CS312:Fall 2016

Assignment 2: 2D Game

Due Tuesday 9/20

Overview: In this assignment you will implement a video game as described below. You are free to implement extra features or change the original specification, as long as you provide sufficient documentation in your README file on how to play the game. A version of what you will implement can be found here (`http://www.pacxon4u.com/space-invaders/`).

We are implementing the now ancient video game called Space Invaders/Galaxian. You control a spaceship (the *ship*) that sits at the bottom of the screen. You can only move the ship left and right. It does not move in the y direction.

Towards(Above) the ship is an array of alien spaceships (the *aliens*). The aliens drop bombs down from time to time. If the ship is hit by a bomb, then you lose a life/the game. The aliens move as a block in a zig-zag motion from top to bottom and get faster as they get closer to the bottom. If any alien reaches the bottom of the screen then you lose even if you manage to stay alive. Your ship can fire missiles up at the aliens. Your ship always fires straight up (unless you are doing extra credit and implement the rotating gun). As soon as the missile hits an alien ship, the alien ship and the missile are both destroyed. When all aliens are destroyed you win (or get more aliens). There is no limit on the number of missiles that can be fired.

Features/Requirements:

- Mouse movement/arrow keys: When the mouse moves, the x-coordinate of the mouse is read. It is scaled to the range [-1,+1] (where 0 should correspond to the center of the screen) and multiplied by a constant to determine the velocity of the ship. The y-coordinate is ignored.
- Left button down/space key: When the left mouse button is pressed, the ship fires a missile upwards. Missiles move at a constant velocity.
- Right button down/'d' key: When the right button is pressed, the game pauses (if it is already paused it stays that way) and 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 ship, aliens, missiles, and bombs and any other information that you find important. By repeatedly hitting the right button/'d' the program will perform consecutive single steps.
- 'r' key: When the 'r' key is pressed, the game resumes if it is paused.
- 'q'/'escape': When the 'q'/'escape' key is hit, the program quits.
- Alien animiation: There should be some basic animation of the aliens as they move towards your ship. For example, if you only have squares, then they should spin once in a while.

- Death animation: When an alien is hit, it should spin then disappear as it shrinks in size. More elaborate death animation can be implemented as extra credit.

The exact specifications of how the ship, aliens, missiles and bombs look are left to your creativity.

Implementation Suggestions:

The game simulation can be advanced by one step for each timer interval. Each object (ship, alien, bomb and missile) is associated with a current velocity and moved by this velocity at each step. The aliens cannot move outside the set x-y area, and when the first alien hits the side, they should all change directions at once. If a bomb or missile leaves the area, then it ceases to exist.

To test whether a missile or a bomb hits an object, you can simply test whether the trajectory of the bomb or missle intersect with the bounding sphere of the object. This is approximately correct but fast. When a missile hits an alien, both the alien and the missile disappear (after death animation). When the game is over, the image pauses and the program waits for the user to hit the 'q' key to terminate the program.

Start with really easy shapes for the ships and bombs, such as squares and circles. Get the game play to work smoothly first before you spend time on making it look pretty.

Tuning the game play is an art. You don't want to make the game too difficult or too easy, neither is fun. This includes how many bombs/missles, how fast can you fire, the speed of the aliens, the amount of shielding (if you are implementing extra credit), how many extra lives, etc.

In general, text rendering in OpenGL is an advanced topic with no easy solutions at this stage. Either keep scoring and display in the console window instead, or forgo scoring entirely. Some game information can be commnunicated without text, such as extra lives can be simply shown as additional stationary copies of your ship somewhere on screen.

Extra Credit Features:

- Advanced model design: Design nicer polygonal models for the ship, alien ships, bombs, etc. There are many free downloadable models on the internet, but the formats vary and it might be work to convert them. You can design your own either by hand-drawing (if you have the artistic inclination), or employ tools.
- Shields: The green walls that your ship hides under. It changes the game dynamics completely and adds data structure/algorithm design challenges.
- Rotating gun: Implement a freely rotating gun on your ship so that you can shoot in any direction regardless of your heading.
- Diving alien: In classic Galaxian an alien can leave its ranks at random and dive down on the ship. You can implement this.
- Feel free to add other features for extra credit.

Live Demos: After the deadline, you will each be asked to demo your game. Past experience tells me that I simply do not have the hand-eye coordination to play some of the fast-action games that you guys can come up with! The demos will be short (3-5 min), so please come prepared to show off all your program features.