Multithreaded Gamification Project

Jash Popat [1], **Nishtha Agarwaal** [2]

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

ABSTRACT

Multithreaded programming is a complex topic, often taught in traditional classrooms with limited engagement and retention. This report describes a gamified learning experience designed to teach multithreaded programming through interactive gameplay, conceptual explanations, and visual representations. The game, THREADY, consists of five levels that multi-threaded programming, thread creation and management, thread synchronization, thread safety, and mutex deadlock and resource contention deadlock. The approach is evaluated through testing with participants of varying programming experience, demonstrating significant improvement in understanding and retention compared to traditional methods. The project was developed collaboratively by team members Jash and Nishtha on GDevelop within a two-month timeframe. Future scope includes expanding the project to cover more programming concepts and targeting a wider audience, incorporating advanced gamification techniques for enhanced effectiveness.

Keywords: multithreaded processes, educational game, gamification, project results, challenges, game testing, interactive learning, technical concepts.

INTRODUCTION

Multithreaded programming has become an essential tool for developing high-performance software applications. However, its complexities can pose challenges for programmers seeking to understand and implement its principles effectively. To address this need, a gamification project was developed to introduce users to multithreaded programming concepts in an engaging and interactive manner. The gamified learning experience comprises five levels, each focusing on a specific aspect of multithreaded programming.

Our gamified approach comprises a series of five carefully designed levels, each tailored to unravel a distinct facet of multithreaded programming. Through engaging gameplay mechanics and interactive challenges, players will delve into the fundamentals of thread creation, explore thread synchronization techniques, and navigate the intricacies of mutex deadlocks and resource contention deadlocks.

Each level is carefully made to introduce a single concept at a time, ensuring that users can fully grasp each topic before progressing to the next. To further solidify their understanding, players will encounter pop-up text boxes throughout the game, providing additional insights and instruction to navigate through the game.

Whether you're a seasoned programmer seeking to refine your expertise or a novice venturing into the world of coding, our gamification project offers an engaging and effective platform for understanding the intricacies of multithreaded programming better.

MOTIVATION FOR OUR WORK

Traditional teaching methods, while effective in conveying fundamental knowledge, often fail to capture the imagination and spark the curiosity of learners, leading to a lack of engagement and enthusiasm. This observation served as the primary impetus for the development of Thready, a revolutionary learning experience that transforms multithreaded programming into an engaging and interactive journey.

Multithreaded programming, a fundamental pillar of modern software development, involves the execution of multiple tasks concurrently, enhancing the performance and efficiency of applications. However, its complexities can be daunting for novice programmers, often perceived as abstract and lacking in practical applications. This perception stems from the traditional teaching approach that emphasizes theoretical concepts over practical applications, failing to connect the dots between multithreading principles and real-world scenarios.

Thready seeks to bridge this gap by transforming multithreaded programming into an immersive and captivating learning experience. Inspired by the captivating world of retro games, we have crafted a pixelated, top-down universe where players embark on a quest to master the intricacies of multithreading. Through engaging gameplay mechanics and stimulating challenges, players are tasked with applying multithreading principles to overcome obstacles, solve puzzles, and achieve objectives.

This innovative approach addresses the shortcomings of traditional teaching methods by:

1. Promoting Active Engagement: By immersing players in an interactive environment, Thready fosters active participation and stimulates their cognitive processes, maximizing knowledge acquisition and retention.
2. Fostering Curiosity and Motivation: The captivating gameplay mechanics and engaging challenges ignite curiosity and spark motivation, transforming learning from a passive process into an enjoyable and rewarding experience.
3. Demonstrating Practical Applications: By embedding multithreading principles within the gameplay, players experience the direct impact of these concepts, recognizing their relevance and practical applications in the real world.
4. Enhancing Understanding through Visualization: The incorporation of visual representations, such as diagrams and animations, simplifies complex concepts and enhances conceptual understanding.

Thready not only enhances the teaching of multithreaded programming but also lays the foundation for a broader transformation of computer science education. By leveraging the power of gamification, we can revolutionize the way we approach complex concepts, fostering a deeper engagement, understanding, and enthusiasm among future generations of programmers.

CONCEPTS EXPLAINED IN THREADY

Level 1: Understanding Multithreaded Processes

In the world of computers, a program is a set of instructions that guides a computer to perform specific tasks. Traditionally, programs executed in a single-threaded manner, meaning they could only handle one instruction at a time. This approach can be inefficient, especially for programs that involve multiple simultaneous tasks.

Multithreaded programming emerges as a solution to this challenge. It enables a program to execute multiple threads concurrently. A thread is a lightweight unit of execution within a process that can be managed independently. Threads share the resources of the process, such as memory and the CPU, but they can execute independently, enhancing overall efficiency.

Level 2: Thread Creation

Thread consumes fewer resources in the process to create and exist; thread shares process resources. The main thread in Java is the thread that is launched when the program is launched. As a result of the main thread, the slave thread is established.

Level 3: Thread Safety

Thread safety is a crucial aspect of multithreaded programming. When multiple threads access and modify shared data, there's a risk of data corruption or race conditions. A race condition occurs when two or more threads attempt to modify the same data simultaneously, leading to unpredictable outcomes.

To safeguard against data corruption and race conditions, synchronization techniques play a vital role. Synchronization ensures that only one thread can access a shared resource at a time.

Level 4: Thread Synchronization

Thread synchronization is the process of coordinating the actions of multiple threads to ensure they cooperate and function correctly. Synchronization is essential for preventing data corruption and race conditions.

Locks: Locks are the most common synchronization tool. They come in two forms: mutexes and semaphores.

Mutexes: A mutex is a mutual exclusion lock, meaning that only one thread can hold a mutex at a time. Once a thread acquires a mutex, no other thread can acquire it until the first thread releases it.

Semaphores: A semaphore is a generalization of a mutex. It can be used to control access to a limited number of resources. For instance, a semaphore could be used to manage access to a pool of database connections.

Level 5: Mutex Deadlocks and Resource Contention Deadlocks

A deadlock occurs when two or more threads are waiting for each other to release a resource that they both need. Deadlocks can arise when threads are synchronized using locks or semaphores.

Mutex Deadlock: A mutex deadlock occurs when two or more threads are each holding a mutex that the other thread requires.

Resource Contention Deadlock: A resource contention deadlock occurs when two or more threads are each waiting for a limited resource, such as a database connection.

LEVEL-WISE EXPLANATION

Overview

- Each level completion represents one program being executed and completed.

- Tracy and Diego, the players, represent the two threads that portray the functionalities of multithreaded processes in the different levels.

Level 1 – Understanding Multithreaded Processes

- Tracy and Diego traverse the path to complete the level.
- Only when both have reached the endpoint, the game progresses to the next level. This shows the multiple threads in a process needed to complete a process.

Level 2 - Thread Creation

- Use of power-ups: red bolt to slow them down and green bolt to speed them up.
- The power-ups represent slave threads of the main threads, which are Tracy and Diego.
- The power-ups (slave threads) help the players clear the level.

Level 3 - Thread Safety

- Tracy and Diego have to protect their objects (the flags) that the other character cannot use.
- They do this by attacking and killing the other to protect their objects.
- The players end the level by collecting the other's flag.

Level 4 – Thread Synchronization

- The players are put in a maze.
- Both work together simultaneously but one by one to collect keys to open the door to the other side of the river and clear the level.

Level 5 – Deadlocks

- The players are put in a maze again but with a race element this time.
- There is an advantageous shortcut path that, once one of them uses, the path gets blocked, and the other player cannot use it.
- This situation represents a resource and mutex deadlock.

TESTING AND RESULTS

Initial Idea

The initial idea for Thready was to create a single game scene that would explain all the functionalities of multithreaded processes. However, after pitching this idea to some computer science juniors, they felt that it was a little too confusing to understand everything in one scene. Additionally, we realized that the functionalities of multithreaded processes are quite complex to explain in such a limited format.

Mid-Term

Based on the feedback from the juniors, we decided to switch to a multi-level approach, with each level explaining one core concept of multithreaded processes. We had originally envisioned a car racing game where the player would control multiple cars simultaneously, demonstrating the concept of multithreading. However, upon starting development, we realized that GDevelop, the game development platform we were using, had limited assets for creating a car racing game.

Final Game

Due to the limitations of GDevelop, we decided to pivot to a top-view 2D game with people as players. Initially, the game did not have text boxes for explanations, and upon testing, we received feedback that players were unable to understand how to navigate through the game without verbal explanations.

Responding to this feedback, we added text boxes to make the game self-explanatory. After testing with both computer science and non-computer science kids, we found that:

- Computer science kids were better able to understand the topics after seeing a visual representation of it.
- Non-computer science kids were able to grasp the concept and get a basic understanding of what a multithreaded process is and its various functionalities.

In conclusion, the development of Thready involved an iterative process of brainstorming, prototyping, testing, and refining. Our willingness to adapt to feedback and limitations led to the creation of a game that effectively explains the concepts of multithreading to both computer science and non-computer science students.

LIMITATIONS

- Platform Restrictions: GDevelop, the game development platform used to create Thready, has certain limitations that impacted the game's scope and capabilities.
- Limited Development Time: The time constraints faced during Thready's development may have affected the depth and comprehensiveness of the gameplay experience.
- Knowledge Limitations: The developers' knowledge and experience with multithreading concepts and game development techniques may have influenced the game's design and implementation.
- Limited Learning Resources: The availability of resources for learning GDevelop could have hindered the developers' ability to fully utilize the platform's capabilities and create a more polished and feature-rich game.

LEARNINGS

Learning GDevelop

- Understanding the platform's features and limitations: Familiarizing oneself with GDevelop's capabilities, including its scripting language, asset creation tools, and level design features, is essential for effectively using the platform.
- Exploring tutorials and documentation: Utilize GDevelop's official documentation and user-created tutorials to gain a comprehensive understanding of the platform's features and functionalities.
- Practicing with sample projects: Engage in hands-on practice by working on sample projects or following tutorials to gain practical experience in using GDevelop's tools and techniques.

Ideation of the Game

- Brainstorming and refining concepts: Collaborate with team members to generate a wide range of ideas for the game's narrative, characters, gameplay elements, and educational content.

- Prototyping and testing: Create prototypes of the game's core mechanics to gather feedback from potential players and refine the gameplay experience.

Better Understanding of Multithreaded Processes

- Thorough research and exploration: Conduct in-depth research on multithreading concepts, including thread creation, synchronization techniques, and deadlock prevention.
- Applying concepts to the game: Effectively translate multithreading concepts into engaging gameplay mechanics that reinforce learning objectives.

Working as a Team

- Effective communication and collaboration: Establish clear communication channels within the team to share ideas, delegate tasks, and provide feedback.
- Respecting diverse perspectives: Value the input and contributions of team members with different backgrounds and expertise to create a more well-rounded game.
- Resolving conflicts constructively: Address conflicts in a respectful and professional manner, focusing on finding solutions that benefit the team and the project as a whole.

Time Management

- Setting realistic goals and deadlines: Establish clear goals for each stage of development and set realistic deadlines to ensure timely completion.
- Prioritizing tasks effectively: Prioritize tasks based on their importance and urgency to make efficient use of available time.

DISTRIBUTION OF WORK

- **Jash Popat**
  - Creation of game scenes
  - Development of demo video
  - Contribution to source code
  - Ideation of game concepts
- **Nishtha Agarwaal**
  - Implementation of game functionalities
  - Preparation of project report
  - Contribution to source code
  - Ideation of game concepts

TIMELINE

- **October**
  - Original ideation and brainstorming of game concepts
  - Familiarization with GDevelop platform and its features
  - Creation of basic game scenes and level layouts
- **November**

- Development of all game levels
- User testing and refinement of gameplay mechanics
- Creation of demo video showcasing the game's features
- Completion of game functionalities and implementation of multithreading concepts
- Preparation of project report summarizing the development process and learnings

FUTURE SCOPE

Thready's potential extends beyond its current form as an educational game. The core concepts and engaging gameplay mechanics can be further developed into a comprehensive learning platform for multithreading. By expanding the game's content to cover more advanced multithreading concepts, introducing multiplayer modes for collaborative learning, and integrating personalized feedback mechanisms, Thready can evolve into a valuable tool for individuals and educators alike. Additionally, exploring the potential for mobile platforms and virtual reality environments could further enhance the game's accessibility and immersive learning experience.

CONCLUSION

Thready successfully conveys the fundamental principles of multithreading through an engaging and interactive gameplay experience. The game's pixelated top-down world and immersive gameplay effectively capture the attention of players, making it an appealing tool for learning about multithreading concepts. While the game's scope was limited due to development time constraints and platform restrictions, it effectively demonstrates the potential of using interactive games to teach complex technical concepts. With further development and expansion, Thready could evolve into a comprehensive learning platform for multithreading, catering to a wider audience and offering a more immersive learning experience.

REFERENCES

- https://docs.oracle.com/javase/tutorial/essential/concurrency/runthread.html
- https://m.youtube.com/watch?v=Yk7BXaotJ24pp=ygUQI3RocmVhZGluZ2luamF2YQ
- https://www.amazon.com/Art-Multiprocessor-Programming-Maurice-Herlihy/dp/0123705916
- https://docs.oracle.com/javase
- https://www.tutorialspoint.com/how-to-create-a-thread-in-java
- https://www.amazon.com/Effective-Java-Joshua-Bloch-ebook/dp/B078H61SCH
- https://docs.oracle.com/cd/E26502$_0$1$/html/E35303/compat-14994.htmlhttps://www.tutorialspoint.com$
$and-states-of-thread-in-java$
- https://www.amazon.com/Java-Concurrency-Practice-Brian-Goetz/dp/0321349601
- https://docs.oracle.com/javase/tutorial/essential/concurrency/sync.html
- https://www.tutorialspoint.com/lifecycle-and-states-of-thread-in-java
- https://www.amazon.com/Java-Concurrency-Practice-Brian-Goetz/dp/0321349601
- https://docs.oracle.com/javase/tutorial/essential/concurrency/deadlock.html
- https://www.tutorialspoint.com/deadlock-prevention
- https://www.amazon.com/Java-Concurrency-Practice-Brian-Goetz/dp/0321349601

Figure 1. Initial Screen

Figure 2. Tracy

Figure 3. Diego

Figure 4. Level 1 Scene

Figure 5. Level 2 Scene

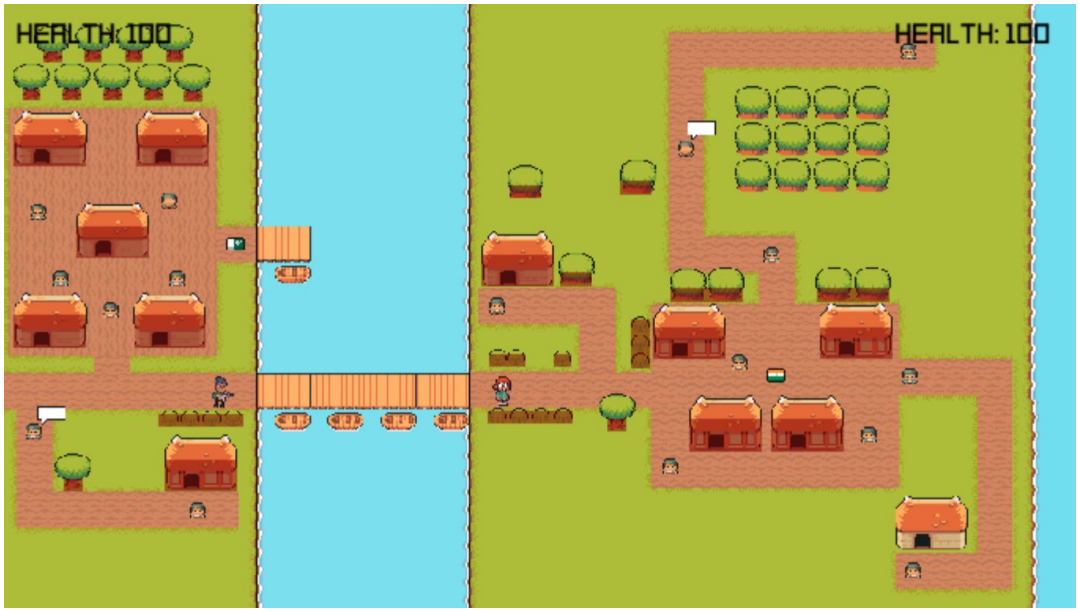

Figure 6. Level 3 scene

Figure 7. Level 4 Scene

Figure 8. Level 5 Scene