Assignment 2
DEV 2 - Year 2015-2016
Methods

The Dev TEAM

January 4, 2016

1 Goal and description

The goal is to improve your design and implementation skills on data structures and methods. For this purpose we created an improved, but still incomplete, version of the city simulation from *Assignment 1*. In this version we included, besides the city and it's roads, new game elements: bridges, rivers, boats and harbors . Your task is to:

- design and implement the boat data structure
- move through the scene both cars and boats

2 Software requirements

To work with the simulation you need `PyGame 3.4` and `Python 3.4`. You can download the `PyGame 3.4 x86` click me. PyGame is a set of Python modules designed for writing games. The simulation comes with a *template project*. The template is available on N@school and GitHub under the voice `Assignment 2`.

3 Details

Classes As you will see in the template we have improved our old classes and the logic:

- **Tile.py**, we added new properties to: river, bridge, and harbor. You can use those properties to properly move cars or boats(a car cannot seal and a boat cannot drive on the motor way)
- In **Game.py**, the function `build_scene` now returns three values: `entry_road`, which is a list that contains all tiles in the game (the first item of this list is the tile on the top left corner); `entry_rivers`, which is a list that contains the top river tiles; and `bridge`, which contains all bridges in the game (a bridge is special road that crosses a river).

NB. You need to study those structures and codes before you start with your implementation.

Game.py We also we provide you a main loop in `Game.py`. The `Main` function is the entry point of the game. Precisely in the `Main` you find the a block of code which runs indefinitely the game. We extended the old main loop so to draw the new game elements. The main loop is `missing` the calls to the draw and update functions.

Node.py If necessary, we also provide you a function `select_one_random`, which takes a list as parameter and returns a random value of the list.

4 Tasks

Task 1 [**class, attribute**] *Design* the `boat` data structure that should at least provide the following attributes:

- A position, which references the tile the boat is in
- A canRemove, which tells whether to remove the boat from the scene
- A texture, where you store the image of your boat

Reuse the car data structure and extend it with the attributes canRemove and texture.

To load a texture from the content folder use the following instruction:

```
pygame.image.load(os.path.join( "Content", <<your_image>> )).convert_alpha()
```

Task 2 [**methods**] extend car and boat data structures with methods for the rendering and updating behaviors.

- **Update**, which updates the current position of the entity (by moving it through the scene respecting the properties of the tiles). If a car enters a garage, or a boat enters a harbor or leaves the map then their attributes canRemove are set to true.
- **Draw** the current entities. The method has the same logic as the draw function from assignment 1. Adapt the code.

Task 3 [**functions**] in the file Game.py add the following codes:

- Add an update boats and update cars functions that iterate through their respective lists and update each entity. Filter the entities whose canRemove attribute is set to true.
- Add a draw boats and draw cars functions that iterate through their respective lists and draw each entity.
- In the main loop find the proper location where to call the above updates and draws functions.
- Add new boats and cars according to some conditions (for example every five iterations of the main loop).

5 Submission and deadline

Contribution: *Groups of 2 students is allowed with individual responsibility*
What: *One PDF per student for all code + comments (comments: explain your code)*

When: *The Friday of week 7*
Where: *On N@school*

GOOD LUCK!!! The Dev TEAM ☺