Object-Oriented Programming using a Platforming Game

Jeet Shah [1], **Suyash Lal** [1]

[1] *School of Computing and Data Sciences, FLAME University, Pune, India*

ABSTRACT

The authors are developing a game to teach the concept of inheritance from the of Object-Oriented Programming paradigm. This game is developed using the GDevelop game engine and emulates the older Mario Bros. games from Nintendo. The concept of object-oriented programming, more specifically inheritance, is being explained by the mechanics of the player character picking up power-ups. The effectiveness of the game has been studied qualitatively by user-testing on students with varying knowledge of object-oriented programming, from complete novices to computer science students. The game performs significantly better for students have some background in the concept of OOP. The feedback received from the game can be roughly summarised to the players coming away with the understanding that inheritance is better than rewriting code.

Keywords: programming paradigm, object-oriented, inheritance, GDevelop, gamification, Mario, Nintendo

1 INTRODUCTION

We chose to explain the concept of object-oriented programming via gamification as the object-oriented programming is deeply entrenched within game development due to its inherent nature of trying to emulate real life objects and scenarios. Previous draft of this concept was to try and give a crash-course on all the different concepts of object-oriented programming by also showing code examples which is used to build the game itself. We forego this idea due to two reasons: the authors were aiming to explain a lot for a relatively short game, and playing a tutorial is never really fun.

We settled on explaining the idea of inheritance using a platforming game due to its popularity as a game genre. Moreover, we wanted the game to resemble something the learners found familiar. This decision allowed us to make the de-facto decision of using the Mario Bros. game franchise from Nintendo as the clear inspiration.

We decided to explain the concept of inheritance by using power-ups (mushrooms, thank you Nintendo) to intuitively demonstrate the idea of extending or modifying the functionality of a parent object. This insight allowed us to also build a story out of this by adding a different power-up that tries to emulate inheritance by opting to build the object from the ground-up, instead of inheriting from the parent object.

The plot of the above is then formed by modifying our first idea itself. The idea that we can access the source code of the game you are currently playing seemed extremely interesting. Therefore, the only two non-playable characters present in our game are the developers of the game. One of them is experienced, henceforth dubbed the senior developer; the second character is, unsurprisingly, the junior developer. The power-ups have been developed by the NPCs who argue with each other about why their implementation is better.

2 GAME OUTLINE

2.1 Level 1

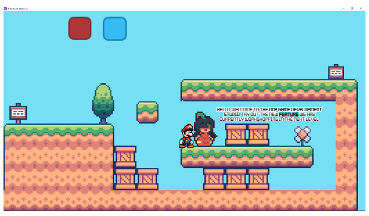
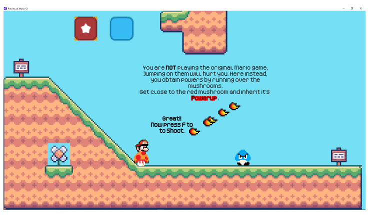

HELLO, WELCOME TO THE OOP GAME DEVELOPMENT STUDIO. TRY OUT THE NEW FEATURE WE ARE CURRENTLY WORKSHOPPING IN THE NEXT LEVEL.

Figure 1. Snapshot from the first level

The first level explores the basic mechanics of the game. It introduces the player to the basic control scheme and movement physics by requiring them to jump up and through floating platforms5 Moreover, we also tease one of the NPCs, the senior developer, in the first level. This is done to make sure that players recognise that there are other interactions within the game. We also introduce walking enemies that damage players - causing them to teleport back to the start of the level if they come in contact with them.

2.2 Level 2


You are **NOT** playing the original Mario game, Jumping on them will hurt you. Here instead, you obtain powers by running over the mushrooms.
Get close to the red mushroom and inherit it's Powerup.

Great!!
Now press F to to Shoot.

Figure 2. Snapshot from the first level

This level places the senior developer at the start of the level, urging the player to interact with them. The senior developer introduces the red mushroom power-up. The power-up is placed on the way to the end of the level. A swarm of enemies are blocking the way to the exit. The player is then informed of the new control scheme where they are taught how to shoot the walking enemies and defeat them. Right before, they exit the level, the senior developer shows up again and talks about how easy it was to implement this feature as we just extend the capabilities of the base player character object.

2.3 Level 3

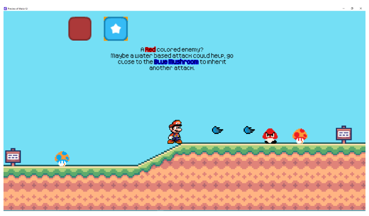

A Red colored enemy?
Maybe a water based attack could help, go close to the Blue Mushroom to inherit another attack.

Figure 3. Snapshot from the first level

Third level is essentially the same as level 2. The key difference being the power-up that is available here - the blue mushroom. Moreover, all the places where there is a senior developer interaction, there will now be a junior developer who talks about how rigid this object-oriented method of thinking is and inheriting the class only limits the imagination. He then mentions how much better this implementation of the power-up is before we exit the level.

2.4 Level 4

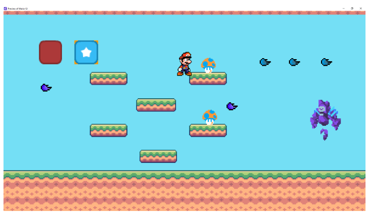


Figure 4. Snapshot from the first level

The fourth level is canonically the final level of the video game that is being worked on by our NPCs. This level showcases a boss fight at the end of the level. However, before you reach the boss fight, you will encounter the junior developer who seems to be scheming about how they will replace the red power-up with their own.

The player then moves forward and encounters a room where there are two blue mushrooms placed. As soon as they get closer to the mushroom, the boss wakes up and start shooting at them. When the player starts shooting back they notice that boss does not seem to be losing any health. The senior developer then shows up at the edge of the screen and urges the player to move towards the boss.

Moving towards the boss causes the game to freeze and the player to then get teleported to a completely black screen.

2.5 Level 5

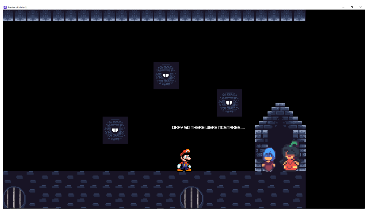


Figure 5. Snapshot from the first level

This level is supposed to provide much needed context as to the unfolding of the plotline in the game. We see an instance of both NPCs towards our right and we realise that this a timeline of their conversation after the buggy blue power-up was discovered. The conversation is used to enlighten the players about the pitfalls of not using inheritance while also providing some arguments as to why it is not always the correct answer but it is something that must be considered, first and foremost.

3 EFFECTIVENESS OF THE GAME

The authors had mentioned the use of the Kirkpatrick framework to perform testing. This has not been the case due to a lack of participants and the lack of time itself.

The authors did conduct qualitative testing of the game at it's various stages of development, however, only one participant agreed to be filmed for the same. Here is the video for the same. The participant commented on how the game felt functionally sound but seemed to build to something without having any sort of pay-off, educationally or emotionally. This comment by the participant gave us the idea of adding the fifth level which explains the concept more explicitly while allowing for a more complete emotional payoff.

Other participant's feedback led to the addition of the appearance of the senior developer in level one, as the participant found the introduction of the characters to be too abrupt if they started being introduced from level two. The authors also tuned the difficulty and length of the final boss fight to allow a player to be sufficiently challenged but not be overwhelmed by the same.

Most of the participant's feedback were used to tweak the dialogue between the characters. The authors believe that this can be further improved as more iterations of the dialogue tree are built. Most participant's had a lukewarm to positive response to the educational content of the game; with some not really engaging with it while others exclaiming that this allows them to understand and approach the object-oriented paradigm with more enthusiasm. The latter response came mostly from computer science students who have had some previous introduction to these topics.

4 LEARNINGS OUT OF THE PROJECT

The authors have learnt a lot about the Unity game engine. The engine is the best example of the object-oriented programming. The authors have learnt the importance of abstraction and building objects from the ground-up. One example that comes to mind is the idea of a character's collision with the ground. The authors wrote the logic to know at all times when an object is touching the ground or not. This collider element has been used for multiple different instances within the current demo.

More importantly, they have also recognised when to kill their darlings and let go of using Unity as the game engine to build a functional game in the designated time-frame. The recognition of over-ambition and the willingness of pivoting to using new tools are one of the biggest learnings the authors have taken away from this project.

The authors have also gotten intimately familiarly with version-control system, mainly git (using GitHub). A lot of hoops were jumped through to ensure that both authors could collaborate and work on the game simultaneously. The biggest thing the authors learned was about git LFS, and how to set it up.

The authors had planned to use the Kirkpatrick's framework to gauge the players' understanding of the topic before and after they have played the game Bachvarova et al. (2012). This caused them to understand frameworks such as the one mentioned to a better degree.

The authors have become increasingly familiar with the no-code game development platform, GDevelop. A lot of different methods were used to get GDevelop to do exactly what was needed, the authors had to fight through buggy physics interaction and a practically non-functional tilemap system. A lot of duplication of action was necessary across the created levels, which caused unnecessary frustration. The authors realised that GDevelop allows one work with an abstracted view of a game engine, however, using Unity will be beneficial proportional to the size of the project being undertaken.

AUTHOR CONTRIBUTIONS

J. Shah has contributed in the development of the game by coming up with the game loop and conduct research about all the tools required for the project. Moreover, all the documentation about the project has been done by them. Some help has been provided with creating the game on Unity game engine. The storyboarding, and some state machine designs have also been their contribution.

S. Lal has contributed in the development of the game by setting up and maintaining the GitHub repository where the game is hosted (while working with Unity). They have done most of the developing of the game on the Unity game engine and the creation of internal documents for easier collaboration. The author continued to take charge of the game development while working with GDevelop. They created and collated all the assets used in the game while closely working with the engine to develop the game.

ACKNOWLEDGEMENTS

The authors would like to extend gratitude towards Prof. Chiranjoy Chattopadhyay. The project implements the concepts that were taught to the authors in his course CSIT302 (Principles of Programming Languages) in semester V of the academic year 2023-24 at FLAME University.

PROJECT REPOSITORY

Here is the link to the drive folder where our project files and assets currently lie.

Here is the link to the video file that showcases the game as it currently stands.

REFERENCES

Bachvarova, Y., Bocconi, S., Van Der Pols, B., Popescu, M., and Roceanu, I. (2012). enMeasuring the Effectiveness of Learning with Serious Games in Corporate Training. *Procedia Computer Science* 15, 221–232. doi:10.1016/j.procs.2012.10.074