
function ballHittingBubbles()
{

    x = ball.position.x
    y = ball.position.y
    //console.log("ballHittingBubbles")
    for(i=0;i<=(bubbleLimit*2);i+=2)
    {
    if((x<=(bubbleLocation[i]+radius) && x>=(bubbleLocation[i]-radius)   )
            &&
             (y<=(bubbleLocation[i+1]+radius) && y>=(bubbleLocation[i+1]-radius)))
                {
            scene.remove(scene.getObjectByName("Bubble"+i))
            bubbleLocation[i]=-fieldWidth
            bubbleLocation[i+1]=-fieldHeight
            score++
            document.getElementById("scores").innerHTML = "Scores: "+ score
            ballBurst -=1
            if (ballBurst==0)
                finishGame()
            //console.log(i)
        }

    }
}
```

//Paddle Movement:

Paddle should move left and right continuously inorder to save the life of the ball .if it fails to move then the ball will touches the ground and game will be over.

```
function paddlePhysics() {
    if(ball.position.x <= paddle.position.x + paddleWidth &&
        ball.position.x >= paddle.position.x) {
            if(ball.position.y <= paddle.position.y + paddleHeight / 2 &&
                ball.position.y >= paddle.position.y - paddleHeight / 2) {
                    if(ballDirX < 0) {
                        paddle.scale.y = 15
                        ballDirX = -ballDirX
                        ballDirY = paddleDirY * 0.7
                    }
                }
        }
}
```

WHEEL ROTATION:

For the movement of the paddle we are using four wheels which will rotate in the x-Axis.

//Rotate wheel

```
function rotateWheel(paddleObject)
{

    for(s=0;s<4;s++)
    {

        wheelObject = paddleObject.getObjectByName("wheel"+s)
        last_rot = wheelObject.rotation.x
        last_rot+=0.04
        if (last_rot>15)
            last_rot=0
        wheelObject.rotation.x=last_rot;
        }
    return;
}
```

RESTARTING THE GAME:

We have 3lives inorder to finish the game,in between these 3 lives,we need to blast all the bubbles on the screne and automatically scores will be recorded.

```
function resetBall() {
    ball.position.x = -fieldWidth/2 + paddleHeight
    ball.position.y = 0

    ballDirY = 1
    paddle.position.x = -fieldWidth / 2 + paddleWidth
    paddle.position.y=0
    pauseGame = false
    lives--
    document.getElementById("lives").innerHTML = "Lives : "+ lives
    if (lives ==0)
        finishGame()
}
```

END OF THE GAME:

.

```
function finishGame()
{

    alert("Game Finish, Please Click Restart to Play Again")

    while(scene.children.length > 0){
    scene.remove(scene.children[0]);
      document.getElementById("scores").innerHTML = "Scores:0 "
        document.getElementById("lives").innerHTML = "Lives:0"
}
   //setup()
}
```

Hence,The game will complete after the completion of all 3 lives and gives us the final score.