Master 1 informatique
UFR mathématique et informatique
Université de Paris

Programmation avancée - Project

Language : Python or C++

Start 7th November, Due 1st of December, 23h59 (strict deadline, the project must be validated on before 23h59)

This project is strictly individual (any copying from other students or online sources will be strongly penalized). It is OK to copy short snippets of code (i.e. less than 5 lines, and not implementing a complete functionality) from an online source, but the source (URL) must be indicated in a comment.

If there is any problem or something is not clear, ask a question to the dedicated forum on moodle. Any question revealing parts of solution to the other students will be considered as cheating and treated accordingly.

On moodle, you need to send:

- the files of your project in a zip file named NAME_PA_PROJECT.zip (where name is replaced by your lastname in capital letters). You are free to divide your work in one or several files, but your program should be launched by executing python on main.py or compiling main.cpp (if the compilation line requires more input, please indicate it in the pdf report),

- a pdf report of 5 pages maximum (no minimum) describing and documenting your work. In particular, you will describe which functionalities have been implemented.

In addition, we will organize a small individual presentation of your project after the deadline.

You are free to import any module/library you would like as long as it is necessary for your project and it does not fully implement one of the functionalities.

Good luck!

Master 1 informatique
UFR mathématique et informatique
Université de Paris

Project: Fancy Fencing

In this project, you will program a basic fencing game. In this game, one player can score points by making contact with the opponent through his sword.

The goal of this project is to demonstrate that you are able to implement a reasonably complex program in Python or in C++. As such, a particular attention will be put while grading on the quality of the code (including coding style and comments) and the usage of the languages' functionalities seen in class. This should also be shown in your report. **In addition, your report should precise which and how functionalities have been implemented**, and how they can be used.

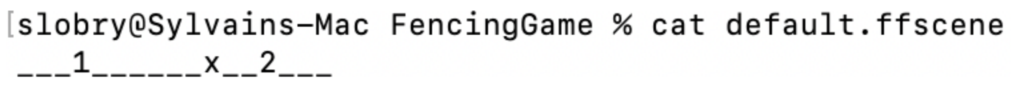
Required (implementing all these functionalities will give you a grade of 10/20):

At minimum, your program should be able to let two local players to play a game with a display in the terminal. The number of frames per second at which the display must be refreshed is a mandatory parameter to the program. The game is set in a scene (i.e. stage) that can be changed by the player through files. The scene to be used must be read from files having the .ffscene extension.

This file must contain one and only one line in which the following characters can be found:

- "_": default element/cell
- "1": starting position of player 1
- "2": starting position of player 2
- "x" an obstacle which can only be jumped through

For instance, the following file



```
[slobry@Sylvains-Mac FencingGame % cat default.ffscene
 ___1______x__2___
```

Is a valid scene.

One player can:

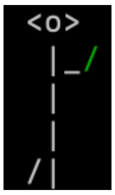
- Move left (by typing character "q" for player 1, left arrow for player 2). One step to the left moves the player by one cell to the left, and takes <movement_speed> frames.
- Move right (by typing character "d" for player 1, right arrow for player 2). One step to the right moves the player by one cell to the right, and takes <movement_speed> frames.
- Jump left ("a" for player 1, "l" for player 2). One jump to the left, moves the player up in <movement_speed> frames, left in <movement_speed> frames and down in <movement_speed> frames. As such the movement takes 3*<movement_speed> frames to complete.
- Jump right ("e" for player 1, "m" for player 2). One jump to the right, moves the player up in <movement_speed> frames, right in <movement_speed> frames and down in <movement_speed> frames. As such the movement takes 3*<movement_speed> frames to complete.
- Attack ("z" for player 1, "o" for player 2). One attack takes <attacking speed> frames to reach the opponent. It succeeds and score one point if it is not blocked and the opponent is less or equal than <attacking_range> cells from the attacking player at the end of the attack. If both

players, succeed an attack at the same moment, no point is scored. In both cases, the players re-start at the starting positions.

- Block (“s” for player 1, “p” for player 2). Effectively blocks the attack if the defending player is closer than its <defending_range> at the moment the attack is performed. The block is active for <blocking_time> frames.

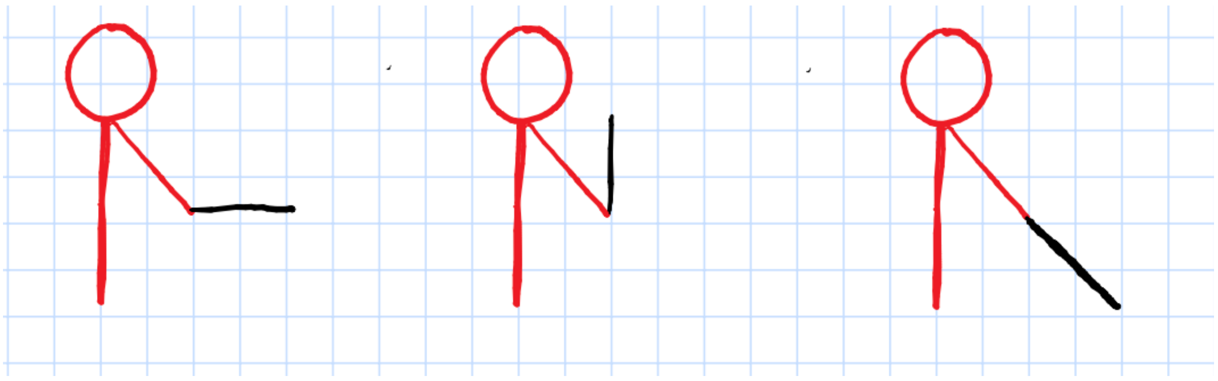
Each <attribute> is an attribute of the player (which values have to be decided as a parameter in the program, or through files). The two players can (and should) have different attributes values.

Players are represented by simple characters. An example is shown below:



```
 <o>
  |_/
  |
  |
 /|
```

A player can have 3 states: attack; block; and rest, schematically represented below in the same order:



Improvements (doing all the improvements in addition to the requirements will give you 20/20):

- Add a pause menu showing the different controls and allowing to quit the game.
- Allow to save and load the game at any point during the fight.
- Allow to choose between different scenes.
- Add a graphical version of the game.

Bonuses (doing some of the bonuses in addition to the requirements will give you additional points. Maximum grade is 20/20):

- Allow for games played over the network
- Add an artificial intelligence controlling one of the two players.
- Add music/sound effects
- Put your code on github (including screenshots and a good readme). The github must be private until the defence.