CSE 251: Graphics - Spring 2016:

Assignment 1: Your Own 2D Game

Due: January 24, 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 projectile shooting game ($\grave{a}$ la angry birds), where you send projectiles from a canon on the left to targets to the right on the screen, either on the ground or positions above. There should be obstacles on the path between the canon and the targets. The player can control the angle and velocity of shooting and gets points based on the difficulty of the target and the number of tries. Define your own rules for earning points, amount of ammunition, number of chances, etc. 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 consists of a 2D ground plane towards the bottom, with a canon on its left. There are objects placed at various positions in the scene to the right, that need to be shot down using canon balls that you fire.

The objects 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.

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. Laws of Motion: Each moving object continues in the same direction and velocity unless acted upon by an external force. The primary forces are: gravity, friction and collision. All objects should come to rest on the ground, if their velocity v falls below a threshold that you determine (choose reasonably).
2. Gravity: Every free moving object is acted upon by a downward force, which causes a constant downward acceleration of k units/sec.
3. Friction: There are two types of friction forces: i) Air friction, which causes a deceleration proportional to the speed in the direction opposite to the objects velocity; ii) Ground friction, which does the same for any object rolling/sliding on the ground in the direction opposite to the motion.
4. Collision: Define a bounding circle (centre and radius) for each object that you create. Two objects will collide if their bounding circles touch or overlap. An object and an element of the world (ground or any other immovable entity) will collide if the bounding circle of the object overlaps any of the line segments or curves that constitute that element.

   Collision between a moving 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 $0 < \alpha < 1$ is the bounce co-efficient. When two movable objects collide, use the center positions of the objects to decide the directions of reflection, and the law of conservation of momentum (velocity if you assume all objects of equal mass) to decide the velocities after collision.

Use the time elapsed since firing a shot 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.

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 above(a) and below (b) using the keys mentioned. Use the faster (f) and slower (s) keys to increase or decrease the speed of firing. 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. Use the position where you click to decide the direction and speed 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 Optional

You are free to add optional features to the world (moving/animated objects, artistic flares, textures, etc.), motion models (bouncing for other objects, spin, etc.), and controls.

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 acceleration due to gravity to be a variable, you can play the game on earth or moon. If you a variable for air resistance, you can play your game under water.

4. Create objects for UI elements such as speed, score, and life indicators and 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 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%.