CSE-251: Graphics - Spring 2015:

Assignment 1: A Carrom Game

Due: January 22, 7pm.

1 The Problem

The goal of the assignment is to make your own 2D game with keyboard and mouse controllers. This would be a 2D board game almost identical to the game of Carrom. You have a square playing surface with circular holes or pockets in the four corners. A larger disk called *striker* is used to hit smaller disks on the playing surface called *carrom men* or just *coins* (See Figure 1).

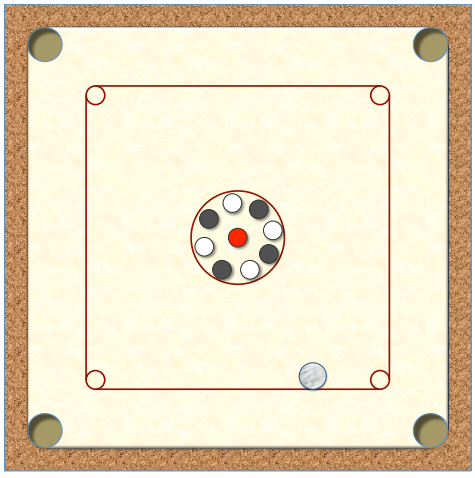

Figure 1: Layout of the carrom board and a possible set of coins.

A player plays either white or black coins. The goal of the game is *use the striker to hit and pocket all coins of your color and the red coin in as less time as possible*. Each coin of your color pocketed will get you 10 points and the red gets you 50 points. Each coin of the opposite color or the striker pocketed will cost you 5 points. The game starts with 30 points on the score board and each second will cost you one point. The current points should always be displayed on the game window.

In the following sections, the minimum requirements are mentioned. Let your imagination free and enhance the game as per your liking. It is *your* game.

You should provide a single page quick start guide to those who want to play the game (aka TAs), describing the additional controls (basic controls should be as described below), and additional features.

2 The World

The world consists of a square 2D ground plane with circular holes to its corners. The layout of the board is given in Figure 1. The striker is always on the lower line at the beginning of the play. The carrom men or coins, which are smaller in size than the striker, are inside the center circle. The size of the corner pockets are slightly larger than the striker.

The object should be filled polygons, and you may add color/style variations as per your choice. The ground plane and any other static entities in the scene may be created with a set of line segments or polygons.

3 Objects and Physics

The objects in the world were described before. You should have a single striker, a single red coin and an equal number of black and white coins (at least 3 each).

You should incorporate the basic laws of physics into the behaviour/motion of the objects. The minimum set of factors you need to consider are:

1. Laws of Motion: Every moving object continues to move in the same direction and velocity unless acted upon by a force. The primary forces are: friction and collision. All objects should come to rest if thay fall into a pocket or if their velocity v falls below a threshold that you determine (choose reasonably).

2. Frictional force: assume that the frictional force is same for all objects sliding on the ground and is in the direction opposite to the motion.

3. Collision: Two objects (coins or striker) will collide if their bounding circles touch or overlap. An object may also collide with the borders of the board if its bounding circle touches or overlaps the border line segment.

   Collision between a movable object and immovable object will result in the movable object rebounding as per laws of reflection, with a velocity $\alpha.v$, where v is the incident velocity and $\alpha < 1$ is a constant. When two movable objects collide, use the position of the centres of the objects to decide the directions of reflection, and the law of conservation of momentum to decide the velocities after collision. Assume that the mass of striker is twice that of any coin.

4. Pocketing: If the center of any object (striker or coin) moves inside the boundary of the pocket, it falls inside (and stops moving). Any coin that is pocketed is removed from the game, while the striker is placed back on the board for next play.

Make sure that you use the time elapsed since hitting the striker or collision or the time of rendering the last frame to decide the object positions. Do not assume that a fixed time has elapsed from the previous frame. Use a function like gettimeofday() to get the current time from system clock. Store this for the current frame so that when you draw the next frame, you know how much time has passed. (Question: What is wrong with assuming that a fixed amount of time has passed from last frame?)

4 Controls

There should be two sets of controls, one using the keyboard, and one using the mouse.

You should be able to move the striker to any position on the bottom line before striking. The striking involes giving an initial velocity to the striker. You should be able to control the direction and speed of the strike. Use the left and right arrows to move the striker on the base line. Use **a** and **c** keys to change the direction of strike either anti-clockwise or clockwise. Use the up and down arrow keys to increase or decrease the speed of strike and finally, the spacebar to initiate the strike. Give appropriate visual indications (say a line from the striker with changing direction and length) for the direction and speed of the strike.

One should also have mouse controls to achieve the above. You should be able to click and drag the striker along the bottom line using the right mouse button. Left click anywhere else on the board and the position where you click should determine the direction and speed of the strike. Release the left mouse button to initiate the strike.

5 Optional

You are free to add optional feature and embellishments to your world and the game. Additional object could include a fancy score board, a dock for pocketed coins or any other element by your imagination. Additional game options could include varying number and initial configuations of coins, different levels of difficulty in terms of rules (say hitting the opposite colors costs points), additional controls, sounds, textures, etc. Note that a small change to the the board and rules and adding a cue stick would make it a pool/billiards game.

6 Useful Hints

1. When you create objects in the world, make them as objects in the program (class, struct) with all the parameters needed for drawing, animating, motion, sound etc. built into the object along with a method to draw themselves. These parameters would include, but not limited to position, orientation, color, state, velocity, etc. This way, you will be able to replicate an object easily and enhance them later on.

2. The main control logic (or game engine) will look at the current state (time, collisions, etc.) and update each element in the world. Create a collision detection function that checks for collision between any two objects. This may be used by the game engine to find any impact and take corresponding action.

3. Start with a simple world and complete the game. You may enhance the objects/motion, once you are done with your V1.0. While creating the objects and

the game engine, think of how to make each parameter that you set to be flexible. As an example, if you design the friction to be a variable, you can play the game on different surfaces.

4. Create objects for UI elements such as scoreboard with methods to draw themselves.

7 Submission

You submissions should include your source code, a makefile and a compiled executable. You need to include a readme file that describes any additional information that is needed in compiling/executing you code. Do not use any libraries other than OpenGL and glut. In addition to these, include a file named help.txt or help.pdf (no word or other proprietary formats) in the submission that gives a one page description of the game and how to play it.

Details of how to submit and any modification to the above submission details will be posted by the TAs towards the submission deadline.

8 Grading

You will be graded based on the correctness and efficiency (speed) of the implementation of the minimum elements described above. This will contribute to 90% of your grade. Remaining 10% will be given based on the improvements that you do over the basic game. In addition, submissions that are found to be exceptional by the graders, will be showcased, and will be awarded extra credits up to 10%.