CSC 341/CSC630:Fall 2011
Assignment 2: A 2D Video Game

The program must be submitted to me (swei@sju.edu) by Tuesday, October 25. Subject of the email should say "CSC341/CSC630 Assignment 2". Name your Project directory as "YournameAssg2". Before submitting, zip your Project directory, and email a single zipped file as attachment. Remember that your code should be fully documented. Check the course syllabus for late policy.

Overview:

In this assignment you will implement a video game as described below. You are allowed to implement extra features, as long as you provide sufficient documentation in your README file of how to play the game (Provide a README file with your submission even if you do not add any extra features.)

We will think of the game as taking place in three dimensional space, but the graphics will be drawn in the plane. We have a set of objects, called things. Each thing is drawn as a colored square on the (x,y) plane (but you may design the things however you like). The things move around within a large square window with constant speed. Whenever a thing hits the side of the square it bounces off, while maintaining its speed. For example, if it hits a vertical wall, then the y-component of its velocity is unchanged and the x-component is negated. Initially the things start with random locations and random velocities.

Each thing floats on a certain level, which can be thought of as its z-coordinate. There are 5 different levels for z = 0, 1, 2, 3, 4. Each level contains 4 things initially. There is also one additional level z = 5 called the ground level. On levels 0, 1, 2, live the good things (drawn in yellow) and on levels 3 and 4 live the evil things (drawn in purple).

The objective of the game is to destroy all the evil things (for which you gain points) while destroying as few good things as possible (for which you lose points). To tell differences in level, the darker colors are for lower levels. Things that are at the same level have the same color.

You destroy things by dropping bombs on them. This is done by clicking the left mouse button at the position that you want the bomb to start. A bomb starts at level 0, and slowly falls to each of the other levels until it hits the ground level, where it disappears. A bomb is drawn as a smaller square (or any design that you may have). To indicate the present level of the bomb, its color changes (to match the color of the level) as it falls. When a bomb at level i hits a thing that lives at level i (meaning their squares overlap) the thing is hit and changes color to black to indicate this. About a second later, the thing disappears (meaning that it is destroyed). After a bomb hits a thing, the bomb continues to fall. Score and other status information should be written onto the screen.

The game is terminated either when the user press the ’q’ key, or when all the evil things are destroyed. The game can be paused by pressing the middle mouse button, and the game can be single-stepped by pressing the right mouse button.

Input Events:

- When the ’q’ key is stuck, the game is terminated.

- Left button down: This causes a bomb to be dropped centered at the current mouse coordinates. There is no limit on the number of bombs that the user is allowed to have active at any time.

- Middle button down: When the middle mouse button is pressed, the game pauses. When the middle button is pressed a second time the game resumes.

- Right button down: When the right button is pressed, the game pauses if it is in a running state. (If it is already in a paused state or if the game is over, it stays in its current state.) The state of the game is advanced by one step, and then the program outputs all the game’s current state to the standard output (for debugging purposes). This should include the locations of the things and the bombs and any other information that you find important. By repeatedly hitting the right button the program will perform consecutive single steps.

- Window Resize: If the window is resized, then the program should do something reasonable. In particular the general look, feel and difficulty level of the game should not change significantly. For example, if the window is made larger, object sizes and velocities should be scaled proportionally. It is not necessary that they remain as squares.

Implementation Suggestions:

The exact specification of how the game looks is left to your creativity. Here are some of the parameters that you can use. Sizes and velocities in the x and y directions are all specified in an abstract quantity called units, which you may adjust. Depth units (z-values) are also in units. You can modify them as the game is too fast or too slow.

- Unit size: 400 pixels.

- Window size in units: 1 x 1.

- Thing width: 0.1 units.

- Thing velocities: uniformly distributed with max value of 0.25 units per second for each coordinate.

- Bomb width: 0.04 units.

- Bomb speed: 1 level per second.
- Points per hit: Level 0: -4000, Level 1: -2000, Level 2:-1000, Level 3: +1000, Level 4: +2000.
- Number of things: 5 levels with 4 things per level.

Grading Policy:

Grading will be based on the following points. Try to implement and test one feature fully before proceeding with the next feature, so that if you do not complete the project you still get partial credit.

- Correct thing movement, pause and single-step: 40
- Correct bomb movement and color: 25
- Destroying action, correct scoring, display of score and final result: 25
- Documentation and clarity: 10

Extra Credit Features:

- Animation of destruction: In the basic version, when a thing is destroyed it just disappears. Instead make its explosion more realistic. You may do this as you like.
- Better depth hints: Give better visual hints about depth. For example you can add shadows to add a sense of depth. Also rather than changing bomb color instantly from one level to the next, change colors gradually between levels.
- Clock: You may add a clock. The game may be terminated when the time is over.
- Feel free to add other features for extra credit.