COL100 Assignment 8

Help Document - Simple Graphics - Rendering of Carrom Board Game

IInd Semester 2014-15

The assignment stub for assignment 8 uses OpenGL Utility Toolkit (GLUT) https://www.opengl.org/resources/libraries/glut/ for graphics.

When you build and run the given stub, you will get the following window.

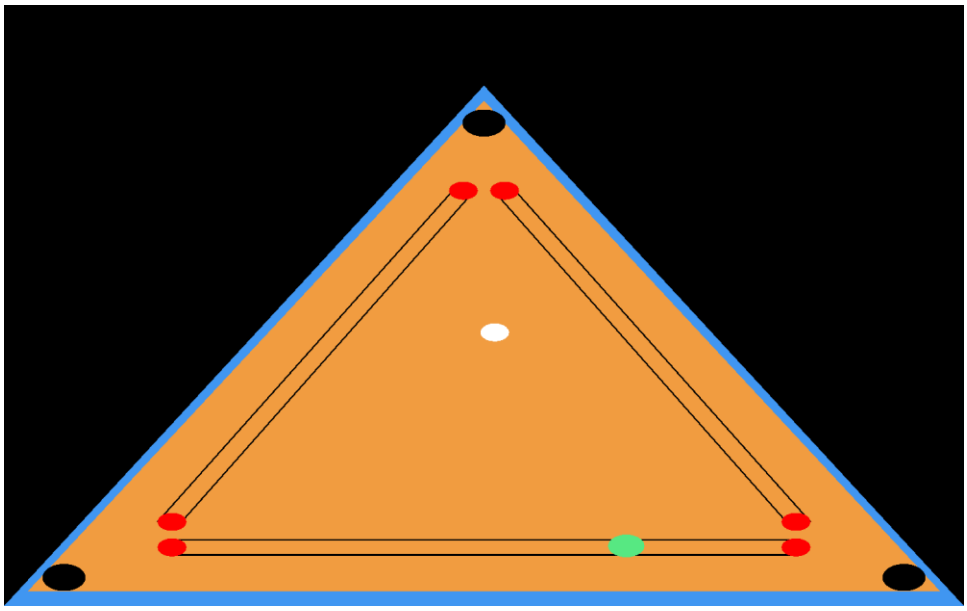

Figure 1: Screenshot of the carrom board (Press 'ESC' key to exit)

Structure of the program

The program has two logical sections: (i) game-simulation, and (ii) display. This is captured in the code using two key data structures (user-defined data types)

- ***gamestate_t*** - contains functionality to manage game-play simulation
- ***renderstate_t*** - contains functionality to display/render the functionality on screen.

gamestate_t::loopstep() contains the functionality that is performed per iteration of the game simulation. It contains 2 statements, update() and render_game(). You don't have to change anything in render_game() because it deals with displaying the game on screen.

gamestate_t::update() changes the state of the game board. The board has 2 carrom pieces 'striker' and 'coin' that are member variables of type *carrompiece_t*. Each piece in-turn has a position and velocity associated with it. Position of a piece is captured by it (x,y) co-ordinates, and velocity is captured by (dx,dy). (dx,dy) is the amount of change in (x,y) per iteration. We assume that the time taken per-iteration is constant, and hence is (dx,dy) is sufficient to capture the velocity. Simply changing the

position of a piece appropriately every iteration is sufficient to move a piece (as shown in the given example implementation of *gamestate_t::update()* function).

Co-ordinate system of the display

The co-ordinate system followed for rendering the carrom board is shown in Figure 2. The center of the screen is the origin (0,0) of the co-ordinate system. The outer vertices of the carrom board lie on (-1,-1), (1,-1) and (0,0.732). The width of the board (blue boundary) is 0.05. The co-ordinates of the pockets are (-0.875,0.905), (0.875.-0.905) and (0,0.607) and the radius is 0.045. The radius of the striker is 0.038, and the radius of the coin is 0.03.

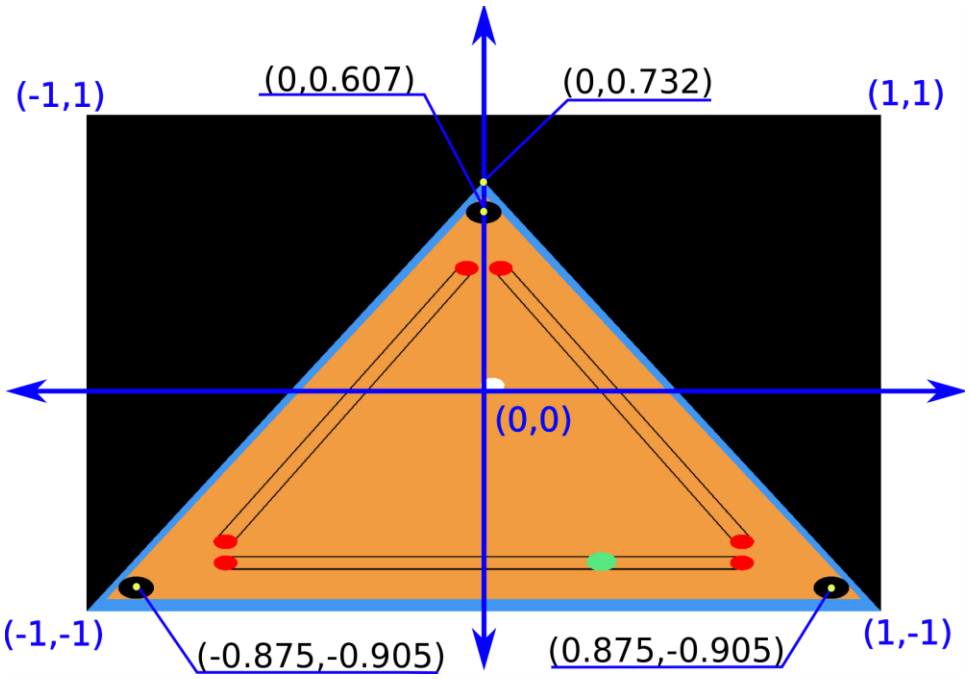

Figure 2: Co-ordinates of the carrom board

Permitted approximations in the simulation

The striker traces the displacement vector between itself and the coin. Therefore, assuming perfect collision, the striker will halt transferring its energy to the coin. You may set the flag 'is_visible' to *false* in order to make the striker disappear from the board. Using this you may avoid detecting further collisions between the striker and the coin.