COM374 Coursework 2: Multithreaded Games

1. Aims

To develop a Java implementation of a multithread game. The coursework involves GUI programming, multithreading, graph and multimedia.

2. Objectives

On completion of this coursework you should be able to:

- construct a user-friendly GUI using various components and layout managers
- create your own threads
- implement the main features of event delegation model
- understand how to play sound in Java
- create animation in Java
- understand how to find the shortest path in a graph

3. Tasks

You are required to develop a multithreaded version of Pacman Game. In this game, the player controls Pacman around a maze eating pills and avoiding ghosts. The player scores points when a pill is eaten and once all the pills are eaten the players moves to the next level. When a ghost touches Pacman the player loses a life. In the corner of the maze there are special power-pills that when eaten allow Pacman for a limited period to eat the ghosts and not die. When Pacman loses all of his three lives, the game is over.

Useful links:

- Wikipedia entry on Pacman
- On-line version of Pacman - so you know how the game goes

Basic requirements

- The basic requirement for this assignment is that Pacman and each of the ghosts **must** each have their own thread used to move them on screen.

- An example of how to implement a maze can be found on WebCT under Assignment/ /Assignment2/ folder. If you want to implement your own version of the maze, the following encoding scheme **must be followed**.

    - 'e' – the ghost box exit
    - 'h' – horizontal line
    - 'v' – vertical line
    - '1' – northeast corner
    - '2' – northwest corner
    - '3' – southeast corner

- ‘4’ – southwest corner
- ‘o’ – empty navigable cell
- ‘d’ – navigable cell with pills
- ‘p’ – navigable cell with power pellet
- ‘x’ – empty non-navigable cell
- ‘g’ – ghost box

Basic features

1) GUI

- Allow users to start/pause/resume the game by a keyboard and a mouse
- Display the current player's score at the top and the number of Pacman's lives left at the lower-left corner
- Display a game over screen, which allows users to start a new game or exit.

2) Control

- **Pacman moves under player control** (using keyboard). Pacman cannot go into the ghost box. When a Pacman eats a pill (including power pellet), that dot disappears.
- **Ghosts move under computer control**. They move in one compass direction (N, S, E or W) until they reach an intersection, at which point they can turn left or right. They will change to another compass direction if they hit obstruction/walls. A ghost does not allow to return to the ghost box unless it is eaten while vulnerable.

3) Animations: All characters are animated when they are moving. For example, Pacman's mouth opens and shuts. Ghost eyes move.

4) Sound effects and music – The opening tune, eating the pills, Pacman dying, etc.

5) Game rules

- Pacman can eat pills and scores points for player (each pill is worth 10 points and each power pellet worth 50 points)
- Ghosts kill Pacman when they collide with Pacman. Pacman starts with three lives. When Pacman loses all of his lives, the game is over.
- Power pellets that allow Pacman to eat the ghosts (the ghosts become vulnerable and turn blue).

Advanced features

1) Extra levels to the game: when a pacman eats all the pills and power pellets, go on to a next level.
2) High-score table: keep track of the top 5 scores and save them to a file. The top 5 scores will need to be displayed at the end of a game.
3) Intelligence for the ghost behaviour. You will need your implementation of Dijkstra’s algorithm developed in Week 9 to help the ghosts capture Pacman.(Hints: You can use an undirected graph to

represent the navigable areas of the gameboard, in which nodes stand for navigable cells and edges connect adjacent pairs of cells. Dijkstra's shortest-path algorithm can then be used to determine the shortest path from each ghost to Pacman.)

4) Any additional features not mentioned above – to exercise your creativity, you are welcomed to add any enhancement to the game.

Documentation: including a list of features you have implemented and your testing results. For each implemented feature such as animation, thread implementation, collision detection, etc, you need to explain its functionality with sufficient details.

Hints: (1) You need to work out how to detect when PacMan and a ghost collide. You can look at the Shape hierarchy to see if anything there can help (make life simple - assume ghosts and Pacman are rectangular) (2) Have a look at Weeks 9 to 10 lab exercises and the sample code provided before you start the thing.

4. Resources

1) Cell.java – This class is used to construct a maze which can be represented as a 2-dimensional array. Each cell in the maze is addressed by a pair of (row and column) coordinates. The type of cell indicated by a char variable is determined by the input map file (see below). Don't make any changes to this class but you can extend it by creating a subclass of it.
2) level1.txt – An example of a map file used to set up the layout of a maze. The encoding is illustrated in the last section.
3) Images: Here are some simple images (which can be downloaded from WebCT under Assignment/Assignment2 folder.) to start the game with. Feel free to create your own.

- 32x32 images:

5. Marking Criteria

- Implementation of basic function (50%)
    - Quality of the user interface (10%)
    - Pacman and each of the ghosts, each having their own thread used to move them on screen (12%)
    - Animation (7%)
    - Sound effects (6%)
    - Implementation of game rules (15%)
- Implementation of advanced features (30%)
    - Adding an extra level (6 marks)
    - High-score table (6 marks)
    - The implementation of intelligent ghosts (10 marks)
    - Implementation of one additional feature (8 marks)
- Demonstration of testing and program structure(10%)

- Report (10%) - please explain with sufficient details the data structure you used, thread implementation and animation, etc.

Please note that you will be marked down if you don't detail your implementation of both basic and advanced features in your report.

Warning

Avoid plagiarism: Do not try to pass off any downloaded code as your own work. You must acknowledge all sources. This includes information from the Internet. Plagiarism is a serious offence and will be dealt with accordingly.

6. Submission

Submission Date

The deadline for submission is **Friday 15th April, 2011**.

Submission Format

You should

1. print out your code with your report and attach a completed coversheet and submit your work to the School Office by **3:00pm Friday 15th April, 2011**. The blank coversheet can be found on WebCT under Assignments/Docs/ folder
2. zip up your

    - source code (.java files)
    - any other resources needed (eg images)
    - report with demonstration of testing

    and upload the ZIP file to WebCT **before 11:00 pm on Friday 15th April 2011**