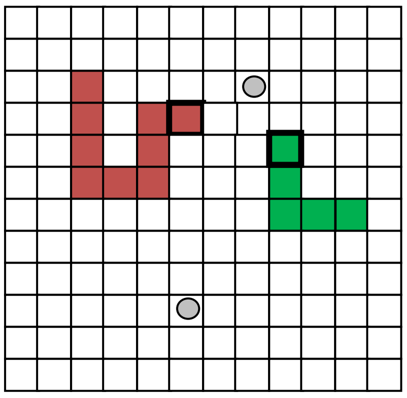
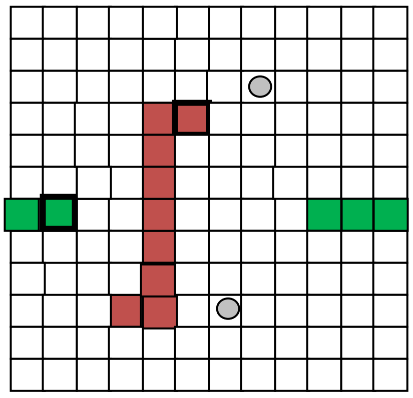
**School of Computer Science and IT**
**Software Architecture Design and Implementation (COSC2391/2401)**
**2013 Semester 1**
**Assignment 1 – Multiplayer Real-time Game (30%)**
**Group Assignment (Maximum 3 students Minimum 2 students per team)**
**1st Draft**
**(Demo/Submission in week 8)**

## 1. Introduction

You are required to write a simple multiplayer game where two snakes compete for food items which allow them to grow. These food items should appear at random locations but not on top of any snake. The aim of the game is to outlast the other snake(s). The snakes are allowed to leave the board and enter in the opposite sides. Each player should be allowed to change the directions using left and right arrow keys. The snakes will continue to move in the direction. The up arrow key may be used to double or triple the speed while the down arrow key may be used to bring it down to normal speed. The first player staring the game should be allowed to specify the number of players (between 2 and 4). The game should start when the required number of players (snakes) joins in. Each player should be allowed to specify one of the four remaining corners as the starting cell (top-left, top-right, bottom-left, bottom-right) in the order they join the game. The snakes should initially occupy only two cells. Whenever two snakes collide head to head moving in the exact opposite direction the larger snake is the winner, and in all other cases the snake running into another is the loser. The game should stop when only the last remaining snake is left, which is considered to be the winner.





### Assignment Objective

This assignment is designed to help an enjoyable experience while you develop techniques and skills needed to design and develop client server applications using sockets, multithreading and synchronization. As this assignment is a group assignment it is important that you spend sufficient time designing the classes, protocols, etc., before you start the implementation phase. This assignment will initially require you to develop an appropriate protocol and architecture that reflects quality attributes such as reducing the amount of network traffic generated and game extensibility (such as changing the number of cells). You should also take into account synchronization issues as two snakes or food items should not be allowed to move into the same cell at any one time. However, you need not put too much effort in the GUI aspects in the early stage. A design check point (carrying 20% of the marks) is provided to allow you get feedback with your protocol design.

## 2. Division of work in among Team Members

All the classes and protocols must be designed jointly. It is suggested that one of the team member undertake to develop the game server program while the other the game client program. If there are three members in the team, the third player should develop the utilities/graphical objects needed by the server and clients; otherwise common utility/GUI (if any) classes may be developed jointly. In the final submission code developed by the individual member as well as the code developed jointly must be indicated clearly. Server files and client files must be placed in separate packages.

## 3. Suggested Development Strategy

**Design the classes and the protocols and get feedback from the lab assistant**
Spend time decomposing the problem and coming up with appropriate protocols to enforce game rules. You may want to leave out the additional requirement and bonus features out at this stage.

## 4. Protocol and Initial Class Design (6 marks)

Show your initial design during your lab in week 6 (starting 15th April) including:

- Protocol Design
- Initial Class Diagrams

## 5. Demonstration (16 marks)

Demonstrate your working system during your lab session commencing 29th April (week 8).

Each group member will be assessed individually on the following:

- Basic functionality on either client, server and utility classes (this can be demonstrated with comprehensive test classes if the full system does not function together)

(4 marks)

Marks will be allocated to each group for:

| | |
|---|---|
| • Server functionality | (2 marks) |
| • Client functionality | (2 marks) |
| • Synchronization/playability | (2 marks) |

Bonus Marks (up to 4) for enhancing the game through any of the following. If you are in a team of three you need two of these features to get the bonus marks.

A. Registration and score tracking mechanism. If two players are involved loser gets 0, winner gets 1. If three players are involved first loser gets 0, second loser gets 1 and the winner gets 2. Similarly for four players.

B. Better animation (double buffering) and Graphics
C. Moving Food Items (Rabbits) using AI or some form of logic.
D. Varying number of cells (allow initial player to specify)
E. Any other feature/rule that enhances the game and playability.

**The basic features must be completed before any additional ones are attempted.**

## 6. Final Submission (8 marks)

One submission per group via *WebLearn* by **Sunday 5th May 9.00 pm**

**Final submission of the program should match the Demonstration version (with the exception of documentation)**

You must include the following:

- The code for client, server etc. divided into appropriate packages
- All classes and test classes (these must be commented specifying whether it is done an individual or as a team)
- Updated protocol diagram
- Updated class diagrams, sequence diagrams or other design diagrams used in the design of your system
- A file called readme.txt which contains a breakdown of each student's contribution, a list of incomplete functionality and known bugs (additional marks may be deducted for not declaring this in the readme.txt file)

Marks will be allocated to each group member individually for:

- Coding (including documentation) of basic functions (4 marks)

Marks will be allocated at group level for:

- Updated design (4 mark)
  (Highlight the changes)

## 7. Group Work

It is expected that each group member will contribute equally to the assignment, as specified by the Division of Work (section 2). The overall mark for students may be adjusted if the readme.txt file does not reflect an even contribution from all students.