Project “Castle Wars”

Computer programming, Artificial Intelligence

By Martina Navarria

Introduction

In this document I will explicit additions and changes I apported to the original instruction scheleton of this project with their relative justification or reasoning.

Assuming the person reading this document has an understanding of the subject and the project’s general idea, I will not explain every function, class or line of code, but only the ones relevant to a full understanding and evaluation of my work.

Description of the goals

My main goal when starting this project was to have a functional game that could entrertain two players for a decent amount of time. I started from the basics with the help of some more expert programmers I found on youtube (mentioned in the references) and some suggestions from collegues. I then evolved my project after an encounter with Professor Ferrari to make the code more efficient and implement some features I was having troubles coordinating (explained in the section “problems I encountered”). In the end I am quite proud of the result and I believe to have a valid code and program.

Design choices

I chose to keep the design of the moving units, as I believe they were easiest to evaluate and distinguish for familiarity.

I changed instead the original designs of buildings because I still wanted my project to be unique throughout the game and have a personal touch. I tried to be constant in style with the design of units, the same goes for the choice of the wall health bar, the background, the text font and the icons indicating the resources.

Regarding the “Game Over” screen, the “Intro” screen and the “Pause” screen, I decided to keep it as simple as possible for better understanding of the commands, keeping the same style as the background to give continuity.

Practical choices

It was my decision to modify some dynamics within the game, each for a reason.

One of the first things I created in the game were a resources limit and a set health maximum for each player, for each then I coded the worker who interacted with them. I didn’t like the idea of always having a lot of resources or always having a unit repairing an already-intact wall, so I decided to kill them as soon as the maximum I set was reached. This feature gives the user another thing to pay attention to and something more to do during fights other than just sending troups continuously.

In additon, I believe at the start, the archer units were too similar to the swordsmen and I wanted to create a bigger division in troup strength, almost like a “special move”. So I incremented the health and power of archer troups so that they could damage more. In order to make so that it was a “special move” and less easy to use, I incremented the cost of the units and set a rule that only one archer for each player can be alive at a time. Obtainig this way a more exclusive move that the players have to strategically play. To make this even more obvious, the index of the amount of resources changes color when it reaches the minimum amount possible to create an archer. This however poses a distraction players have to pay attention to. If created immediately when the index changes color, this troup will determine the last move of it’s player unless a worker is creating resources at the same time.

Because of the above implementation, I found useless to have an “unleash all” button. Instead I chose to have troups leaving as soon as their training time was over if and only if their command key had already been pressed. In alternative they leave after their training time whenever the command key is pressed.Otherwise to obtain the same effect as “unleash all” button, a user can create all troups, let them train and then press the command key, unleshing them all at the same time. Having only one main kind of troups I found much more practical using the same key for this other feature.

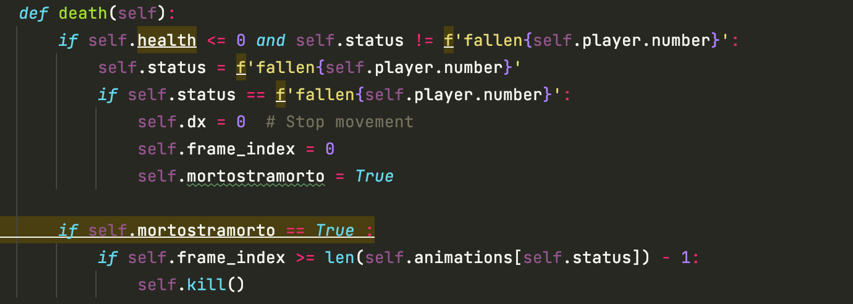
Regarding wall arrows, I really struggled a lot and although working on it until the very end I wasn’t able to obtain a functional result, so I kept my only working design that doesn’t make use of any particular math or calculation, it simply coordinates the moment an enemy collides with the tower range with the shooting of an arrow in a specific spot, but I hated it so I removed the feature.

Testing and results

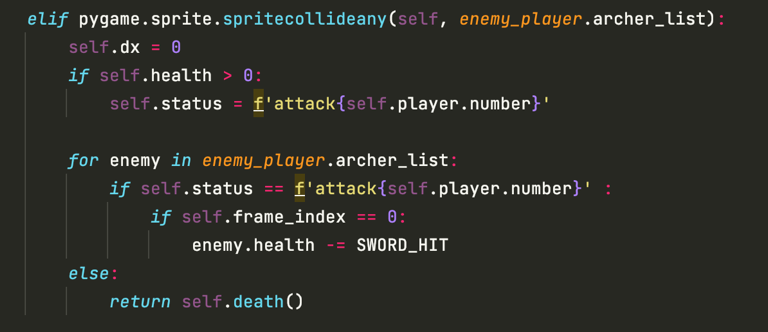
The only thing during the testing of the whole game that worked immediately with no problems was the saving and the pausing of the game, for which I used the pickle library, and the movement of sprites, for which I used the os library. For the rest I had to perfection the code after testing it. Many issues were just distractions and easy fixes, such as money reducing while creating units, units dimentions, units speed, healthbar of the wall not moving correctly. Some other issues were not so small and I had a harder time finding a solution. I will explain them in detail in the next section.

Problems I enocuntered

Some things I struggled with were coordinating the end of troups health points with the animation of them falling. But I managed to solve this with the function “death” and the flag “mortostramorto” in the Unit class, that defined better within the program the moment of death and the start of the new sprite loop.



Another issue I faced was causing my troups to die all together. I solved it by specifying that each unit was facing an enemy unit for distinction.



My biggest problem was wall arrows, I just couldn’t seem to find a formula to automatically find the position of the sprite and calculate the shooting moment. I will keep working on this even after the evaluation and figure out an efficient way to make it work, but for now I deicded to disable this option, but I hope to make up for it with other implementations and improvements I creatively apported.

Improvements

The health bar only disappears as the wall health lowers instead of having a border indicating it’s amount when full. I find this more clean and less predictable then knowing the exact amount of wall health left. However it does change color when reaching the last 25%, giving the user a chance to save their castle before its health reaches zero.

The speed of each unit is based on how I saw better fit with regards to their role and power.

The amount of wall health is based off the power and speed of units.

I chose for the amount of resources to reach maximum 100 to keep the user attentive and focused on all parts of the game.

I added a column for each player to keep count of the number of troups currently on the field since many times the troups can ovelay or be synchronized therefore be confused. It helped a lot while testing collisions and sprites loop.

References

For youtube references I visited the channels called:  
-Clear Code

-Sally Maree

-Coding With Russ

For further help I confronted myself with many of my collegues.