Programming Project report

Super Mario

Jiahao Chen & Aryan Verma
Group 89

Computer Science

Abstract

This project tries to emulate the game Super Mario. After following the steps indicated on the project requisites, a final functional version has been reached. The code has been designed taking into account future maintainability, code understandability, and new game functions.

Table of contents

1. Classes Design
2. Most Relevant Fields and Methods
3. Most Relevant Algorithms
4. Performed work
5. Conclusions
6. Personal Comments

1. Classes design

We have created a total of 20 **classes** to fulfill all the requirements of the Final Project. We used inheritance, information hiding, encapsulation and other object oriented techniques.

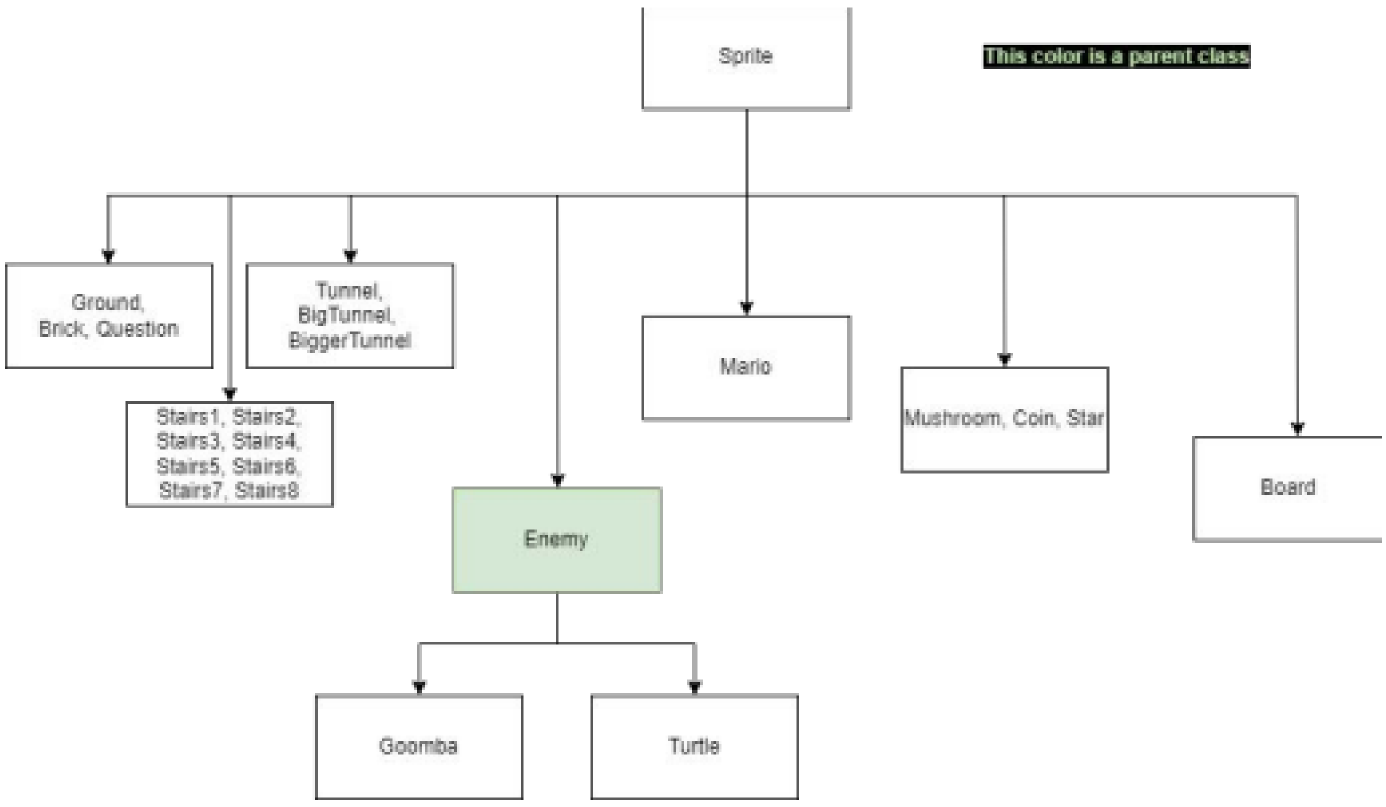
Sprite
This color is a parent class
Ground, Brick, Question
Tunnel, BigTunnel, BiggerTunnel
Stairs1, Stairs2, Stairs3, Stairs4, Stairs5, Stairs6, Stairs7, Stairs8
Mario
Mushroom, Coin, Star
Board
Enemy
Goomba
Turtle

Fig 1. *Classes Design scheme*

Board

This is the most important class and contains all the information about the world. It contains functions that allows the player to kill the enemy, move, jump. It also updates the board while drawing the map of the world which is stored in several lists about the objects and the blocks imported from another separated file. This class is responsible for everything that the player sees while playing the game.

Board also makes sure that the enemies and the blocks spawn and function how they are supposed to.

Enemy

This is the mother class of all the in-game enemies, Goomba, and the Turtle. It contains some common functionaties and parameters of the enemies such as the move function and whether these enemies are alive or dead.

Goomba, Mushroom

One class per type of enemy with their own characteristics and their own sprites.

Mario

Contains the code related to the sprite of Mario who is controlled by the user. Here can be found the player's movements, like walking, running and jumping, which also includes killing by jumping. At the same time, it contains some fundamental functions that allow Mario to interact with the enemies (like the one to check if mario is on an enemy) and special objects like Mushroom.

Ground

Contains the code related to the ground, it acts as a basic floor for Mario to stand on.

Brick, Question

Contains the code related to the bricks and the question blocks, the key difference between these and the ground block is that these are intractable. Mario can break the bricks, when he is in his Super Mario form and can get Mushrooms out of the question block.

Tunnel, BigTunnel, BiggerTunnel

Contains the code related to the green tunnels seen across the map, it acts as a basic obstacle for Mario which he can jump over to move forward. Three different classes were made to have three different sizes.

Stairs1, Stairs2, Stairs3, Stairs4, Stairs5, Stairs6, Stairs7, Stairs8

Contains the code related to the stair blocks seen across the map, it acts as a basic obstacle for Mario which he can jump over to move forward. Eight different classes were made to have Eight different sizes.

Mushroom

Contains the code related to the conversion of Mario to Super Mario when it is touched by Mario.

Coins

Contains the code related to the coins in the game, It helps the player accumulate points as he goes along the map.

2. Most relevant fields and methods

The most relevant methods are related to the movement and the interaction of the objects.

move: This is a method from the enemy class that allows all enemies to move in the same direction with a constant speed assigned to them which allows them to change their direction once they hit an obstacle. Also, a method with the same name and similar functionalities was implemented for Mario.

in_the_ground/in_the_enemy: They allow Mario to know if he is in the ground or not. The difference between the two is that in the case of in_the_enemy() we can evaluate if Mario is above a specific enemy (which was very useful to shoot the Koopa) instead of going over all the enemies.

dead_turtle: This algorithm in the Board class implemented for the turtle/Koopa was interesting. It was necessary to use several parameters of the boolean type which relates to the three possible states of Koopa: normal alive turtle, paralyzed turtle, and shell-turtle. Once Koopa is dead (or paralyzed), he is able to revive after a period of time (5 seconds in this case). If during this period of time, Mario steps on it again, then it is shot. In this state, Koopa is able to kill all Boomba he comes across on the road. And finally, after 5s, it disappears.

restart: this long method of the Board class is essential to play the game multiple times. We can either restart the game because Mario died or because the game is over (when Mario lost all his lives, which is set to 3 by default). We can enter an optional boolean called game_over, and the method will react accordingly by setting the necessary parameters to their original value.

3. Most relevant algorithms

The most relevant algorithm that has been used is located in the Board class. Once the game is initialized, the program allows the player to continue playing and controlling Mario, until all three of his lives are exhausted, which can be either by dying from the enemies or by falling down the map from strategically placed gaps in the map.

In the case of Mario, a jumping function was implemented. It is based on the concept of a constantly accelerated movement, just like there was gravity in the game. We created a parameter called jump_force, which is constantly decreasing while Mario is jumping until it becomes 0 and Mario stops increasing its y-coordinate and starts falling (which is another function inside the Board class based on the same concept.

The collision issue was separated into 4 parts: up, down, left, and right. An evaluation system based on comparing the coordinates of Mario with the position of the objects that Mario can interact with was developed for each side. As a result, Mario is able to recognize when he can move in a 2D space and when the road is blocked.

In addition to just being regular obstacles, The bricks and the question blocks can be interacted with in their own different ways, the question block gives the user a special object, a mushroom that allows it to become Super Mario or a coin. while the brick block can be broken by Super Mario.

Finally, in order to move the map, we used several parameters called progress and Big_x (complemented with the previous_progress parameter from Mario class). They allow us to know the place we are on the map, and therefore printing the objects in their corresponding position. By subtracting progress and previous_progress, we can know how much distance we advanced in the last movement.

4. Performed work

We were able to achieve most of the steps that were asked for in the final project, which was writing code that was efficient. We were successfully able to replicate the entire map of the game by spending countless hours on the visuals of the game. Mario spawns with three initial lives, he has to hit a question block to gain the powers of a mushroom to become Super Mario. Along with Mario, several mushroom and turtle enemies also spawn in the map randomly. Mario is able to kill these enemies by jumping on them, while if he touches them, he loses a life.

When Mario is in his Super Mario form he attains two total lives and does not die from one direct contact from a mushroom or turtle enemy.

The turtle enemy can be used as a way to kill other enemies directly, as our code allows the player to use the shell of a dead turtle as a bullet, and use it to kill any Goomba that comes in its path.

An extra level of difficulty is added by the presence of a timer. The timer limits the amount of time the player has to complete the map, if the time runs out before the player has completed the map, he loses a life and has to restart all over again.

The controls remain the same as the ones that were shown to us as in the sample game, the arrow keys move forward and backward and Z to jump, and X to sprint. All the controls are made to be extremely responsive.

We were able to make the entire map, block by block to the very last point. Each and every block has been designed by hand in the pyxeleditor, the various blocks include Ground block, brick block, mystery block, tunnels, and even the stair block. All these were made with extreme precision to give the game the original feel.

5. Conclusion

During this project, we have learnt a lot of pyxel, python, and object oriented programming, although we have done so by working very hard and putting in countless hours.

The journey was cumbersome and challenging but it also helped us learn through trial and error: like creating a copy of a data instead of working on the original form can be extremely convenient, the use of classes and functions to emulate more abstract ideas of the real world, … Yet, for us, the more important lesson of this project would be group cooperation.

GitHUB was extremely useful as we used it to keep each other updated of all the changes we had made. Also, JETBrains code with me was also very helpful and helped us maximize our efficiency while coding together.

6. Personal comments

We have learnt a lot in this process and we are excited that we were able to make a close replica of a legendary game like Super Mario. This project helped us understand the value of teamwork and hard work.

We personally believe that students must be given a little brief about the uses and the features of GitHUB as it is an extremely useful and essential tool for any programmer.

All the bugs posed as roadblocks in our journey but they also gave us a sense of accomplishment as we solved them and continued on the journey of making our version of Super Mario.

To sum up: This was a wonderful experience which gave us real life experience in game development and object oriented programming, and also taught us about the significance of teamwork, which we are thankful for.