Final Project

Computaion III

Pygame - Endless Wilderness Explorer

23rd October - 17th December

NOVA IMS

2024

Contents

1 Project Rules

This project aims to evaluate your ability to implement the knowledge obtained during both the practical and theoretical classes of Computation III while also assessing your capability of solving problems in a creative and technical way.

The groups are build by 3 students and are chosen by yourself. You must Submit your groups on the corresponding moodle link from ***October 28th*** to ***October 31st, 23:59h***. The deadline for the delivery of this project is the **17th of December**. Groups who will not respect this deadline will suffer a value for each day of delay (up until 3 days, afterwards the project will not be accepted and the students will have to do the second phase exam).

All projects must be defended on December 20th . The defenses will start at a soon to be announced time on the **20th of December** and a detailed schedule will be provided in that same month. In the project defense, you may be asked questions regarding your implementation of the project as well as questions about the theoretical concepts.
All members of the group will be asked questions and students who will not be able to provide satisfactory answers will be discounted and potentially fail the class.

You must deliver a **UML class diagram** of your project.

Projects will be delivered through Moodle (Using the correct delivery link).

A specific list of the elements of the project delivery and their corresponding rules are provided in Section 4.

Make sure you read every section carefully!

2 Project Description

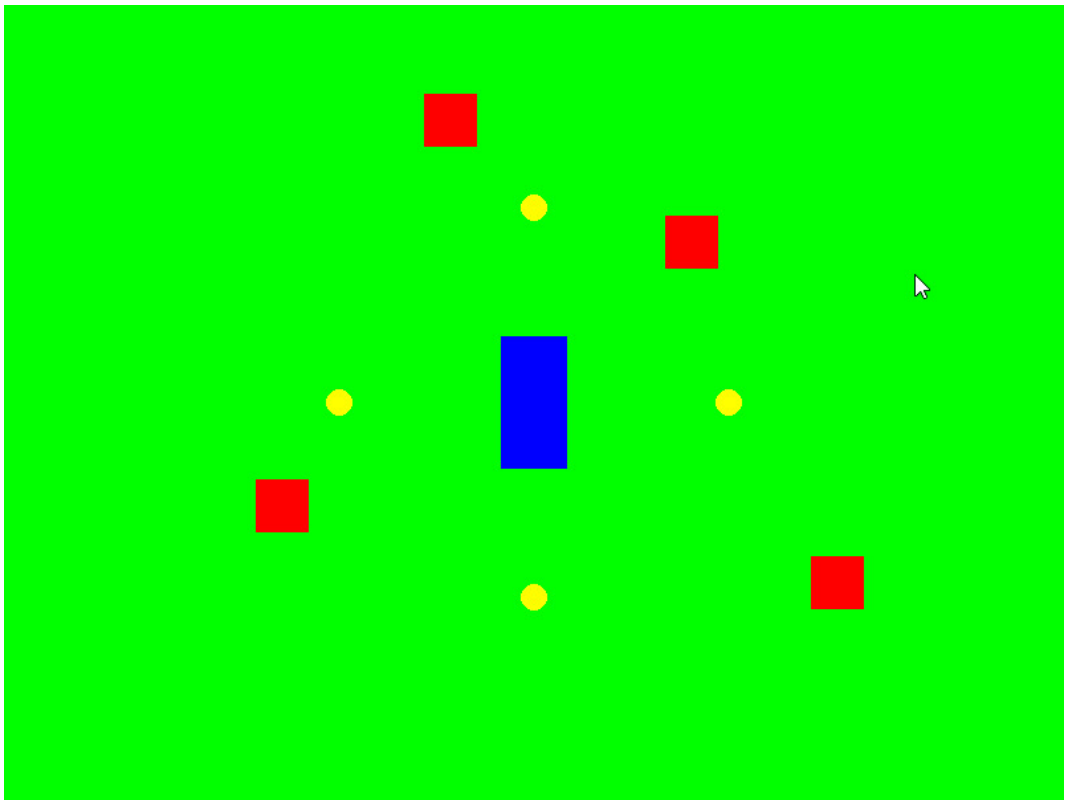
Your goal in this project is to implement the necessary classes and code in order to complete the Pygame implementation of the Endless Wilderness Explorer game we will start developing in the practical classes.

Your imagination and implementation ability is going to be tested! You will be given a general idea of what further improvements on the game are the basis of this project but you will, however, be given the liberty to make it come to life in the best way you find possible! Be creative, think out of the box!

Overall the improvements on the base game consist on the creation (and implementation) of different **"Power-ups"** and **Weapons** that will affect the game in some shape or form. Additionally, you will be requested to add a **Treasure chests** to the game.

Furthermore, you are asked to **enhance the UI** created during the classes in the best way you see fit. The improvements or modifications made to the UI will greatly impact the overall perceived quality of your game, so be sure to dedicate effort to this step!

The specific requisites for the project (and their further explanation) are found below. Make sure you read the rules carefully and implement all the **mandatory** classes/methods and **necessary** classes/methods for the functionality of your program in function of the project requisites.



3 Project Requisites

Overall you are requested to:

- Implement the **two** mandatory "Power-ups" (described below).
- Implement at least **two** "Power-up" of your choice.
- Implement treasure chests in the game which contain three different weapons/skill options from which the user must choose.
- Implement at least two new weapons of your choice
- Improve the UI.
- Add an instructions panel to the game where the rules of the game and the "Power-ups", Weapons and Treasures are explained to the user.
- Add the functional limitations necessary to make the game operational and aligned with the goals of the game. For example, The character should not be able to go outside of the map.
- Give the player the option to shop for new weapons. Thus, a monetary system must be implemented alongside the option to purchase weapons/abilities.
- Add the saving feature that stores all the progress the player has made.
- Add one extra functionality to the game beyond the game improvements and previously described additions. This functionality needs to be novel and is totally up to you! This is another great chance to improve the game and impress us!

Note that bonus points will be attributed to groups that used GitHub **throughout the development** of the project. If you do so, please provide the GitHub repo's URL.

3.1 "Power-ups"

The PowerUp class represents the concept of a "Power-up" to the game. Simply put, a "Power-up" is something a player can "catch" by walking on it. Catching a "Power-up" will affect the game in some way for a **temporary** amount of time.

A "Power-up" must have a specific look so that the players try to catch it. Different "Power-ups" have different looks (e.g., one is shaped like a pink star while, other is

shaped like an orange triangle, etc...) since they have a different effects.

"Power-ups" appear randomly during a game with a certain probability. While you have full freedom to determine your own probabilities, you must implement a system where **some "Power-ups" are more common than others.**

It must be clear that a "Power-up" is in play. Therefore, catching a "Power-up" and benefiting from it **changes the player colour** for as long as the "Power-up" is still active.

More specifically, regarding the "Power-ups", you are requested to:

- Implement an abstract class called PowerUp. This class will contain *at least* two abstract methods called affect_player and affect_game.
- Implement the specific "Power-ups" as children of the mother class PowerUp.
- Implement different "looks" for different "Power-ups".
- Implement the appearance of a "Power-up" in the game based on a certain probability.
- Implement the necessary visual modifications so it is clear there is a "Power-up" in play and who is benefiting or suffering from it.

3.1.1 Mandatory "Power-ups"

The two mandatory "Power-ups" you are requested to implement are:

1. Invincibility: The invincibility "Power-up" makes the player invincible for a limited amount of time. More specifically, monsters hitting/attacking the player will not cause the player any damage.
2. De-spawner: The de-spawner "Power-up" removes from the game a certain amount of monsters (probabilitically determined). Additionally, while the de-spawner "Power-up" is active, the number of monsters that are spawning needs to be reduced (not to 0).

3.1.2 "Power-ups" of your choice

You must implement **at least two** "Power-up" of your choice. Now that you know what "Power-ups" are and which ones are mandatory, it is up to you to surprise us with original ideas that can make the game more challenging and interesting. Possibilities are endless!

Invest time into making these "Power-ups" impressive and fun to play with.

3.2 "Treasure Chests"

Treasure chests are another type of "catchable" sprite that you need to implement in your project.

Specifically, treasure chests should have a relatively low chance of spawning on the game map. When the player catches a chest (by touching it), the game should pause and present the player with three randomly selected options to choose from. These options should be chosen with certain probabilities from a wide range of possibilities that you have implemented. These could include weapons, temporary power-ups, or skills (e.g., permanent upgrades to player damage, health, etc.).

After the player selects one of the three options, the game resumes from where it paused, with the chosen ability now active.

4 Delivery

The delivery of this project **must** contain the following:

1. Your code. You are not requested a report and therefore your code **MUST** tell the story of your project. You must **document** your code well and comment everything!

    - As was mentioned in class, other than the "story telling" comments you **must** document all the code by writing the **docstrings** using the **numpy style**, as was covered in the theoretical classes. Once again, treat your code as your report or, better yet, treat the docstrings as your report so it has to be written in a proper standard.

2. The resulting UML of your project

You must deliver a zip file with both your code and the UML on moodle, using the link that will be provided to you at the time.

5 Final Notes

Make sure you impress us with your implementation of the "Power-ups", Weapons and "Treasure Chests"! This means taking time to work on the different looks of the "Power-ups", the way having a "Power-up" in play alters the look of the field, the originality of the "Power-ups" you implement, the wide variety of options that can appear in a chest and the way the game looks overall and its playability. Your job is to improve what we did in class, do not stop at just the mandatory requisites as those are the minimum required and not enough for the highest marks! Keep in mind that your code efficiency, organization and alignment with OOP directives will be evaluated.

Make sure you read the project requisites (found in Section 3) thoroughly so you don't miss a mandatory requisite!

Good work! ☺