15-440 Lab 2

10-Sept-08

1 Introduction

In this lab you and a partner will design, create, test, and demonstrate a game. The game will be in java, using the java2d package. It is a large project and will proceed in stages, as outlined below. You are expected to write your own code, although you may discuss ideas with your classmates. Code from previous projects will need permission of the instructor. Your game will progress from single player on a single machine to a final version of a multiplayer game across a network. You should design your game with the end & final configuration in mind.

2 Important Dates

- Proposal Due: Fri 12-Sept-2008
- Proposal Returned: Mon 15-Sept-2008
- Checkpoint 1: 24-Sept-2008: Demo single player on single machine
- Checkpoint 2: 1-Oct-2008: Demo multiplayer across network
- Lab Due: 8-Oct-2008

3 Proposal Writeup

Your proposal should outline at a high level how your game will meet the game requirements. It should also outline the important internal data structures, locking mechanisms, and control flow for the game (the 'guts' if you will), both on the client side and server side. Your proposal must also include a (large) section about how you intend to test and debug your game. Please do not forget to include the name and andrew id of both partners. Your game proposal should be typed/typeset. If we cannot read it, you do not get credit for it. As a guide, the proposal should be roughly 2 pages in length. Content, not length is what is important to us, and hence to you.

4 Game Requirements

These are the minimum requirements for the game. You are, of course, encouraged to add features; the staff likes whistles and bells, but only if everything else functions.

4.1 Environment, Game Board

The game will take place in some sort of two-dimensional world or board. Players have an associated position and the ability to move around the world. There must also be obstacles and items on the board which impede or affect the player in some way; in other words, your code will need to perform collision detection between players and other objects in the world.

4.2 Players

For the first few checkpoints, your game will be single-player. The players must be able to move, perform actions, and have an effect on the game world; they will also have at least some value (s̈core, ïtems, etc.) associated with them.

4.3 Goals

There must be a goal for the game, such as to be the first or last player to achieve some end, or to have the highest score after some time period. The game also must have at least three distinct stages; these can be different levels or boards, staged difficulty settings, new ”enemies”, or some such.

4.4 Objects

Each level should have at least five different objects of three distinct types. Objects are game pieces which have some impact on a player; they could, for example, add a score bonus or provide some special ability to the player. You may have objects which impede the player as well, but at least one must have a positive effect. Objects are be picked up by players who reach the object on the game board. At least one type of object must be shared among players in some form, as well; for instance, one player may elect to give another player an object, or cause it to affect the other player.

4.5 Menu

Your game should have a menu, which displays: - The name of the game and its authors
- Rules
- Options to start a new game, or exit.

4.6 Network Code

When you start making your game multiplayer across a network, you must make your code robust to malicious clients and servers. A malicious client tries to damage or exploit the game server running the game, and a malicious server is a server that tries to compromise a client running on a player's machine. Since the game lab is written in Java, buffer overflow exploits and the like are less of a concern. You must still be careful to avoid ”cheating” clients, and invalid network communications should not cause a client or server to crash with an exception or loop infinitely.

4.7 Concurrency

The game needs to include, as a matter of design, situations that involve concurrency control – both in the multiplayer version and also in the network version. They might also occur in the single player version, depending on how you structure your game. In the network version, you will need to, at the very least, maintain a consistent view of the world for all players. Please keep this in mind as you design your game to ensure that your game will provide you with these learning opportunities.

5 Grading

5.1 Checkpoint Demos

You and your partner will be expected to demo your game in its current state during recitation the week a demo is due. Both partners should be present for each demo. If you are unable to make a demo, please make arrangements with the course staff. Demos are part of your grade. It is very important to stay on track with the demos. Otherwise you are in danger of being unable to complete the lab.

5.2 Grade Breakdown

Points will be assigned in the manner below. Extra credit may be given for interesting game features at the discretion of the course staff.
20 % code quality, style
55 % meets functionality and robustness requirement
10 % playability, funness, creativity
05 % checkpoint performance
10 % proposal writeup

5.3 Technical Requirements

- Your game must function on the Andrew Macintosh computers.
- Your code must compile properly in Java 1.5.

6 Nota Bene

- Set up and use version control. Now.
- Settle any major style wars with your partner before starting to write code.
Inconsistent code is painful to read and will be reflected in your grade.
- Document your code (and its bugs) as you write it. Do not put documentation off until the last minute.
- Think/write test code for your game code concurrently with developing your game. How will you know it works?
- The web is full of excellent java 2d tutorials. Make use of them.
- Set aside time each week to meet with your partner and work together for a minimum of 6 hours a week.
- Remember to go to office hours sooner rather than later.