# CHAPTER 3

# SOFTWARE DESIGN

## 3.1 Overview

This project has been created using the OpenGL interface along with the GLUT( OpenGL Library Toolkit), using the platform Eclipse as the compiling tool. This game has been designed in a simple and lucid manner so that further developments can be made, and run on many platforms with a few changes in the code.

We begin our game by providing a simple menu option upon right click of the mouse to display 2 levels of the game which the user can choose, and a third option to exit the game.

Upon selecting the first level, a red ball is continuously produced and moves upwards at the right side of the screen. Hitting the 'f' key on the keyboard produces an arrow, which the user must aim to hit the balls with. On successful contact with the ball, the game registers it as a 'HIT' and increments a counter variable. A count of 3 (i.e 3 hits) takes the player to the next level of the game.

Level 2 increases the difficulty level by producing2 colours of balls- a blue one and a red one. While the red is as before produced continuously and moves from bottom to top, the blue one moves from top to bottom at irregular intervals to confuse the player. The player needs to aim correctly and target only the red ones. Player needs to again score a minimum of 3 'hits' to be a Winner.

Level 2 can be directly entered in the beginning itself ( without playing level 1). We provide this option so that the player can choose his difficulty level.

The underlying logic of the game is Collision detection. The bounds of the ball are detected with the bounds, or the continuously changing position of the arrow head. 2 variables 'pos' (for the arrowhead) and 'up' (for the ball) are kept tab of, and are updated with their changing values.

If the given position vector does intersect with the current value of the 'up', we infer that a collision has been detected and register it as a hit, and increment 'counter'. Other wise the function returns a NULL.

Once a 'hit' takes place, we call a Sleep() timer ( as buffers are too fast to see the collision in our game), show the ball burst ( without particle effect or engines) and reset the ball to its initial position and the cycle starts all over again.

Careful observation of these values have been made while developing the game.

To model the objects as 3D entities we also use the effects of lighting in our game via 4 different mechanisms:

AMBIENT - light that comes from all directions equally and is scattered in all directions equally by the polygon.

DIFFUSE - light that comes from a point source and hits surfaces with an intensity that depends on whether they face towards the light or away from it.

SPECULAR - light is reflected more in the manner of a mirror where most of the light bounces off in a particular direction defined by the surface shape.

EMISSION - the light is actually emitted by the polygon - equally in all directions.

Texture is yet another feature we have implemented to show the material properties of the object, in whichever small way we could