linea orizzontale

**Manuel Dellabona,**

**Bachelor in Artificial Intelligence, computer programming course.**

Castle War

**15/03/2022**

The final exam project consists in the design and implementation of a strategy game named “Castles War”. The game is intended to be a 2D side-view game, implemented using the library pygame.

# Goals

1. Handle and learn the basics concept of graphic
2. Get used to an object-oriented programming paradigm
3. Be able to manage an entire project in a limited amount of time

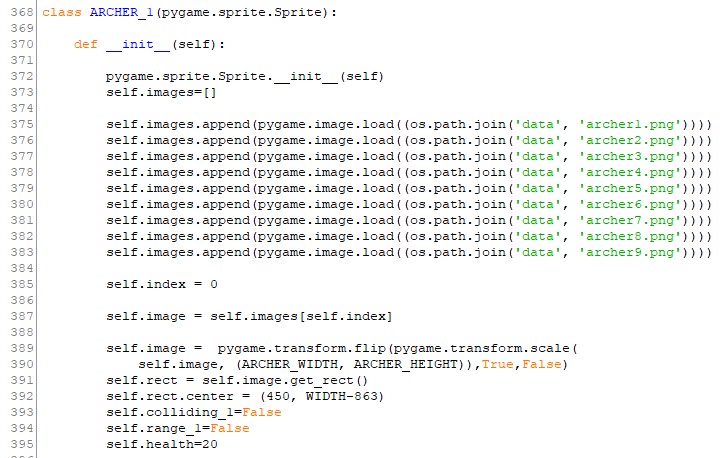
# PROJECT SECTIONS

# Draw background and static graphics

I have decided, for all the elements that do not need an interaction with other objects, to print them on the surface as static graphics. For those static graphics we can simply load the images and scale them to the right height and width and then use the function blit to display them.

## Designing the Sprite object

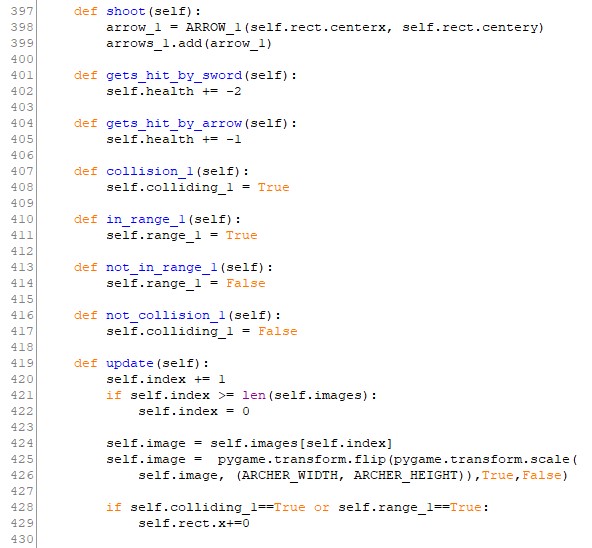
In order to handle animations and the stats of each element in the game we need the Sprite objects. The classes that can create the pygame objects are designed in this way:



In the \_\_init\_\_() function we can store all the frames for the animations, set all the initial statistics for that element (ex. self.health) and the variable to check for the collision that involves that sprite.

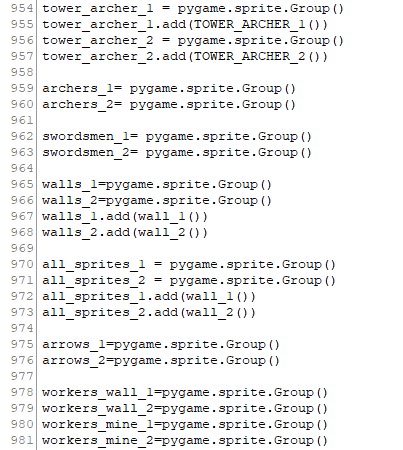
The methods in these classes are used to attack (ex. spawn arrows from archers) and modify the statistics (ex. reduce self.health due to a sword hit ) .

And at the end we need to set up the update(), a method that will be called each frame in the game loop and allows us to let our objects move and be animated.



## Sprites’ groups

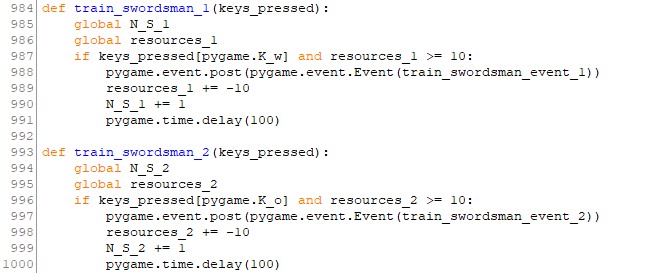
## The sprites’ groups are used to manage sprites/objects with the same characteristics and print them on the screen.



## The sprites’ groups help us a lot also in the interaction between objects. In fact I have decided to exploit the pygame.sprite.spritecollide that checks the collision between a sprite and all the sprites in a group.

## Game functions and game logic

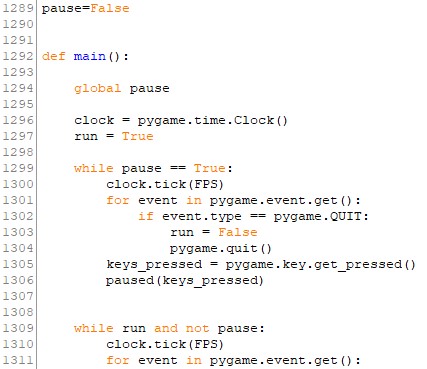
We need functions to be called in the main game loop, such as



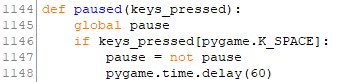
These functions can determine the whole game logic, in this project are used to train and spawn the troops, to manage the resources and attack the opponent. In this example they take as an argument the keys\_pressed variable that contains the action that the user has done on the keyboard and they modify the global variables that represent the players’ resources and the number of troops in the barracks.

## Game loop

In a game, all the changes happen in the game loop, that is a loop that we want to update x time each second, where x are the FPS (frames per second) of our game.



as seen in this picture, we can interrupt the game loop simply quitting the game or by calling the paused() fun, a function that freezes the game state.



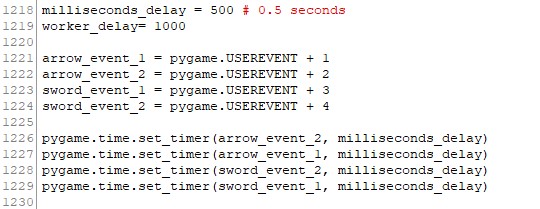
## 

## 

## 

## Events

The pygame module uses an event paradigm that allows the user to modify the game loop.

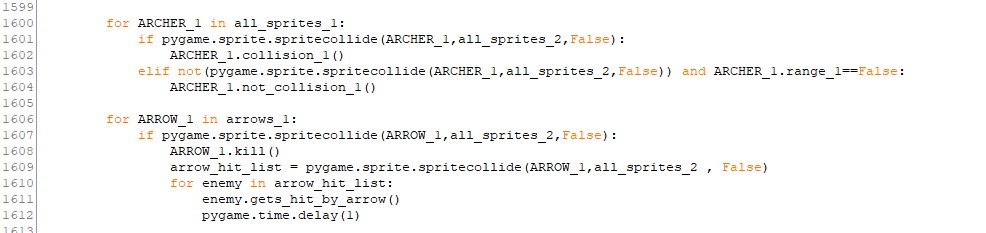


## 

I have exploited the events to start a timer that limits the archer to spawn an arrow every 0.5 sec, the swordsmen to attack every 0.5 sec and the workers to extract resources or repair walls every 1 sec.

## Looping sprites’ groups

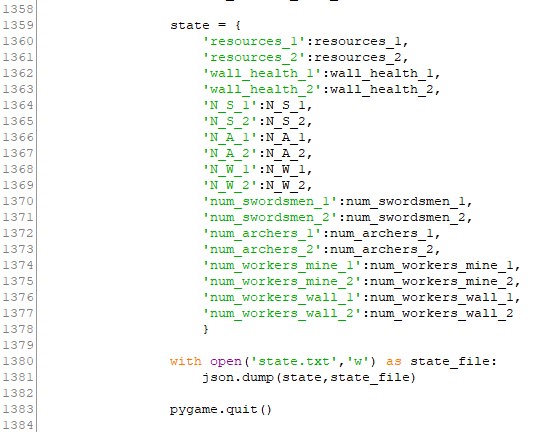
In order to execute a function for all the sprites involved in a game change we can loop the sprites’ groups and call a method for each object in that group.



With this trick I can recall the methods that activate collisions for all the objects in a group that are in the list that is returned by the spritecollide function.

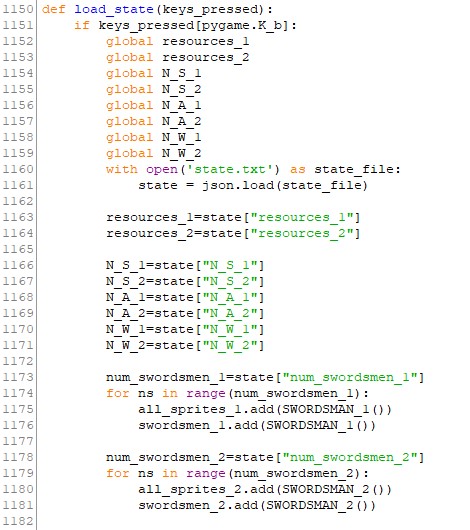
## Save the game state

When the game is quitted without any winner, the program automatically saves the current game state that can be reloaded once the game is restarted.



This can be done by saving a python dictionary as json text that contains all the relevant statistics that define the game state. I have decided that the variables that define the game state are the ones that represent: each player's resources, the health of walls and the number of troops and workers in the barracks and deployed in the game.

## Load the game state



Through a call of the function load\_state(), the game read the jason text created before and reconvert it in a dictionary, and from that restore the game state by setting the statistics and respawning the troops.

## Main difference from the assignment

I have decided to reduce the “turn” into the actual frame, in this way the two players can train, unleash and gain resources almost simultaneously.



This design choice leads to a more dynamic gameplay where you continuously need to perform actions in a limited amount of time.

The players have to gain an amount of resources that will permits them to deploy a massive attack to reduce the opponent’s wall health.