University of California Los Angeles

Computer Science 32: Introduction to Computer Science II

Super Peach Sisters

February 27, 2022

Haoxuan Li

Contents

List of Figures

List of Tables

Declaration

I, Haoxuan Li, hereby honestly declare that all work within this project is my own with the exception of those referenced in the acknowledgements.

Acknowledgements

I would like to thank Professor Carey Nachenberg for providing the game template and assets required to implement a visually appealing 2D platformer game, as well as Professor David Smallberg for providing the specifications for the project and opportunity to work on such an interesting project. Additionally I would like to acknowledge the Department of Mathematics of the University of California San Diego for providing a foundation to this LaTeX document.

Abstract

Super Peach Sisters, a game inspired by the hit Nintendo platformer, Super Mario Brothers, replicates the key mechanics of the original game but with a fresh coat of paint. This project involves the recreation of the platformer with a focus on developing class and object oriented logic typical in game design. Upon these composite data types, additional objects were created through inheritance and polymorphism, simplifying the code in both performance and readability. The objective of this project was to understand the similarities between game objects and design their interactions accordingly through shared actions and states.

Overview

In Super Peach Sisters, Peach starts out a new game with three lives and continues to play until all of her lives have been exhausted. There are multiple levels in Super Peach Sisters, beginning with level one, and during each level except for the last Peach must reach a flag to continue to the next level. On the last level, Peach must reach Mario in order to win the entire game.

The Super Peach Sisters screen is exactly 256 pixels wide by 256 pixels high. The bottom-leftmost pixel has coordinates x=0,y=0, while the upper-rightmost pixel has coordinates x=255,y=255, where x increases to the right and y increases upward toward the top of the screen.The provided GameConstants.h defines constants that represent the game's width and height. Every object in the game will have an x coordinate in the range 0 to viewWIDTH-1 inclusive, and a y coordinate in the range 0 to viewHEIGHT-1 inclusive. These constants can be found in the same header file.

Each level has its layout defined in a data file, such as level01.txt or level02.txt. It is possible to modify these files or create new levels by editing through text editor.

Peach is capable of moving left, right, or jumping upwards. Peach can also move left and right while jumping in the air as well, which makes it easier for her to jump from platform to platform.

As Peach works through each level, she must avoid coming into contact with any of her enemies. Coming into contact with an enemy, unless in certain situations pertaining to powerups. will cause her to die, either resetting the level, or ending the game if she has no lives left. Similarly, Peach must avoid all fireballs shot by Piranhas or she will die. If Peach dies three times, the game is over. At the beginning of each level and when Peach restarts the current level because she has died, the level will be reset to an initial state. That is, all actors will be placed in their initial positions and start with their initial states. All blocks will be reset, so they again release any power ups when hit by Peach, etc. Peach also starts out with no special powers

Some of the blocks in each level may hold special power ups, and if Peach interacts with them with her head by jumping into them, she can cause them to release their power ups, which Peach can pick up.

There are three different types of power ups: star power ups, mushroom power ups and flower power ups. Star power ups give Peach invincibility for a limited amount of time, so she can't be injured by coming into contact with enemies like koopas, goombas, piranhas, or piranha fireballs. Mushroom power ups give Peach the ability to jump extra high, allowing her to reach platforms that would otherwise be out of reach. Finally, flower power ups enable Peach to shoot fireballs to destroy her enemies.

The effect of a mushroom or a flower power ups lasts until either Peach comes into contact with an enemy, or she completes the current level. Similarly, invincibility from the star power ups lasts about 10 seconds or until Peach completes the current level, whichever comes first.

Assuming Peach has Fire Power,the player may hit the space bar to have Peach shoot a fireball. A fireball will destroy the first enemy it contacts, and then disappears from the game. A fireball will fly forward until either it hits an enemy, or it hits a block or a pipe in a horizontal direction. Beyond damaging enemies, fireballs will otherwise pass over all other objects in the game. For example, fireballs will pass over flags, Mario, mushrooms, flowers, stars and shells without damaging them. Fireballs fall to the ground as they move forward, so for example, if one flies past the edge of the current platform of blocks it will start falling down.

Each level may have up to three different types of enemies: koopas, goombas and piranhas. Koopas and goombas simply move left and right on their platforms, stopping before they fall off the edge of their platform or if they run into a block or pipe, and then turning to move in the opposite direction. Piranhas don't move, but will turn to face Peach if she gets close enough. Piranhas will also shoot fireballs at Peach if she gets too close. A fireball fired by a piranha will hurt Peach just as if it were an enemy coming into contact with Peach. A fireball fired by a piranha will fly forward until it hits either a block or a pipe, but beyond damaging Peach, it will otherwise pass over all other objects in the game without damaging them. A fireballs fired by piranha falls to the ground as it moves forward, so for example, if one flies past the edge of a platform of blocks, it will start falling down to the level below.

If Peach destroys an enemy, the enemy will disappear from the level and Peach will get points. When Peach destroys a koopa, it will produce a shell which will travel like a projectile until it hits a block, pipe or an enemy, at which point it will dissipate. Shells float right past Peach, they come into contact with.

As Peach plays, if she comes into contact with an enemy and she does not have any type of power she will lose a life, and the level will reset. If Peach has Star Power then coming into contact with an enemy will not hurt Peach, and will immediately destroy the enemy. If Peach has Jump Power and/or Shoot Power and comes into contact with an enemy, then she will immediately lose those power(s) but she will not lose a life, and her enemy will not be hurt either. In this case, Peach will have a short amount of temporary invincibility so she does not immediately die on the next tick as the enemy continues to maintain contact with her. If, after losing a life, Peach has one or more remaining lives left, the level and all of the actors are reset to their initial state and Peach must again play the level from scratch. If Peach dies and has no lives left, then the game is over.

If Peach reaches the flag on the level, then the level ends and Peach advances to the next level. If Peach reaches Mario on the final level, then Peach wins the entire game and the game is over.

Once a level begins, it is divided into small time periods called ticks. There are dozens of ticks per second (to provide smooth animation and game play).

The above is a shortened version of the original specifications and superficial overview of the project at hand by Professor Carey Nachenberg.

Experiment

1 Inheritance Charts

Active and Inactive Entities : In order to display the best possible inheritance that can be achieved. Additionally, the entities in the same vertical columns can also exhibit a possible use for polymorphism as although they function differently, they share similarities such as movement and tick behavior. It is easier to separate these trials of inheritance because as mentioned before, it is much realistic to picture possible ways of changing certain function, also known as polymorphism in order to facilitate the most effective path to design the entities. Figure 1 shows the active or more specifically moving parts of the game while Figure 2 shows interact-able but inanimate objects. If it is not apparent already, the amount of entities that exist in this game would be excessively complicated to code individually.

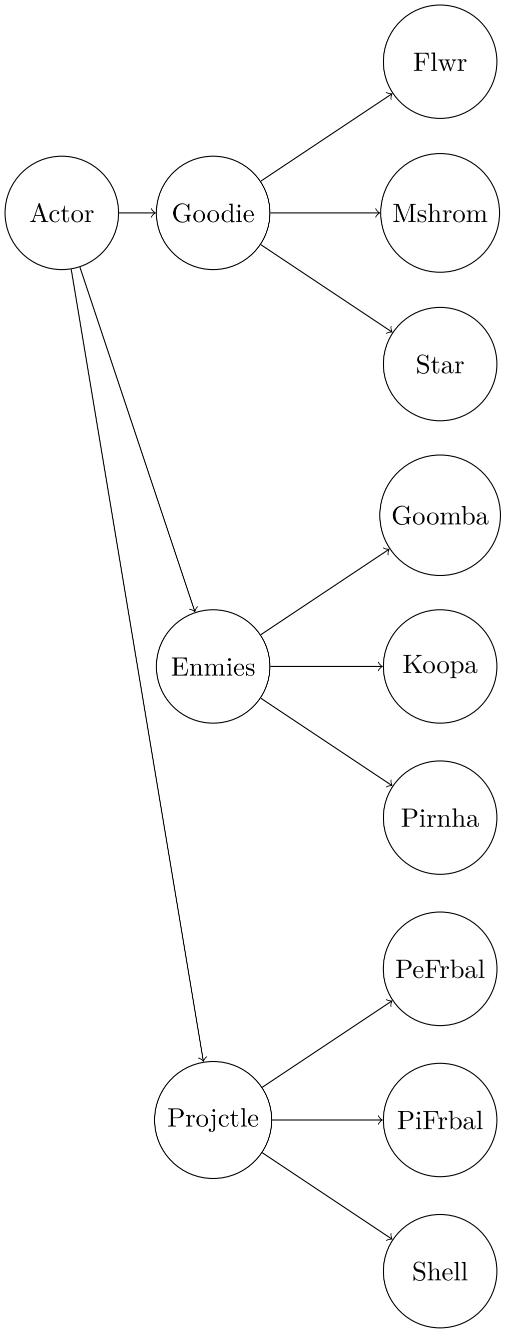

Figure 1: Inheritance of active entities.

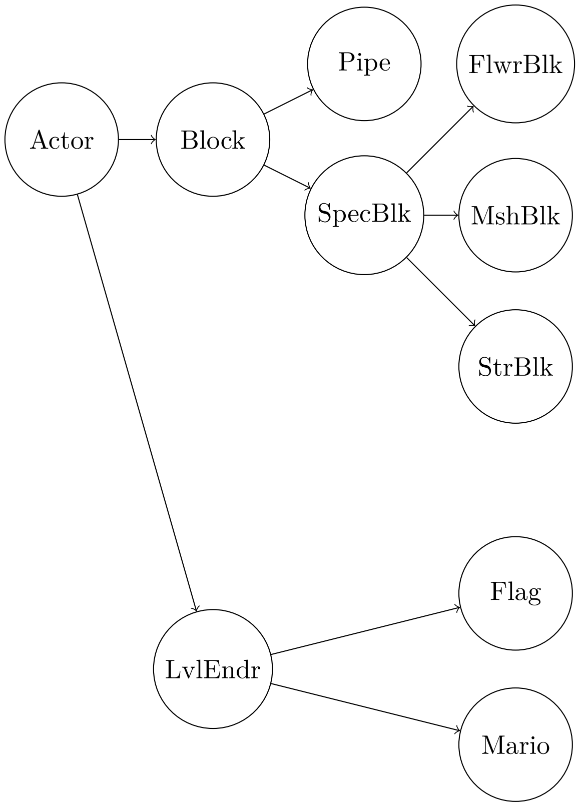

Figure 2: Inheritance of inactive entities.

Player Controlled Entity It also important to note that the player controlled Peach entity will not be able to inherit as much as the previous entities. However, certain functions can be morphed to functioned in a similar manner such as collision detection or movement physics. Therefore like most other entities, Peach will be derived from the base entity, also known as Actor, and end there.

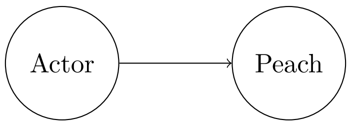

Figure 3: Inheritance of peach, a user controlled entity.

2 Peach States

In order to prepare specific interactions between the environment and Peach, we must be able to identify the proper states that Peach can exist in and how their interactions differ from the base Peach state. For example, although Peach could have both mushroom and flower powers, if she were to collide with a sensitive entity, then she would lose both her powers. On the contrary, the star power is solely time sensitive as Peach is unable to be sensitive to anything else and therefore would lose the star power after a specified time frame. Knowing these states allows us to simplify the functions that change Peach's state according to an overall effect rather than a specific one that could cause errors later on.

Goodies Removal Interactions and Sensitivity				
State	Enemy Sensitive	Projectile Sensitive	LevelEnder Sensitive	Time Sensitive
Peach	✓	✓	✓	X
Peach w/Mshrm	✓	✓	✓	X
Peach w/Flwr	✓	✓	✓	X
Peack w/Mshrm,Flwr	✓	✓	✓	X
Peach w/Str	X	X	X	✓
Peack w/Mshrm,Str	X	X	X	✓
Peack w/Flwr,Str	X	X	X	✓
Peack w/Mshrm,Flwr,Str	X	X	X	✓

Table 1: All combinations of possible Peach states with and without goodies.

3 State Changes

Enemies Movement A more simplest way of viewing an enemy entity movement is through state diagrams. From Figure 4, it can be identified that the basic movement of a goomba or koopa is left or right depending if it sees an air block at its next step. Therefore, we can assume that these two enemies will never have a falling contingency and therefore does not need a function to detect its lover parts of the sprite. Additionally, the piranha plant can also be simplified to a state diagram as the entity only has two states, attacking and not attacking. Therefore, the piranha plant can only exist in these states and therefore does not need functions to account for movement or collision below it's sprite. Figure 4:(0:start, 1:right edge detected, 2:left edge detected) and Figure 5:(0:Peach detected, 1:Peach not detected).

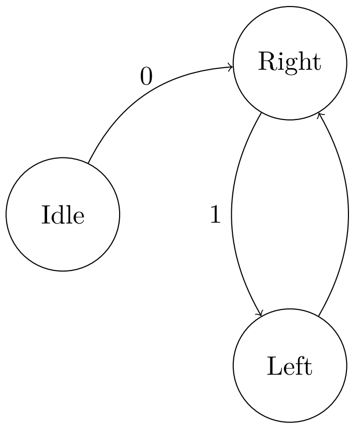

Figure 4: Possible states of a goomba or koopa.

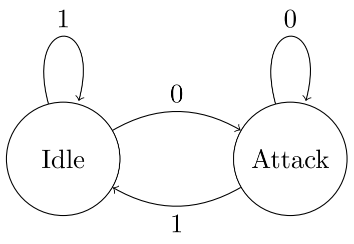

Figure 5: Possible states of a piranha flower.

Goodies Movement A goodie or powerup movement is a bit more complicated compared to the enemies. Since enemies continue to exist even after colliding with Peach, goodies functions differently. Tthe goodie can experience the physics of gravity as it can fall off edges and continue moving in the same direction while also falling. In that case, the goodie only changes direction when it hits a wall. Figure 6:(0:start, 1:right wall detected, 2:left wall detected, 3:collected by Peach).

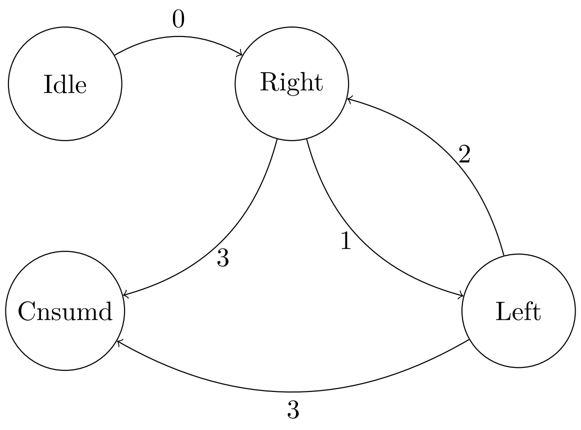

Figure 6: Possible states of an exposed goodie.

Conclusion

In discussion of the broader context, this game mirrors a rather simplified process of game creation. Additionally, we can also relate the use of inheritance and polymorphism to the process of optimization. The idea being that a well built game would be optimized to run as effectively as possible disregarding hardware limitations. Therefore, by building game elements off of each other, the code exhibits less repetition overall and runs more efficiently. It is important to note that although this is not an intensive program to run and minor adjustments might see little to none improvements on to the actual performance of the game, the code serves as a foundation to develop future game skeletons from scratch.

Future Scope

Although the project is very much completed and functions according to the specifications, there still seems to be some improvements that could have been made to the code in order to improve performance and overall smooth out entity interaction. For example, each active entity has a pixel movement that differs from other entities. This proved a problem early on in the project as some enemies would phase through Peach without triggering the appropriate state changes. In order to remedy this problem, separate detection functions in complement to the entities' movement functions were made in order to sense the necessary interactions. This made sense although the amount of these functions were inconsistent and seemed to only exist for specific cases even though a general sensing function could have improved performance. This greatly slowed down the frames between interactions, especially grinding the game to a halt if too many entities were spawned in and triggering the appropriate sensing and moving functions. This issue undermines the optimization part of the project as a fatal error can potentially end the game. Therefore, future recommendations in redoing this project would most likely include reworking the sensing methodically so that it functions the same to as many entities as possible, therefore correcting a potential error and creating an overall more robust code.

References

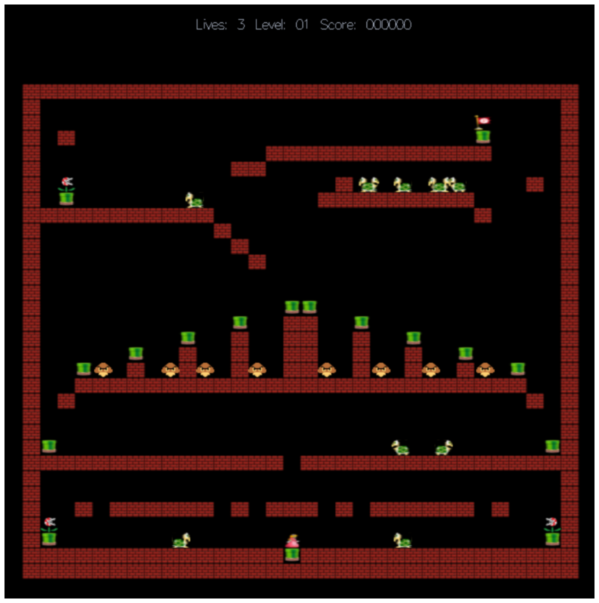

Figure 7: A screenshot of the first stage of the game.

Appendices

To view the raw code, read the project specifications, or download the game for you own viewing pleasure, all details can be found at my GitHub.

`https://github.com/supercoder-hao/SuperPeachSisters`