PROJECT REPORT

Topic: Bouncing Football

Group 8:

Bhavesh C: S20190010034

Khadyothan D: S20190010040

Gireesh C: S20190010036

Karthik K: S20190010100

PROJECT STATEMENT

Implement a Bouncing Football using OpenGL techniques.

ABSTRACT

Computer graphics and animations have in recent years become very common. Computer animation, or CGI animation, is the process used for generating animated images by using computer graphics. Computer-generated animations are more controllable than other more physically based processes.

Main aim of this Mini Project is to illustrate 3D Bouncing Football using OpenGL Computer graphics. The concepts of OpenGL glut library and python has been used to create bouncing ball. There is user interaction in this Computer Graphics program. It also includes how ball will rebound after collision.

INTRODUCTION

This program is implemented using various OpenGL functions and user defined functions. The bouncing ball effect can be achieved through various transformations, in our case translation. We use different transformations for this program some of the are

S.no	Types of transformations	Input	Parameters	Output
1	Translation	(x,y)	(tx,tx)	(x+tx,y+ty)
2	Scaling	(x,y)	(a, b)	(ax,by)
3	Rotation	(x,y)	θ	X=xcosθ +ysinθ Y=xsinθ -ycosθ

Using the above transformations and also other OpenGL functions to make the ball bounce.

METHODOLOGY

We will first initialize a 2D coordinate system In Opengl, once we have specified that we work with GL_MODELVIEW mode and load the identity matrix, we specify the translation parameters for the ball's center to move using glTranslatef function. It takes in 3 parameters, and moves the the current position by the amount specified by the parameters in their respective directions. To move the ball, we have actually moved the center of the ball and then constructed the ball every time we moved the center. To Construct the circle we have used the GL_TRIANGLE_FAN mode, which uses series of triangles with one common vertex to construct a circle, common vertex is the center of the circle. To Specify the other coordinates of the triangles, we use the parametric equation of the circle = (rcos(**θ**), rsin(**θ**)) First we decide the number of segments of triangles used to construct the circles and decide the vertices of these triangles using the parametric equation. To Make the ball look somewhat similar to a football, we tried to apply some texture to it. We used the

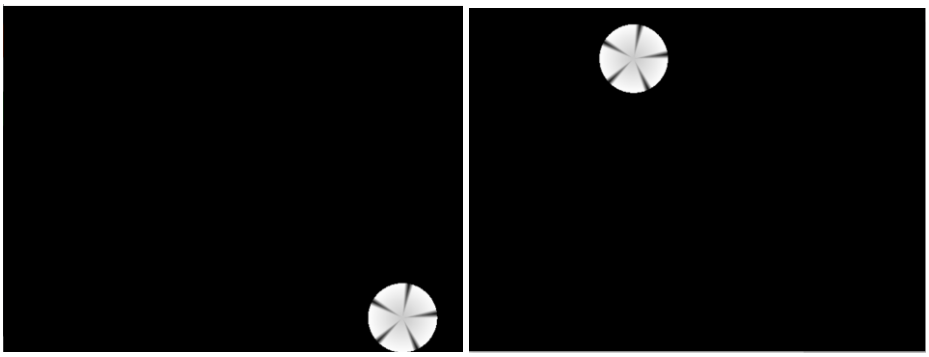
fact that since the circle is constructed by the triangles. We applied coloring in such a way that out of every ten triangles we have one triangle which is colored black while the others are colored black. We ended up with a texture that looks similar to a football.

WORKING

The radius and position of the ball are specified. We also define boundary restrictions, such as Xmax, Xmin, Ymax, Ymin, so that the ball moves only within those parameters. Every 30 milliseconds, we set a timer to redisplay the display function. We calculate the new centre coordinates for the following iteration at the conclusion of each iteration, and then render these coordinates when the timer ends and the display function is called again. We calculate the new centre coordinates for the following iteration at the conclusion of each iteration, and then render these coordinates when the timer ends and the display function is called again. The ball has speeds (xspeed, yspeed) that indicate how fast it is travelling along the respective axes. When the display function is called, we construct the circle using the centre and then calculate the new centre of the circle using (xspeed, yspeed); when the timer expires and the display function is called again, the circle is constructed using this new centre, and the same process is repeated, calculating the centre for the next iteration, and so on. Using the translation function, we specify the coordinates of the circular centre for each iteration. We build up the identity matrix first, then translate to the centre of the circle obtained in the previous iteration. The glTimerFunc() function in opengl is used to implement the timer functionality.

EXPERIMENTAL RESULT

When we execute the program in OpenGL, we can see the movement of the ball, how the ball bounces and collides with the wall and rebounds. We can also observe the ball repeats the path after n number of rotations where n is depends on the size of the box.



CONCLUSION

By using OpenGL, we successfully made the ball to bounce , rebound after collision using glTranslatef and many other functions mentioned in the code.