Assignment 3

Instructor: Musard Balliu

November, 2015

Abstract

In this assignment, you will write a multi-threaded application with a graphical user interface, using external APIs from the `becker.robot` package.

The task

In this assignment, you will use the `becker.robots` package in order to create a simple version of the PacMan game.

The graphical user interface can be subdivided in three parts:

1. the menu;
2. the game world;
3. the player controls.

The menu

The menu bar contains two items: the actions menu and the settings menu.

The actions menu allows the user to pause and resume a game in progress, and is depicted in Figure 1.

The settings menu selects the difficulty level of the game and is shown in Figure 2

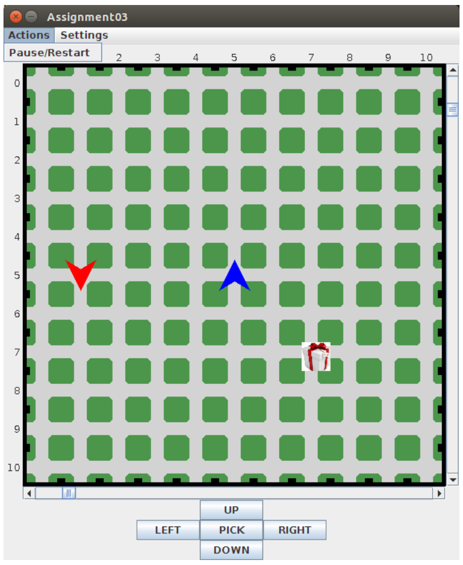


Figure 1: The Actions menu

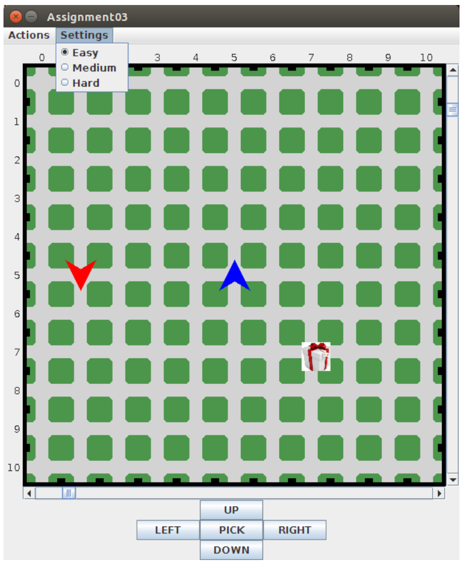


Figure 2: The Settings menu

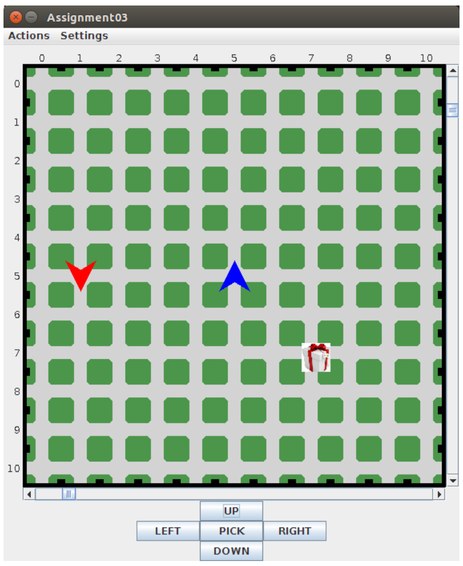


Figure 3: The game world

The game world

The game is played in a rectangular city enclosed by walls, in which there are two robots: one, controlled by the user, and another one, controlled by the computer. The two robots are distinguished by their color (in this document, the user controls the blue robot, the computer the red one, but this choice is not compulsory). Inside the city there is also a prize. The game world is depicted in Figure 3.

The game controls

The controls provided by the game allow the user to control his robot using the mouse, by clicking one of the four motion buttons *up*, *right*, *down*, *left* or the *pick* button. The user's robot will only move or change its direction if the corresponding button is clicked. If the robot is facing some direction d, say *north*, and the user clicks on the button associated with the same direction d, say *up*, then the robot will move one step towards that direction, that is *north*. The controls are seen in all previous figures in the lower part of the game frame.

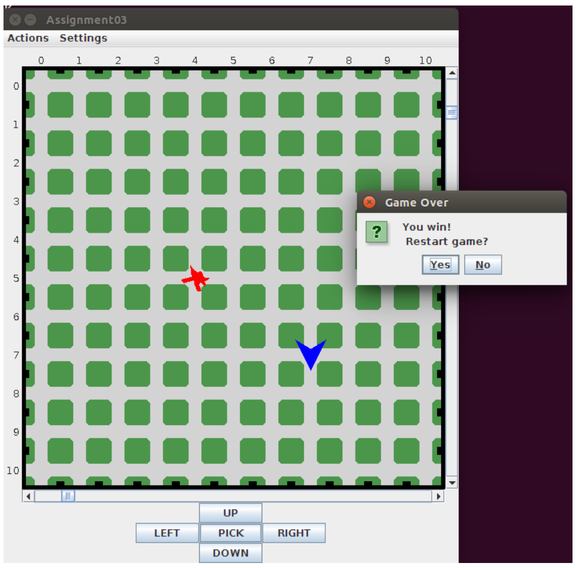


Figure 4: Game over, the user has won

The game

When the application is started, the game is in progress, there is no need for the player to explicitly start the game. The initial level is “Easy”, which should cause the adversary robot to move much slower than the user robot (at one fifth of the speed, for instance).

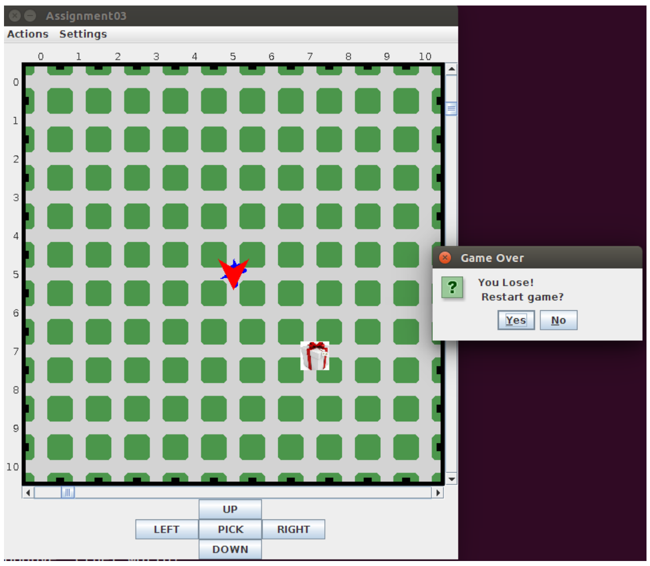
The user can pause the game and then resume it, using the actions menu. Therefore, this menu provides the same functionality as Becker’s standard `Start` button.

Using the settings menu, the user can change the difficulty level. “Medium” should cause the adversary to move faster, but still slower than the user (half the speed, for instance). If the setting “Hard” is selected, the adversary should move as fast as the user robot.

In all these cases, changing the difficulty level causes the game to restart at the selected difficulty level. It is not necessary (though certainly acceptable) to continue the current game at the new difficulty level.

The aim of the user is to pick the prize. If the user manages to do that, the adversary robot is destroyed and a dialog is displayed, as in Figure 4.

The adversary robot moves at random. If it encounters the user robot, then the user robot is destroyed and a dialog is displayed, as in Figure 5.



You Lose!
Restart game?

Figure 5: Game over, the user has lost

In either case, if the user chooses the "Yes" option, the game is restarted, at the level of difficulty of the previous game. If the user selects "No", the frame is closed.

Remarks and Notation

This assignment may seem intimidating, but we covered all you need to complete it in lectures 9 and 10.

You can download the icon used in these images from the course home page, but you can use another icon if you like it better. This is reqiored in order to get a VG (see Grading below).

You can make the user interface richer, if you want, but the elements mentioned here **must all be present!** Not having a settings menu as specified above will cause you to fail! Having more settings, for example for the color of robots, or the number of robots, etc. is, on the other hand, perfectly acceptable.

You must use the `becker.robots` **package!** There might be other similar packages available, but you must use this one. Failure to do so will result in a failing grade.

Finally, **do not use a forms designer or similar tools**. You need to show that you can select and use the appropriate graphical components in order to pass this assignment.

The remaining explanations will probably not make sense before you start working on the assignment.

The adversary robot must be started in its own thread. This is the reason why it is easier to restart the entire game when the difficulty level changes, rather than changing the properties of the robot on the fly. There are problems when running robots in independent threads.

Similarly, there are problems if the user attempts to control the robot before the game is started, or after the game has stopped. This is the reason for starting the application with a game already in progress, and for making sure, using the dialog, that after the game is over either a new game is started, or the application is closed.

Of course, if you have solved these problems, then you can do things differently, but let me know (and mention it in the `README`).

Grading

- To get a G, you can use the appearance of the Thing (already present by default in the library) to represent the prize.
- To get a VG you must, however, change the appearance of the Thing used to represent the prize and use the icon available for download from the course webpage. A default instance of Thing is not acceptable for getting a VG.

Note that these requirements are necessary, but not sufficient, to get the respective grades (see Administrative matters).

Administrative matters

Strive for readable code with appropriate comments! While the ultimate test of a program is that it does what it's supposed to do, **I should be able to read the program and understand it**.

The assignment is to be completed in groups of at most three students. (Three is the preferred size, you get no bonus points for submitting in a smaller group, but you don't lose points either.)

It must be uploaded to the GUL system **by 23:55 on November 11**.

Please note: **The submission is to be made via GUL. It's no good sending it to me, either before or after the deadline!**

Only java source files should be uploaded, no class files. Your java files should be put in a zip-archive with the following name:

```
        assign3_author1_author2_author3.zip
```

where author1, author2, author3 are the surnames (family names) of the group members. You must also supply a README-file containing the names of the authors and their social security numbers, together with a brief statement about each author's contribution. For example, "author1 has been responsible for the menu interface, author2 for the game logic and author3 has worked on the user controls". A statement such as "All authors have contributed equally to all aspects of the program" is also acceptable.

Note that **each author must submit individually via the GUL** (even though it's the same program!). Only students who submit via the GUL can be graded.

After the deadline has passed, each group **must present solutions to the TAs during the first Thursday supervision session**. Exceptions are to be agreed with the TAs. Group members are expected to know and explain all parts of the code despite their statement of contribution.