RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

ThreadRush: Virtual City Thread Joyride

OPERATING SYSTEM- CS235AI

UNDER THE GUIDENCE OF
Dr.Jyoti Shetty (Assistant Professor)

SAMARTH G(1RV22CS174), RISHAB R(1RV22CS415), SACHIN ANNIGERI(1RV22CS168), SANTHRUPTH H R(1RV22CS416)
DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Abstract

Students studying computer science or related fields face difficulty in understanding thread management concepts in operating systems. This occurs due to the abstract nature of the concepts and the lack of practical demonstrations, leading to challenges in grasping these concepts during the early stages of learning. The issue is commonly observed in educational environments where thread management is taught, affecting students who rely on traditional teaching methods that offer limited visual representations. To address this, there is a need for innovative teaching approaches and interactive learning experiences that provide students with opportunities to explore, experiment, and apply thread management concepts in a more engaging and practical manner.

Introduction

Learning thread management in operating systems can be a bumpy road for students, especially in the beginning. Here's a breakdown of the common challenges:

Abstraction Overload: Thread management deals with concepts that are inherently abstract. Threads are lightweight processes that run concurrently, sharing resources within a program. This can be difficult to visualize without practical demonstrations.
Limited Practical Application: Traditional teaching methods often rely on theoretical explanations and code examples. While valuable, these methods lack the opportunity for students to "get their hands dirty" and experiment with threads themselves. This can lead to difficulty grasping the practical implications of the concepts.
Visual Learning Gap: Textbooks and lectures often lack engaging visual representations of thread behavior. Students who learn best visually might struggle to connect abstract concepts to real-world scenarios.

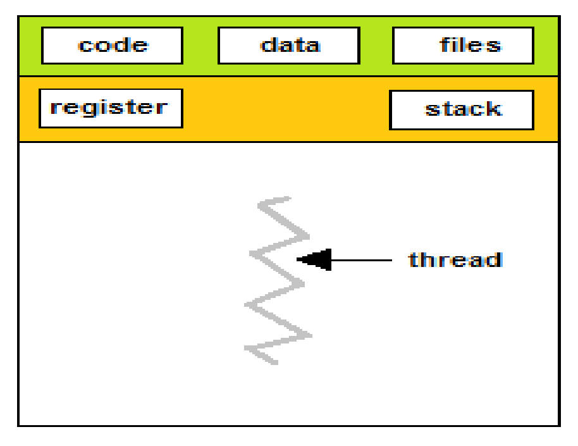
Impact on Students: These challenges can lead to several issues for students:

Difficulty Grasping Core Concepts: Students might struggle to understand how threads work, how they interact, and how to manage them effectively.
Limited Problem-Solving Skills: Without practical experience, students might find it difficult to apply their knowledge to solve real-world thread management problems.

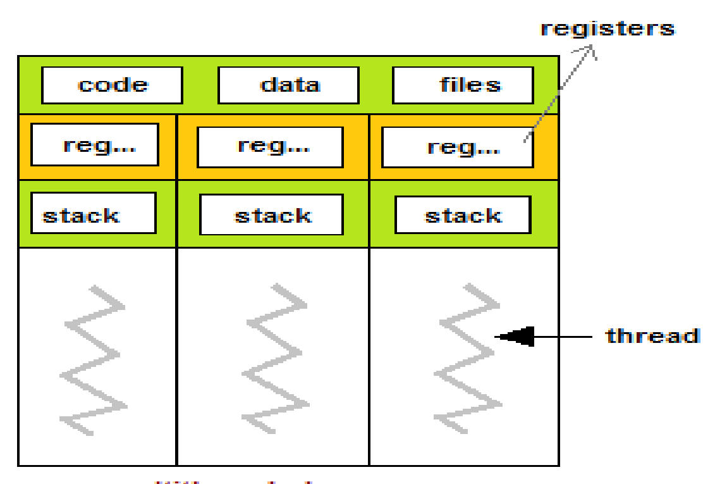
VR Thread City: A Potential Solution

VR Thread City concept has the potential to address these challenges by providing an interactive and visually engaging learning experience. By placing students in a virtual environment where threads are represented as cars navigating a city, you can offer them the opportunity to:
Visualize Thread Behavior: The city metaphor can represent critical concepts like synchronization (traffic lights) and resource sharing (shared roads).
Experiment with Scenarios: Students can create and manipulate threads (cars), observing how they interact and impact the virtual environment.

Apply Thread Management Principles: They can practice managing threads to achieve specific goals within the city, promoting problem-solving skills.
This VR approach has the potential to overcome the limitations of traditional methods and make learning thread management a more engaging and effective experience for students.



single-threaded process



multithreaded process

Relevant Operating System Concept and APIs used:

- **Threads**: Students will be able to visualize threads as cars navigating the virtual city. Each car represents a lightweight process within the program, executing concurrently with other threads.
- **Processes**: The VR environment itself can be considered a single process, spawning multiple threads (cars) to manage different aspects of the city (e.g., traffic control, building construction).
- **Synchronization**: Traffic lights within the city can represent synchronization mechanisms (e.g., mutexes, semaphores) that ensure proper thread coordination. Cars approaching an intersection (critical section) would need to acquire a "green light" (lock) to proceed safely, preventing collisions (race conditions).
- **Resource Sharing**: Roads and shared spaces within the city represent resources that threads (cars) must access concurrently. Students can observe potential issues like deadlocks if multiple threads (cars) try to occupy the same road segment (resource) at the same time.

Potential APIs(Exact APIs syntax as in OS is not used but the program written for the API behaves almost in the same way:

- **Thread Creation and Management:** APIs for creating new threads (spawning cars), terminating threads (removing cars), and manipulating thread priorities.
- **Shared Memory Access:** APIs for accessing and modifying shared memory regions (e.g., simulating data exchange between threads at ific locations within the city).
- **Syspecnchronization Primitives**: APIs for implementing synchronization mechanisms like mutexes, semaphores, or condition variables (represented by traffic lights, yield signs, etc.)

Design of solution to problem

City Background: This is a great visual metaphor for representing a computer system. Various dropdowns on road can represent resources and next location to move, and roads can represent common resources shared.
- **Cars as Threads:** Each car signifies a thread, with its destination representing the thread's function. Movement speed can represent the thread's priority.
- **Traffic Rules:** These represent various OS concepts like synchronization (e.g., traffic lights) and resource management (e.g., limited parking spaces).
- **User Input:** Users can spawn new cars (threads), change their priorities, and introduce obstacles (e.g., road closures) to simulate different scenarios.

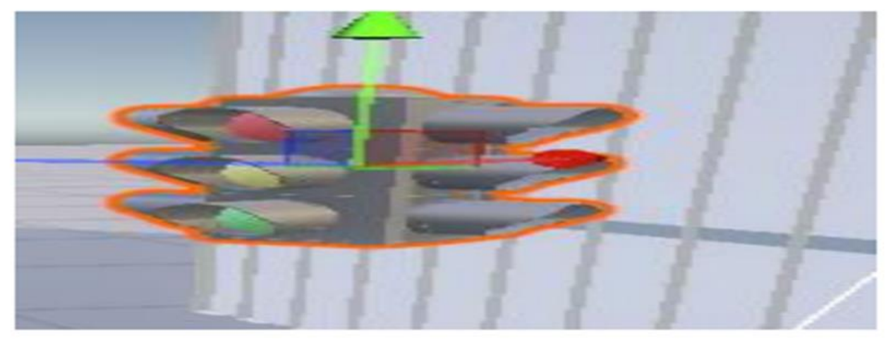
Implementation:

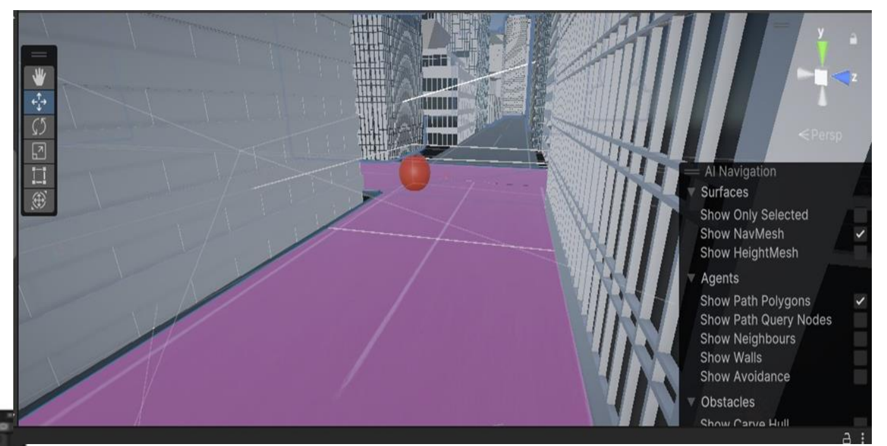
Unity and C#: This is a great choice for VR development. Utilize Unity's physics engine to simulate car movement and handle collisions. C# scripting will control car behavior and implement thread concepts.
Car Movement:
- Defined different car prefabs based on priority levels (e.g., color, size).
- Implemented pathfinding algorithms to guide cars towards their destinations.
- Used Unity's physics engine for realistic movement and collision detection.
- **Traffic Lights:** Implementing traffic lights at intersections to represent synchronization primitives like mutexes or semaphores. Cars need to acquire the "green light" (resource) to proceed.
- **Yield Signs:** Used yield signs at specific locations to represent situations where threads voluntarily relinquish control. Cars slow down and allow others to pass based on programmed logic.

User Interaction & Output Display

- Provide a VR controller or hand tracking to allow users to:
- Spawn new cars (threads) with different priorities.
- Change a car's destination dynamically (thread context switching).
- Introduce obstacles (e.g., road closures) to simulate system events like I/O operations.

- Dashboard: Create a virtual dashboard within the VR environment displaying:
- Information about each car (thread ID, priority, status).
- Resource utilization (e.g., parking space occupancy).
- Performance metrics (e.g., average wait time at traffic lights).
- Text Pop-ups: Display contextual messages above cars to highlight specific OS concepts being demonstrated (e.g., "Acquiring resource lock", "Context switch").





Conclusion

This project has explored the development of VR Thread City, an innovative approach to learning thread management concepts in operating systems. By utilizing virtual reality (VR) and a city metaphor, the project aims to address the challenges students often face in grasping these abstract concepts.

Overall, VR Thread City has the potential to revolutionize the way students learn thread management in operating systems. By leveraging the power of VR and interactive learning, this project can make thread management concepts more accessible, engaging, and ultimately, more understandable for students.