Software Distribution and Mobility Seminar 2010
Flikken

e-mail: egonzale@vub.ac.be
office: 10F731

Deadline 21 June 2010.

Delivery Both the documentation and code of your project should be delivered via the drop box of Pointcarre in the form of a <name>-project.zip file on the day of the deadline **before 16:00**.

1 Project Assignment

The purpose of the project is to implement the Flikken game. You have already implemented a small functionality of it during the lab session 3/7. The project's aim is to grow that exercise into a *playable* application. Flikken is a virtually augmented game in which players equipped with mobile devices interact in the physical environment augmented with virtual objects. Players are organized in two teams which determine their role in the game. The policemen work together to shoot down dangerous gangsters on the loose before they achieve their goal of earning 1 million euro by committing crimes. What follows are the rules of the game for this assignment.

When does the chase start? At the start of the game, both the policemen and the gangsters are in their headquarters (HQ). When a gangster leaves his HQ, the policemen HQ is informed which in turn informs the policemen present there and the chase starts.

How can gangsters commit a crime? Gangsters know the location of places with large amounts of money (e.g banks, casinos, etc.). In order to commit a crime and get the reward, a gangster needs to collect burgling equipment around the city (e.g knives, explosives, guns, etc.). When the gangster is nearby a place of crime, he can commit the crime if he has the necessary burgling items. He loses the items but gets the reward. When a gangster commits a crime, policemen are informed of the location of the crime and the amount of money stolen.

How can policemen shoot down gangsters? Players can see the position of all nearby team members. Additionally, both gangsters and policemen are periodically informed of each other's positions (e.g. every 5 minutes players can see the position of opposite team members). Policemen get a gun with 3 bullets at the start of the game. When a policeman sees a gangster, he can shoot him and the gangster may die if he is within the damaging area of the bullet[1]. If a policeman doesn't have more bullets, he can return to his HQ and get more.

How can gangsters defend themselves? Gangsters get a (non-rechargeable) gun with 3 bullets at the start of the game. Additionally, a gangster gets three items for his defense when the game starts: a gas bomb (which kills anybody in a determined radius where it was dropped for a period of time), a radio jammer (which disrupts the connectivity of the nearby policemen, preventing them to know his location) and a bulletproof vest (which protects

[1] To keep it simple, assume that a bullet kills the first person reached within a certain radius where it was shot.

him for a time interval against one single shot). He can also pick up more items in the game area when he is nearby.

2 Non-Functional Requirements

There are some requirements w.r.t. the distributed design of the game.

1. The application should be fault-tolerant such that failing computational units do not hamper the game from being played. You must assume that *every* computational unit in the network can fail at *any* point in time. More concretely:
   - Players may enter and leave the network at any point in time. While a player is disconnected, he and the other players can continue playing. Hence, he may pick up virtual items and commit crimes.
   - The game state should remain consistent in the presence of player disconnections. When a player reconnects, game state should be reconciled. For example, a virtual item denotes a *unique* object so that if a player already picked a virtual item up, a player reconnecting to the game shouldn't see this item anymore.
   - In order to make the application fault tolerant, you should exploit as much as possible a peer-to-peer organization. However, you may assume the presence of a device at the HQs for bootstrapping the game. This device may act as server providing players with necessary information to play the game when they move out of the HQ. For example, when the game starts, the gangsters HQ could provide gangsters with the necessary equipment and the location of crime targets. However, keep in mind that virtual items should only be visible to the gangsters when they are nearby it (even if their device carried the information about it from the start).
   - Message sends to players can fail.
2. Note that failures are (unreliably) detected using timeout as a heuristic.
3. Two entities (e.g. virtual items, players, etc) can be considered to be nearby each other if their locations are within a certain radius.
4. You may assume that there is only one game that is being played at a time.

3 Testing and Report

Besides implementing the application, you also have to create interesting scenarios to test your implementation. These test scenarios show how your application behaves w.r.t. the different operation modi. In order to test those scenarios, you can extend the Java GUI or the AmbientTalk unit tests provided in the lab session 3/7, or you can implement whatever other means you may find convenient. Keep in mind that the distributed design should be the focus of your project.

You will write a small report about this project. This report must be **no longer than 10 pages** (excluding figures or diagrams) and serves as a guide for evaluating your project. The report should include:

1. An overview of your application and how the implementation fulfills the requirements.
2. What were the important cases and design choices (w.r.t. distributed aspects) to consider for this project?
3. Description of test scenarios and behaviour of your application.
4. A small "manual" about how to run and test your application.
5. Which AmbientTalk build you used.

4 Evaluation

The project counts for half of your final score for the SDM seminar. It will be evaluated mostly on good distributed design. What is **important**:

- You should aim to fully exploit a peer-to-peer architecture and have a fault-tolerant design.
- The testing should be focused on trying the application from the distributed point of view. You don't have to provide a fancy GUI to test your application, a testcase in AmbientTalk is as valid. In other words, the GUI is of **no** importance.
- The quality and the structure of the code **are** important.
- The project must be implemented in AmbientTalk. As in the lab sessions, you can exploit the symbiosis between Java and AmbientTalk to make use of Java classes for e.g. your data structures such as a Hashmap. However, all distributed communication **must** be implemented in AmbientTalk. In other words, you **cannot** use other technologies (e.g. Java RMI) as your distributed computing framework.
- Make sure that any Java code is compatible with **version 1.5**.
- This project assignment is to be made individually. Cooperation **is not** allowed in any form.

Being creative and implementing additional functionality of your own will be appreciated.

Good luck!