CSE 251: Graphics - Spring 2017:

Assignment 1: Brick Breaker
Drawing in 2D with OpenGL 3.0

Deadline: January 17, 5:00 PM.

1 The Problem

The goal of this assignment is to get yourselves familiarized with OpenGL 3.0 as well as to brush up your linear algebra basics. This assignment will let you develop your own 2D game with keyboard and mouse controllers. This would be a 2D laser shooting game , where you fire laser beams from a canon on the left to targets that are randomly dropping from the top of screen. There should be some mirrors placed in the path where laser rays can be reflected. The player can control the angle and vertical position and firing of the canon and gets points for shooting the target (black bricks). Any mis-targeting would cost negative points. Additionally, on the bottom of the screen there should be two movable baskets to collect color specific bricks. The game is over if any of the black brick is collected in any of these two baskets at bottom or more than a finite number of non-black bricks are targeted by the player. Define your own rules for earning points and penalties for mis-firing. The points should always be displayed on the top right of the game window.

In the following sections, the minimum requirements are mentioned. Use your imagination 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 is 2D and contains a rectangular base at the near (lower) end with two baskets. A set of black, red and green bricks are falling from random positions at the far (top) end. You have a laser cannon that can shoot a short laser ray in the direction pointed by the cannon (see Figure 1). You can move the canon up & down on vertical axis and also control it's tilt. The game should display the final score when it gets over.

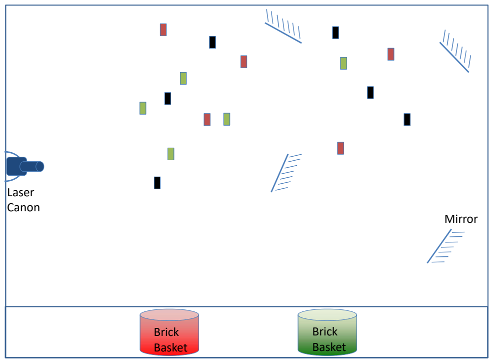

Figure 1: Sample *Brick Breaker* with few bricks and the cannon shooting a black brick. The picture given above is indicative and your models of objects may look different.

Your goal is to capture as many red and green bricks in the basket as possible and shoot down as many black bricks as possible. However, if any black brick is collected in the red or green basket then game is over.

You should be able to control the baskets and the cannon by using the mouse and the keyboard. Details of the control are given below. Specifications of the world should be read in when you start your program.

3 Objects and Physics

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. Mirror Reflection: Laser rays's follow the standard optics for their reflection behavior.

2. Laser Firing Rate: Once you shoot a laser ray, the canon takes one second to recharge, and then only you can shoot again.

3. On World Boundary: Laser rays are absorbed by the boundary wall of the 2D world.

4 Controls

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

You should be able to tilt the canon upward and downward using the keys (a and d) and move it up and down using the keys (s and f). Use the keys (n and m) to increase or decrease the speed of bricks falling (within the permissible limits). Left and right movement of baskets can be controlled with Ctr+left and Ctr+right (for the red basket) and Alt+left and Alt+right (for the blue basket). The left/right arrow keys should be used to pan the scene and the up/down keys to zoom in and out respectively. One should be able to shoot with the space bar.

One should also have mouse controls to achieve the above. You should also be able to select any movable object by clicking on or near it (highlight the selected object). Then you can move baskets left or right and canon up and down by dragging. Use the position where you click to decide the direction of the shot. Use the mouse scroll wheel to zoom in and out. Use the right mouse button to pan left/right when you click and drag.

5 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, 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.

4. Create objects for UI elements such as speed, score, and life indicators and with methods to draw themselves.

6 Optional

Display a running scoreboard at the top right corner of the world. Another element that might add more interest and difficulty is the presence of moving obstacles that can deflect laser beams. Including audio could make the game more interesting, and so will improving quality of the object model. You may also add any additional item that you feel would make the game more interesting. Make sure that you explicitly list out the additional factors that you have included, their behavior, and how to control it, in your game in the *Readme* file that you upload.

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 non-standard libraries. 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%.