Homework Assignment 6 COMP53 Fall 2012
due Dec 7, 11:55 PM

**Programming Project: Interactive 2D Game Programming**

**Purpose of Assignment**: This project is designed to:

- give you experience dealing with larger programs,
- expose you to the event driven programming and basic issues associated with real-time interactive programs,
- introduce you to 2D graphics, and
- utilize data structures to control dynamic objects.

**Summary of Assignment**: Breakout is an arcade game in which a player controls a paddle to direct a ball into a formation of bricks. The basic goal is to break all the bricks by hitting them with the ball.

You will be given the basic skeleton of a Breakout game. You need to turn it into a playable game and expand it to make it more interesting and more fun.

**Code Structure and Development:** You're given C++ code containing the basic framework of the game, built on top of the InterAct2D library. The initial project has static bricks, a moving ball, a keyboard controllable paddle and a game clock. It handles collisions between the ball and the walls (window boundaries) as well as collisions between the ball and the paddle.

The new version of InterAct2D (ver 2.0) uses OpenGL and GLUT, so it is no longer dependent on Windows or Visual Studio. Additional documents are included with the assignment to describe how to build OpenGL/GLUT projects and how to use InterAct2D.

You are free to modify any of the code that implements the game (Ball, Brick, Clock, Paddle or Breakout classes), but do not modify the InterAct2D code (**`InterAct2D.h`** or **`InterAct2D.cpp`**). If you do modify InterAct2D, you will have problems if corrections or updates need to be release later.

**Assignment Details:** This assignment is worth 20 points. This is twice the weight of the previous programming projects.

There are three required tasks totaling 14 points. You can then select from the optional tasks to earn the remaining points. You can earn a maximum of 22 points (10% extra credit) by correctly implementing optional tasks. The required tasks must be completed before extra credit points can be earned.

Required Tasks:

Task 1 [4 points] Break the Bricks:
Complete the collision detection between the ball and the bricks, so that bricks are removed once hit. This code will need to expand on the code already given in **`Breakout::detectCollisions()`**.

Task 2 [6 points] Additional Types of Bricks:
Create at least two specialized types of bricks. All new brick types must be implemented as subclasses of the existing **`Brick`** class. The specialized behavior of your new bricks can be anything that you like, such as unbreakable bricks, bricks that require multiple hits to be destroyed or bricks that contain and release other objects. Brick types should be visually distinguished by different appearances (color, size, shape, etc.). (Bricks that release other objects would be useful for the power-up and multi-ball optional tasks.)

Task 3 [4 points] Scoring:
Add a score counter and score display. The details of the scoring algorithm can be of your own design, but might include number of bricks destroyed and time used. Use the existing **`Clock`** class as an example on how to implement the score display.

Optional tasks:

Task 4 [2 points] End the Game:
Terminate the game when the clock runs out. Give the player a status message and allow them to restart the game using the RESTART button (which is already implemented by InterAct2D).

Task 5 [4 points] Power-ups:
Add power-ups that fall downward when released from specific broken bricks. Power-ups are activated by catching them with the paddle. Power-ups might change the paddle size, give additional time, change ball speed or size, or add a gun to the paddle. You must add at least two different power-ups to complete this task.

Task 6 [2 points] Lives and Death:
Add a limited number of player lives, and some mechanism by which the player can die. Death generally happens if the ball strikes the bottom of the screen. This should include some visual indication of remaining life and some means of terminating the game if all lives are exhausted.

Task 7 [4 points] Multi-ball:
Add some mechanism for generating and controlling additional balls. This might occur as the result of power-ups or they might be released directly by certain broken bricks. Extra balls should exhibit the same basic behaviors as the primary ball.

Task 8 [2 points] High-Scores:
Record and display a table of high scores. This must include some means of allowing the player to enter their name or initials into the table. The table must be persistent.

Task 9 [2 points] Collision Response:
Improve the reaction of the ball when striking bricks. The current code only causes the ball to bounce downward when hitting a brick from below. Allow the ball to bounce appropriately when striking the sides and top of the bricks as well.

**Submitting your project:** Submit all code for your project to Sakai as a single archive file (.zip or .tar). Submit only C++ code (.cpp and .h files). Be sure that your name is at the top of every file that you have written or modified.

Include a document with the following information:

- The programming environment used to develop your code (compiler/IDE and operating system).
- A basic user's manual with instructions on how to play your game, with an emphasis on the unique features that you have added to the game.
- A list of tasks completed, with an indication of where the code to implement that task can be found (class names and method names should be sufficient).

**Teams:** You may choose to work solo or as a team of two. If you work as a team, you will earn full points for the three required tasks, but optional tasks will only earn half the designated points.