# ESIR SR – Projet Monitoré 2015

## 1 Objective – Overview

You are asked to design and implement a simple *distributed game*.

To solve this problem, you can decide to use a standard client server design code in Java RMI (standard design). You can also decide to explore more adventurous designs (e.g. peer-to-peer), and languages / technologies (Scala with Akka agents, Erlang, Node.js, a PaaS platform, GWT, etc.). Choosing a more adventurous design will earn you extra marks.

Precise marking details are provided at the end of this document.

## 2 Objective: Gaming System

Your system should implement the following game: several players share a field strewn with sweets (Figure 1). The player who collects the most sweets wins. A game ends when there are no sweets left in the game. Note that two players cannot end on the same position, and that one sweet can only be collected by at most one player.

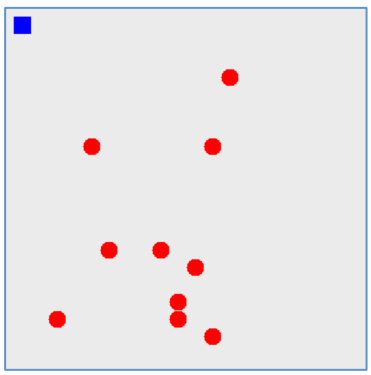

Figure 1: A game field with 10 sweets and one player

This gaming system should consist of a central gaming server and one client program:

- The client program should enable a player to connect to a gaming server, and start collecting sweets, while displaying the position of sweets and other players. The client should display the player's current score, and whether the player is currently winning (i.e whether she has the top current score).
- The gaming server should provide the game logic, and handle client connections and disconnections. When a game ends, the gaming server should notify all clients of who the winner is, and start a new game.

In addition to the above, you should also build an automated suite of tests to assess the robustness of your game engine. Testing a distributed application can be challenging, and you might need to include dedicated methods, interfaces, and stubs (*bouchons*) to improve the testability and controllability of your code.

## 3 Notes:

- You are **not** expected to provide a **sophisticated user interface** for the client program (simple geometric shapes are ok). **A basic straw man example** demonstrating how to display a geometric shape that can be moved with the keyboard **is provided as a starting point.**
- Your game engine should ensure that concurrent actions by the clients always result in a **consistent state of the server**.

## 4 Deliverables and Marking Scheme

You are asked to work in pairs for this coursework. Each pair of students will be asked to **demonstrate** their solution during the final marking session (see introductory session and on-line course site). In addition to your functional code, you should implement a **series of automated tests** to exercise your game engine. You are also asked to **create a repository** of your code (e.g. on the ISTIC forge: http://forge.istic.univ-rennes1.fr/ or GitHub). Specific marks are allocated to the quality of your tests, and the use of the forge.

In addition you are asked to write an **individual report** of roughly 900 words (~ two pages) that you **should submit electronically** along with the **source code** of your solution. Your report should describe the *overall design* of your solution, and its *motivation* (why have you chosen to do things the way you did them). It should also highlight any *technical challenge* or *limitation* you encountered. Finally you should discuss how your work could be further *improved* and *extended*.

Finally, some bonus marks are reserved to those students who choose to explore an **innovative design or language** (or both).

Your final coursework grade will be calculated as follows:

| Overall Component | Additional Information | Weighting (%) |
|---|---|---|
| **Basic Server (Level 1)** | **First Marking Session** | **50%** |
| | *Completion of client and server programs as specified (demo, in pair)* | *30%* |
| | *Appropriate suite of automated testing of the game logic* | *20%* |
| | *Use of an appropriate versioning repository* | *10%* |
| | *Bonus for innovative design* | *20%* |
| | *Short report (~ 900 words, individual)* | *20%* |
| **TOTAL** | | **100%** |

## 5 Deadline, marking, and submission:

IMPORTANT:
The marking session for the coursework will **be announced on the on-line course web site**. Make sure you know how to access it, otherwise ask.
Your solution (archive containing your source code + individual report) should be **submitted electronically** through the course web site **by midnight the day before each marking session.** (The archive containing your source code may be submitted only once for each pair of students.)
Please use the **naming convention**: "FAMILYNAME1_Firstname1_ FAMILYNAME2_Firstname2.zip" for your archive, and "FAMILYNAME_Firstname.xxx" for your report.
Late submissions receive a grade of zero.
You will be asked **to demonstrate** your work from your submission.

- You need to come to the marking session with your exercise completed. We will not be able to provide support during the marking session.
- You should be able to explain what you have done clearly, to show that you understand the concepts introduced.
- Checks for plagiarism and collusion between pairs of students will be carried out.