Table of Contents

Objectives

This project aims to simulate real-time 3D fire behavior using OpenGL. The primary goal is to visually replicate fire phenomena through the Navier-Stokes equations, implemented in a 3D grid environment.

Specifically, the project intends :

- To implement fluid simulation steps (Advection, Force Addition, Diffusion, Projection) to drive fire behavior.
- To leverage GPU acceleration via OpenGL to maintain high frame rates and support interactive features.
- To support real-time control over fire sources and camera view, enabling immersive visual feedback.

Introduction

Fire simulation is an active research area in computer graphics, particularly for creating realistic effects in games, virtual environments, and scientific visualization. We base our model on NavierStokes equations, widely used for simulating fluid dynamics, and apply them within a 3D voxel grid.

Each voxel in this grid stores data such as velocity, temperature, and pressure. The fire is modeled as a fluid and evolved through several computational steps. This project utilizes Jos Stam's "Stable Fluids" method, which allows for stable and visually convincing results using semiLagrangian advection and Jacobi iteration for pressure projection.

To render the simulation, we use OpenGL for real-time visualization, where particles represent fire elements and their behavior is mapped to GPU-driven shaders. The user can interactively manipulate the scene, making this simulator useful for both demonstration and learning.

Application

The real-time 3D fire simulator using OpenGL has a wide range of applications across multiple domains. These applications leverage the simulator's realistic fire behavior, interactive control, and GPUaccelerated performance.

1. Game Development and Visual Effects :

Modern games and cinematic experiences require highly dynamic and realistic visual effects to create immersive environments. Traditional methods often use pre-rendered fire animations, which lack adaptability and interactivity.

This simulator can be integrated into game engines (e.g., Unity, Unreal) to generate real-time, interactive fire effects that respond to the player's actions or environment conditions. Explosions, spreading fires, torch flames, or environmental hazards can all be dynamically generated, improving realism and engagement.

Moreover, in visual effects (VFX) production, the simulator enables quick prototyping and realistic previews without time-intensive rendering.

2. Fire Safety and Emergency Training :

Training firefighters and emergency response teams in real-world fire scenarios can be dangerous, costly, and logistically complex. This simulator allows for safe and repeatable training environments where fire behavior can be visualized in diverse 3D settings (e.g., houses, factories, forests).

Using the interactive controls, instructors can simulate different fire sources and analyze how flames spread under various conditions such as air ventilation or structural layout. The simulator can be further extended to simulate smoke density, temperature gradients, and escape route visibility to aid in realistic preparedness training.

3. Virtual and Augmented Reality (VR/AR) :

The system can be embedded into VR and AR platforms to provide immersive experiences for education, training, and simulation. In a VR headset, users can walk through a burning building and experience realistic fire visuals, helping them understand evacuation strategies or the effects of fire.

This is particularly useful in:

- Fire drill simulations in schools and offices.
- Augmented reality educational apps explaining fire dynamics.
- Mixed reality hazard simulations in industrial safety training.

4. GPU Programming and Academic Learning :

The simulator serves as a practical example of advanced GPU programming. It introduces students and researchers to:

- Navier-Stokes equations for fluid dynamics
- Compute shader development
- Parallel computing
- Real-time rendering with OpenGL

It acts as an educational tool for exploring numerical methods and graphics optimizations, allowing learners to study and modify real-time simulations with immediate feedback.

5. Experimental Research and Simulation Platforms :

For researchers exploring fluid mechanics, computational physics, or graphics techniques, the simulator offers a customizable environment where new ideas can be implemented and tested. With mouse-based interaction, users can dynamically reposition fire sources, manipulate simulation resolution, or visualize heat distribution.

This flexibility makes it ideal for:

- Academic experimentation
- Prototype testing for visual computing
- Physics-based simulation platforms

Features

Starting Screen:

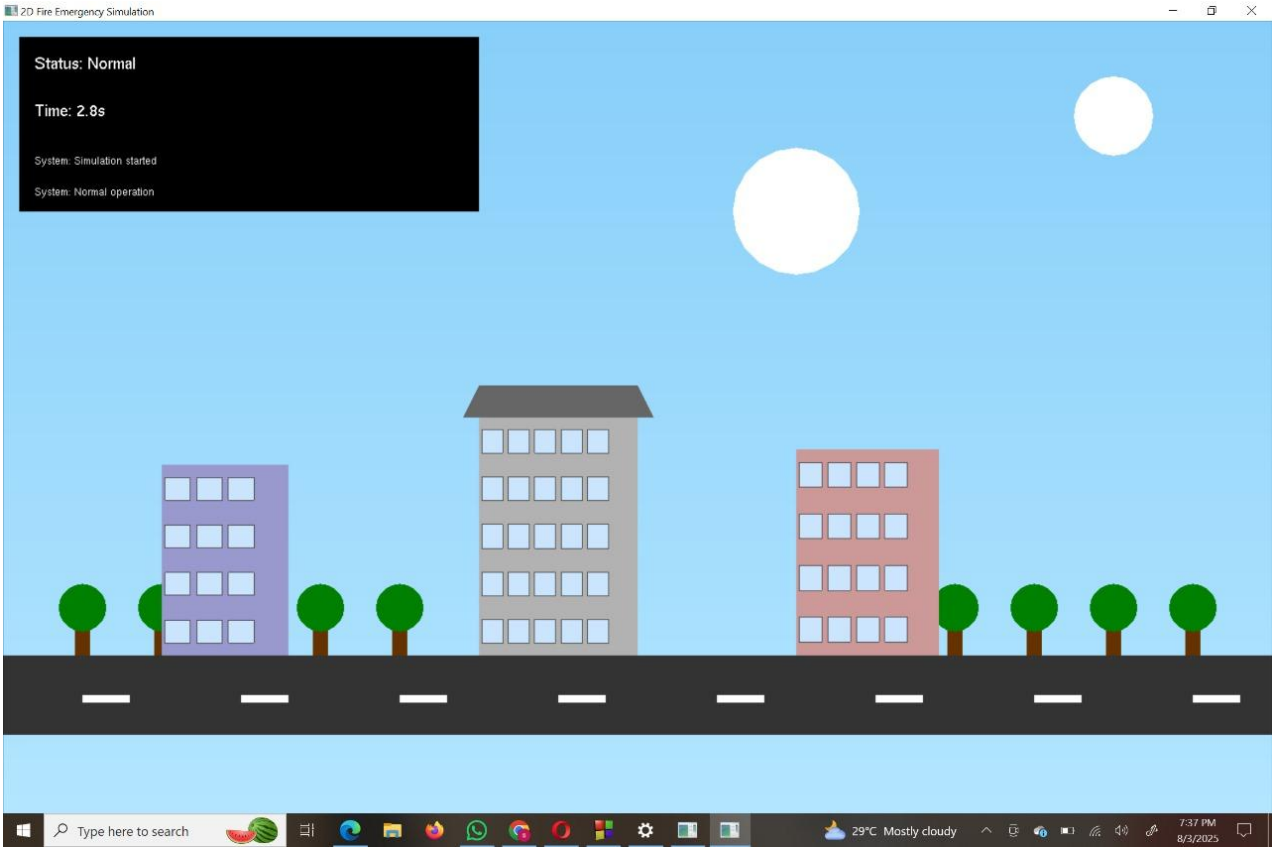

Figure 1: First Scenerio

The system is composed of multiple interconnected modules that handle fire simulation and rendering in real time.

1. House:

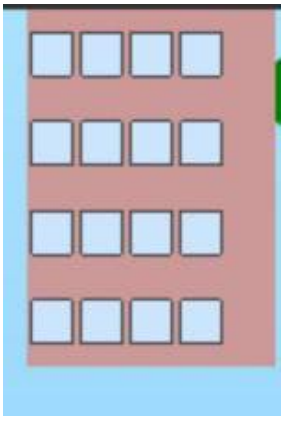

Figure 2 : Building

The function `drawBuilding` is responsible for rendering buildings in the simulation scene. It takes parameters that define the building's position, size, number of floors, and windows. Using OpenGL, the function first draws a rectangle to form the building structure with a specific color based on the building index. It then uses nested loops to draw windows on each floor. The windows are represented as smaller rectangles, and their color dynamically changes if a fire is active in that particular window, creating a glowing effect. Additionally, the function draws a roof for the main building (index 0), distinguishing it from the other side buildings. This function helps create the visual impression of a multi-story building with multiple windows, enabling the simulation of fire spreading within a structure.

2. Cloud:

Figure 3: Cloud

.The function `drawCloud` is used to create visual representations of clouds in the sky. It takes the position `(x, y)` and `size` as parameters to determine the location and scale of each cloud. Using OpenGL, the function draws a circular shape by plotting multiple vertices in a loop, forming a filled polygon. The color is set to white to simulate the natural appearance of clouds. By calling this function multiple times with different positions and sizes, the sky background is made more dynamic and realistic, enhancing the overall atmosphere of the scene.

3. Fire:

Figure 4: Fire

The `drawFire` function visually represents the fire effect in the simulation. It works by iterating through a collection of fire particles, each with individual properties such as position, size, life, and color. Depending on a particle's remaining life, it is colored with different fire shades—orange, yellow-orange, or gray—to simulate flames and smoke. The particles are rendered using OpenGL's `GL_POINTS` with blending enabled, allowing for a glowing and transparent fire effect. This function is only active when the simulation state indicates an ongoing fire. Overall, it creates a dynamic and realistic visual of burning windows, contributing significantly to the emergency scene.

4. Trees:

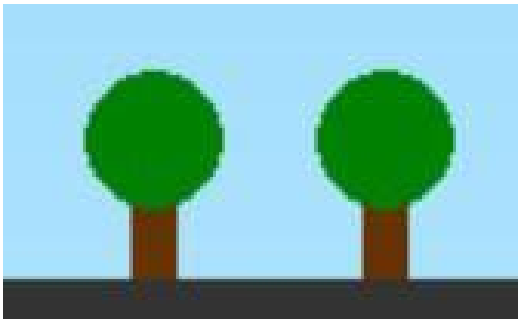

Figure 5: Trees

The `drawTrees` function is responsible for rendering trees along the roadside in the simulation environment. Using a loop, it positions multiple trees at regular intervals, excluding areas blocked by buildings. Each tree is composed of two parts: a brown rectangular trunk and a circular green foliage canopy, both drawn using OpenGL primitives. The circular canopy is formed using a polygon with multiple vertices to simulate the roundness of tree leaves. By repeating this process across the scene, the function adds natural elements and enhances the visual realism of the environment.

5. Road:

Figure 6: Road

The `drawRoad` function creates the visual representation of the main road in the simulation. It begins by drawing a large rectangular area using OpenGL to represent the asphalt surface, filled with a dark gray color. To simulate road markings, it then draws evenly spaced white rectangles along the center of the road. These horizontal dashes help depict a standard road layout. The road serves as a base layer in the scene and provides context for the movement of fire trucks and the arrangement of buildings and trees, enhancing the overall urban setting.

6. Fire Truck:

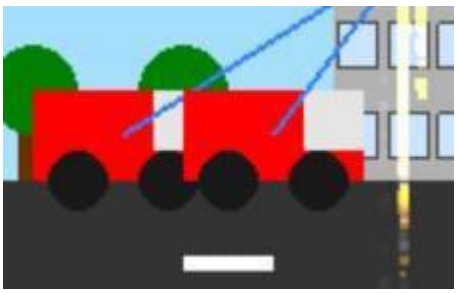

Figure 7: Fire Truck

The `drawFireTruck` function is used to visually render fire trucks in the simulation scene. Each truck is drawn using multiple OpenGL primitives: the main body as a red rectangle, the cabin as a lighter gray rectangle, and the wheels as circular polygons. The function also includes logic to visually represent a water hose when the truck is in the "spraying" state. The hose is shown as a blue line extending from the truck to the burning window, simulating the act of extinguishing a fire. This dynamic rendering reflects the truck's current role in the emergency response and contributes to the realism of the firefighting scenario.

7. Human:

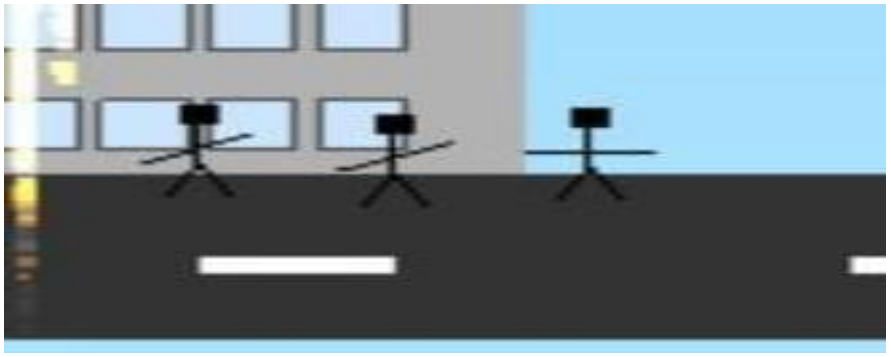

Figure 8: Human

The `drawHumans` function depicts human figures arriving at the emergency scene. It draws multiple simple stick figures consisting of a circular head and lines for the body, legs, and arms. The arm movement is animated with rotation to simulate walking motion. This function activates when the simulation state indicates the arrival of emergency personnel or civilians. The presence and animation of humans add life and realism to the scene, emphasizing the emergency response aspect of the simulation

8. Sky:

Figure 9: Sky

The `drawSky` function creates a gradient background representing the sky. It uses OpenGL quadrilaterals with smoothly varying colors from a deeper blue at the top to a lighter blue near the horizon. This gradient effect simulates the natural appearance of the sky and sets the overall atmospheric tone of the scene. It forms the backdrop over which all other elements like clouds, buildings, and trees are rendered.

Normal:

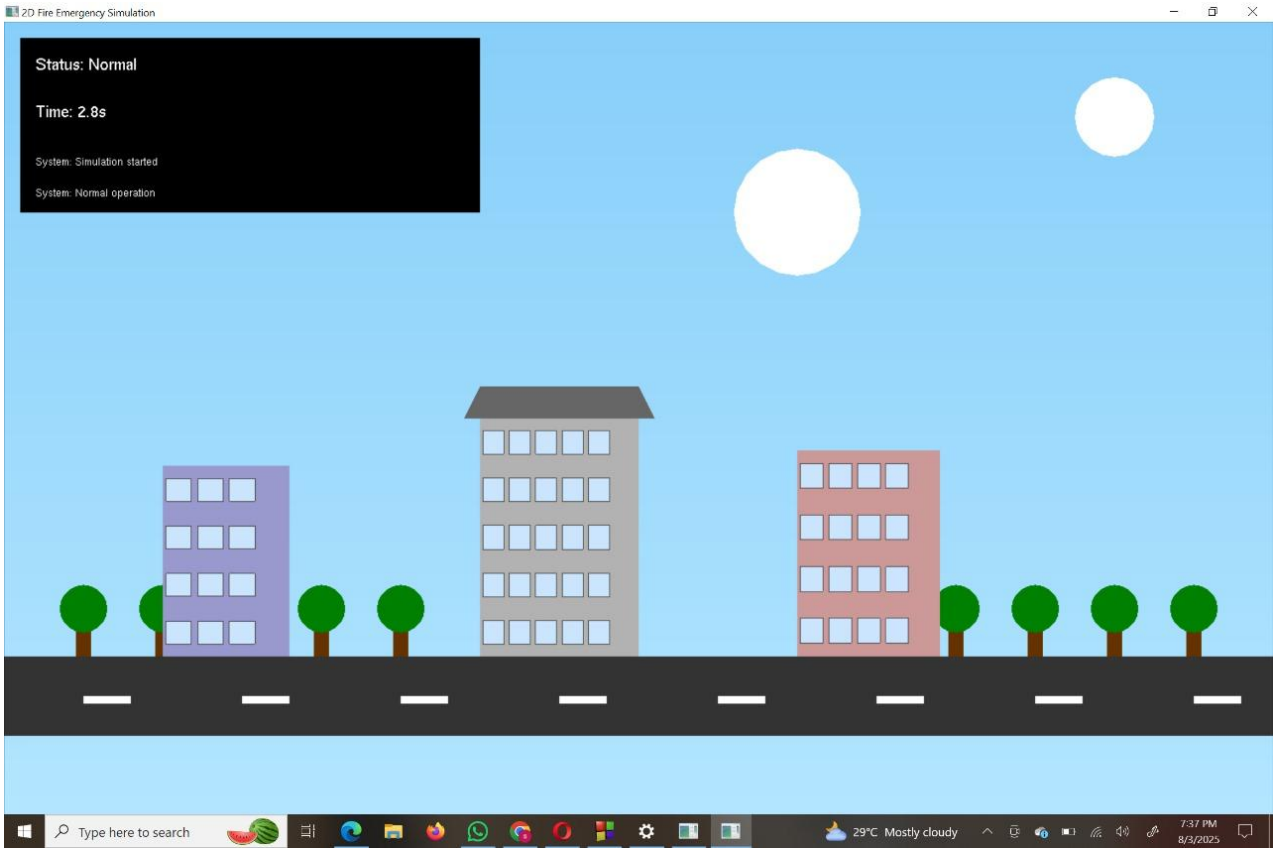

Figure 10 : Normal Situation

The normal situation is represented by the initial simulation state where no fire or emergency activity is present. During this state, the scene is rendered with all static elements such as buildings, trees, road, and sky in their default colors and positions. No fire particles or alarms are drawn, and emergency personnel and fire trucks remain off-screen. The system status is updated to "Normal" and displayed in the interface panel, reflecting that the simulation is in a calm and stable condition prior to any incident.

Fire detected:

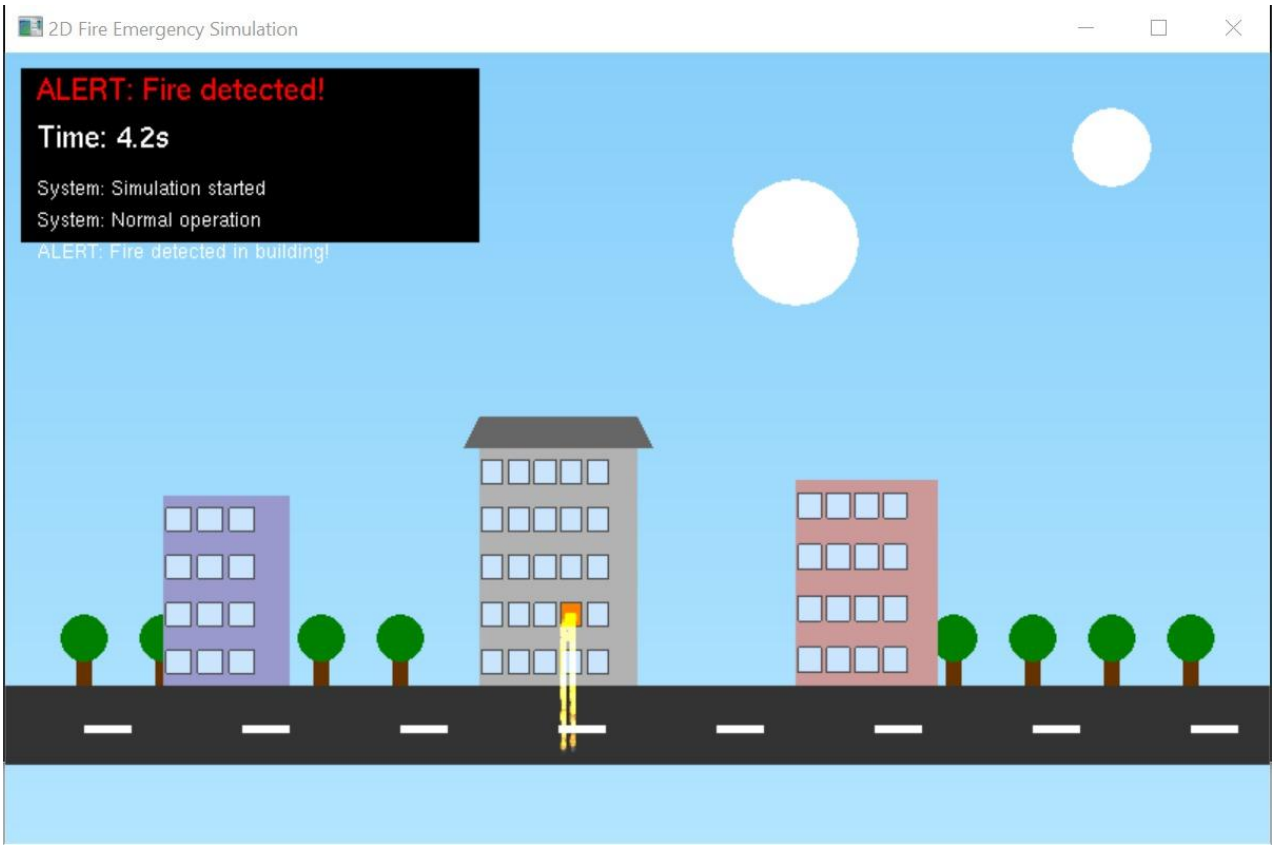

Figure 11 : Fire detected

The fire detected state triggers when a fire incident is first identified in the simulation. At this point, one of the building windows is randomly selected to be on fire. The `drawBuilding` function highlights this window with a fiery glow, while the `updateFireParticles` and `drawFire` functions begin generating and rendering fire particles that simulate flames and smoke emerging from the burning window. Simultaneously, the system status changes to alert mode, updating the interface panel to display “ALERT: Fire detected!” in red text. This state marks the start of the emergency sequence, visually and functionally signaling the onset of a fire event.

ALERT : Alarm activated :

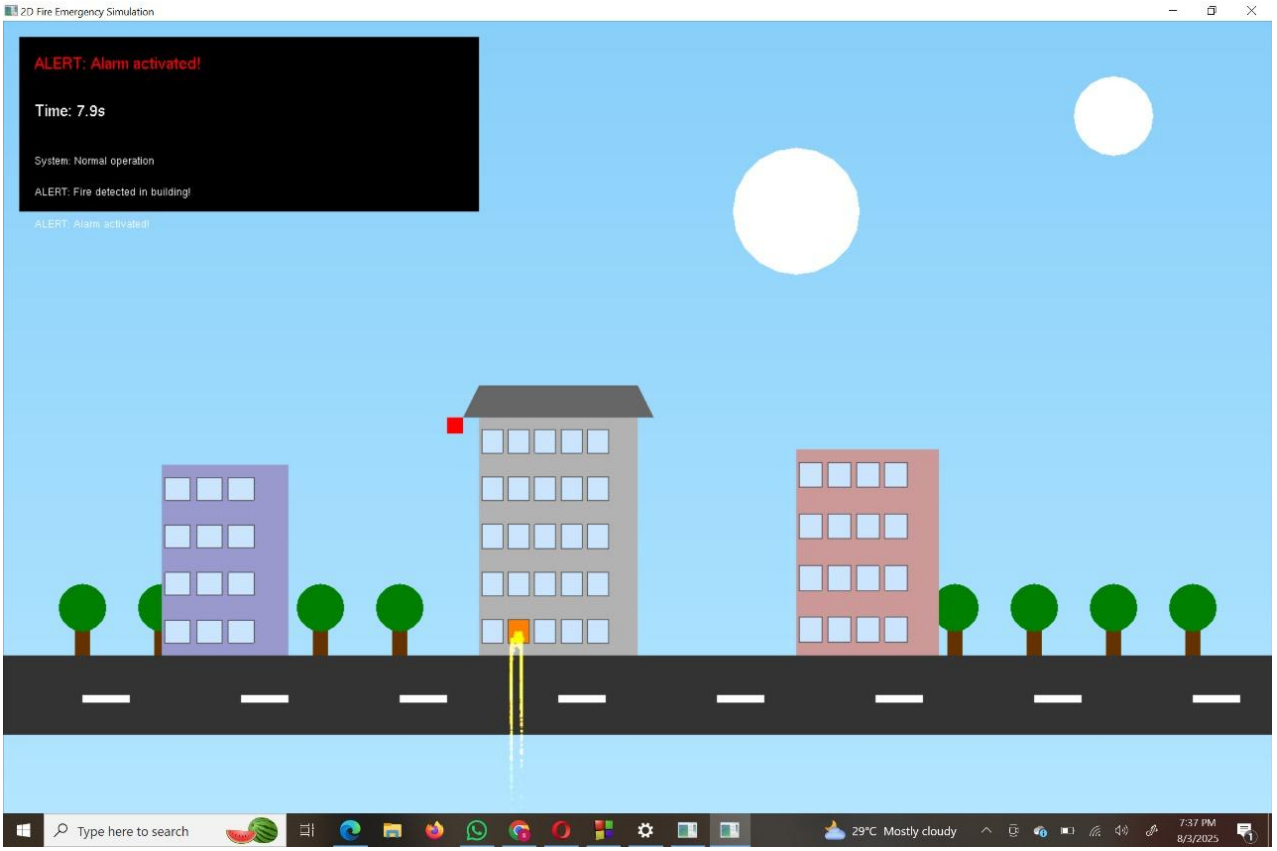

Figure 12: ALERT : Alarm activated

The alarm activated state begins shortly after the fire is detected, signaling an escalation in the emergency response. During this state, the simulation activates a blinking red alarm indicator to visually alert the user. The `drawAlarm` function manages this effect by rendering a red square that intermittently appears and disappears, simulating a flashing alarm light. Meanwhile, the interface panel updates its status message to “ALERT: Alarm activated!” in red text to emphasize urgency. This state prepares the scene for the arrival of emergency crews and heightens the sense of emergency within the simulation.

Emergency crew arriving:

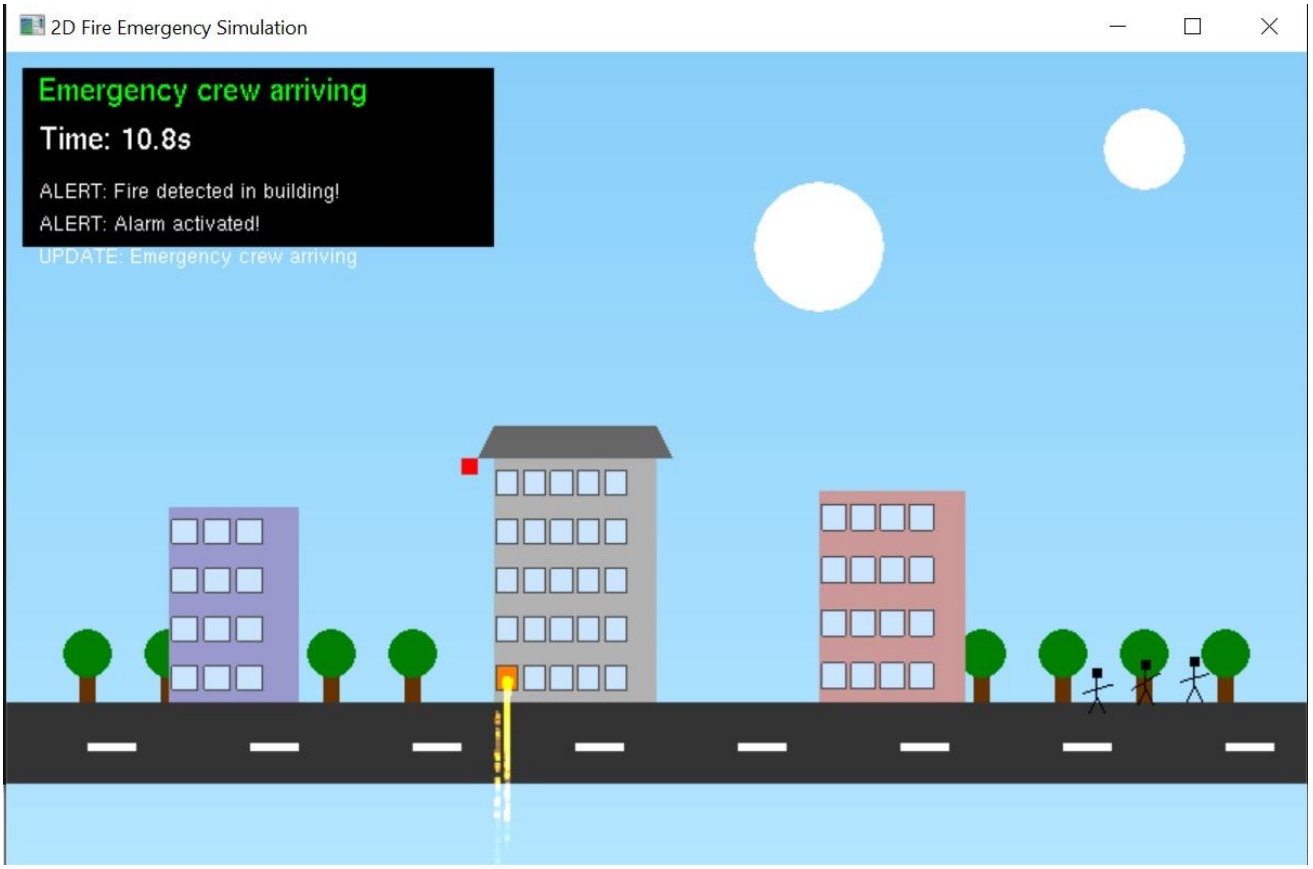

Figure 13: Emergency crew arriving

The emergency crew arriving state represents the initial response phase following the alarm activation. In this state, human figures symbolizing emergency personnel are animated to move toward the scene from the right side of the screen. The `drawHumans` function draws these stick figures with simple animations simulating walking motion. Their position is updated gradually to simulate arrival. Concurrently, the interface panel displays the status message “Emergency crew arriving” in green, indicating the progression of the emergency response. This state enhances the realism of the simulation by showing active human involvement in managing the fire incident.

Firefighters arriving:

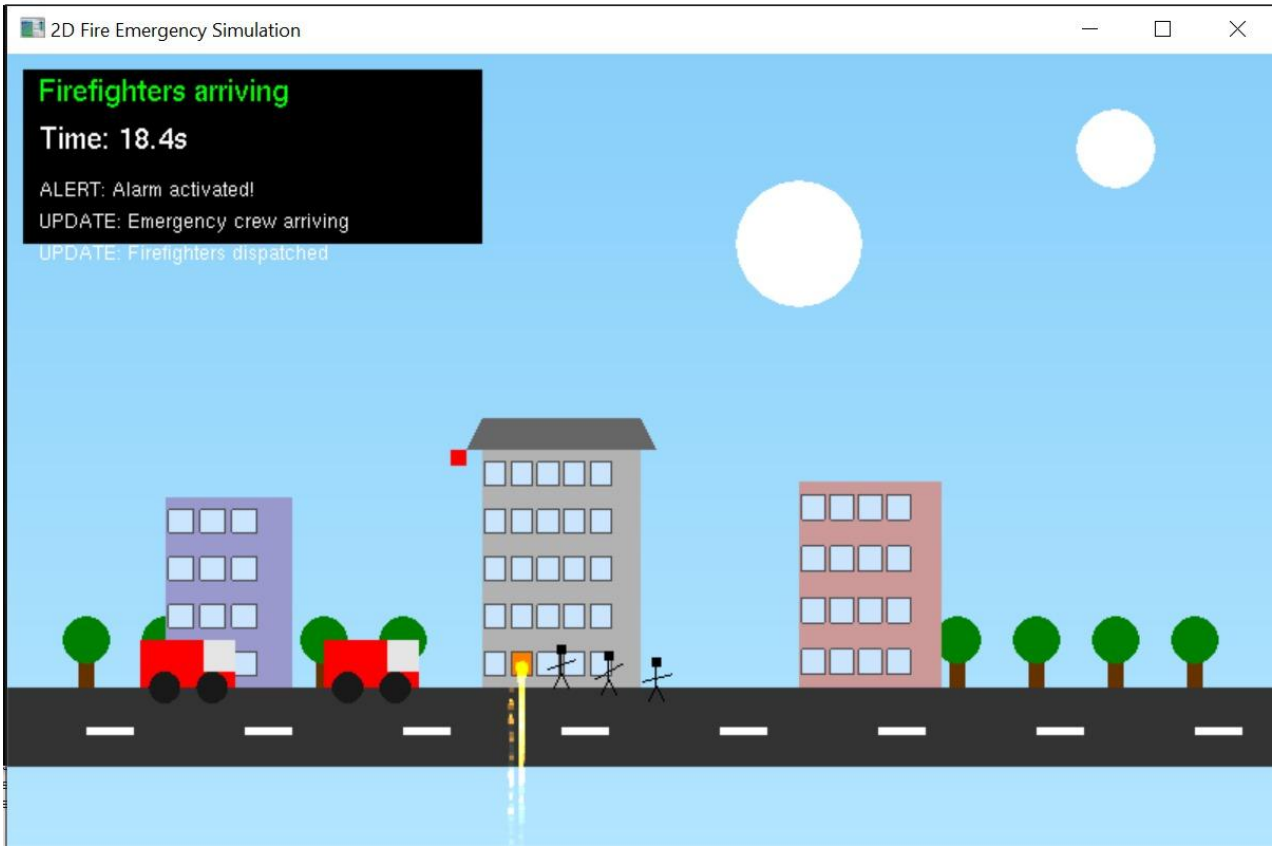

Figure 14: Firefighters arriving

The firefighters arriving state signifies the deployment of specialized emergency responders to the fire scene. In this phase, two fire trucks progressively move onto the screen toward the burning building. The `updateFireTrucks` function manages their movement, controlling their position and arrival status. Once the trucks reach their designated positions, the state transitions to enable firefighting actions. The interface panel updates the status message to "Firefighters arriving" in green, indicating that the firefighting team is now actively engaging with the emergency. This state visually and functionally bridges the initial emergency alert with active fire suppression efforts.

Extinguishing fire:

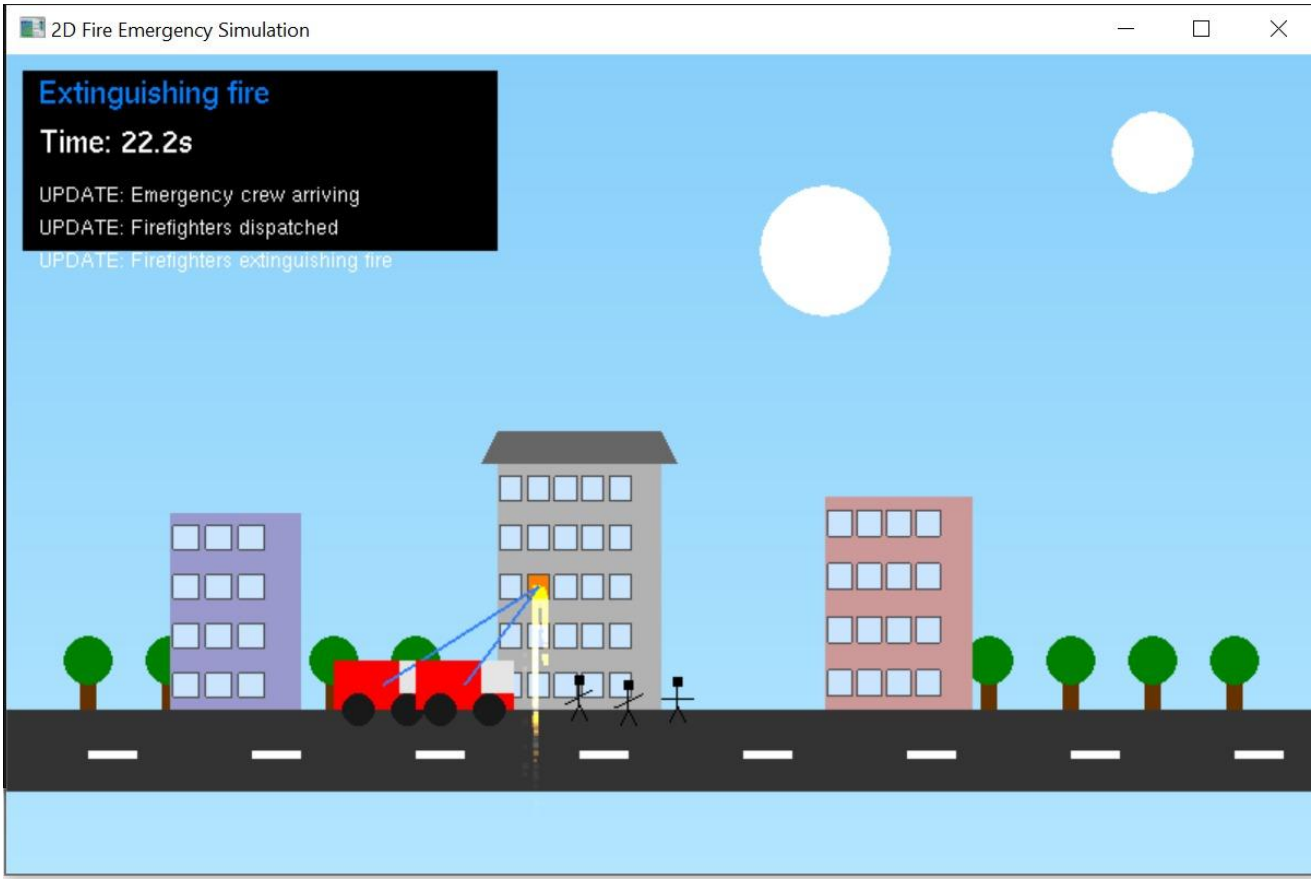

Figure 15: Extinguishing fire

The extinguishing fire state represents the active firefighting phase in the simulation. During this state, the fire trucks remain stationary near the burning building, and their water hoses are visually extended toward the burning window using the `drawFireTruck` function. Simultaneously, the `updateFireParticles` and `drawFire` functions gradually reduce the fire particles to simulate the fire being put out. The interface panel reflects this state by displaying the message “Extinguishing fire” in blue, indicating ongoing suppression efforts. This dynamic interaction between the trucks and fire particles creates a realistic depiction of fire extinguishing.

ALL CLEAR - Fire out:

Figure 16: ALL CLEAR - Fire out

The "All Clear" state signifies the successful extinguishment of the fire in the simulation. During this phase, all fire particles are removed, and no fire effects are rendered on the building. The fire trucks stop spraying water and remain stationary for a brief period. The alarm and emergency indicators are deactivated. The interface panel updates the status message to “ALL CLEAR - Fire out” in green, signaling that the emergency has been resolved. This state marks the conclusion of the firefighting sequence and a return to normalcy in the scene.

Conclusion

This 3D fire simulator combines physical accuracy and visual realism using OpenGL and fluid simulation principles. It delivers a responsive and immersive experience that is computationally optimized for real-time rendering. By integrating mathematical models, parallel GPU processing, and interactive rendering, the system serves as a valuable tool for

applications in education, entertainment, and safety training. The project is expandable to include features like smoke modeling, heat maps, and fire-environment interactions, making it a solid foundation for future development in physically-based simulations.