FPGA-Driven 'Forest Fire and Ice' Game: Design and Implementation

Yifan Mao
ym3064@columbia.edu

Weijie Wang
ww2739@columbia.edu

Yonghao Lin
yl5763@columbia.edu

Yang Cao
yc4535@columbia.edu

Zhenqi Li
zl3508@columbia.edu

1 Introduction

This project aims to implement a simplified version of the popular game "Forest Fire and Ice" on the FPGA (Field-Programmable Gate Array) platform. The game will feature two characters, Fire Boy and Water Girl, navigating through a series of levels filled with various obstacles, traps, and puzzles. The game will be rendered in a 2D wireframe style, with the FPGA handling the rendering of the game environment, character animations, and physics calculations such as collision detection, speed, and acceleration.

The game will be controlled via a USB input device (e.g. keyboard), and the display will be output to a VGA monitor at a resolution of 640x480 pixels. The game will run at 60 frames per second, ensuring smooth game-play.

2 Game Levels and Features

1. Map
   - **Description**: The game will feature a series of levels, each with a unique map layout. The map will be represented as a 2D grid, with each cell containing information about the terrain, obstacles, and interactive elements.
   - **FPGA Implementation**: The map data will be stored in on-chip memory, and the FPGA will render the map in real-time using a wireframe representation. The map will be scrollable, allowing the player to navigate through the level.
2. Trigger Platforms
   - **Description**: Certain platforms in the game will only appear or move when triggered by the player (e.g., stepping on a switch or pressing a button).
   - **FPGA Implementation**: The FPGA will handle the logic for detecting when a trigger is activated and will update the platform's position accordingly. The platform's movement will be synchronized with the game's frame rate.
3. Fixed Traps
   - **Description**: The game will include fixed traps such as spikes, fire pits, and water pools that will harm the player if touched.
   - **FPGA Implementation**: The FPGA will detect collisions between the player and the traps. If a collision is detected, the player's health will be reduced, and the game will respond accordingly (e.g., resetting the player's position or ending the level).
4. Light Reflection Mechanisms
   - **Description**: Some levels will feature light reflection puzzles where the player must redirect light beams using mirrors to unlock doors or activate mechanisms.
   - **FPGA Implementation**: The FPGA will calculate the trajectory of light beams based on the position and orientation of mirrors. The light beams will be rendered in real-time, and the FPGA will detect when a beam hits a target (e.g., a door or switch).

5. Seesaws
    - **Description**: Seesaws will be used to create dynamic platforms that move based on the player's weight distribution.
    - **FPGA Implementation**: The FPGA will simulate the physics of the seesaw, including the balance and movement based on the player's position. The seesaw's movement will be synchronized with the game's frame rate.
6. Pulleys
    - **Description**: Pulleys will be used to create moving platforms or to lift heavy objects.
    - **FPGA Implementation**: The FPGA will handle the physics of the pulley system, including the tension and movement of ropes. The pulley's movement will be synchronized with the game's frame rate.
7. Portals (Customizable)
    - **Description**: Portals will allow the player to teleport between different locations in the level. The portals can be customized to have different behaviors (e.g., one-way portals, timed portals).
    - **FPGA Implementation**: The FPGA will handle the logic for detecting when the player enters a portal and will update the player's position accordingly. The portal's behavior will be customizable via software.
8. Score Points (Red, Blue, Green Diamonds)
    - **Description**: The game will feature collectible diamonds (red, blue, and green) that the player can collect to earn points.
    - **FPGA Implementation**: The FPGA will detect when the player collides with a diamond and will update the player's score accordingly. The diamonds will be rendered in real-time, and their positions will be stored in on-chip memory.
9. End Points (Red and Blue Doors)
    - **Description**: Each level will have two endpoints: a red door for Fire Boy and a blue door for Water Girl. Both characters must reach their respective doors to complete the level.
    - **FPGA Implementation**: The FPGA will detect when both characters reach their respective doors and will trigger the level completion logic. The doors will be rendered in real-time, and their positions will be stored in on-chip memory.

3 Additional Design Elements

1. Character Animation
    - **Description**: The characters (Fire Boy and Water Girl) will have simple animations for walking, jumping, and interacting with objects.
    - **FPGA Implementation**: The FPGA will handle the rendering of character animations using a sprite-based approach. The animation frames will be stored in on-chip memory, and the FPGA will cycle through the frames based on the character's actions.
2. Speed and Acceleration
    - **Description**: The characters will have realistic speed and acceleration, allowing for smooth movement and jumping.
    - **FPGA Implementation**: The FPGA will simulate the physics of movement, including acceleration, deceleration, and gravity. The character's position will be updated in real-time based on these calculations.
3. Collision Detection
    - **Description**: The game will feature accurate collision detection between the characters, obstacles, and interactive elements.
    - **FPGA Implementation**: The FPGA will handle collision detection using bounding box checks. When a collision is detected, the game will respond accordingly (e.g., stopping the character's movement, triggering an event).

4 Hardware-Software Interface

The game will use a memory-based hardware-software interface, where the software (running on HPS) will generate the game state (e.g., character positions, object states) and store it in shared memory. The FPGA will read this data and render the game in real-time. The HPS will handle user input (via USB) and FPGA will synchronize the game's frame rate with the VGA display.

5 Major Tasks

1. **Design Document**: Create a detailed design document outlining the game's overall architecture, clearly defining the responsibilities of the software (HPS) and hardware (FPGA) subsystems. The FPGA's role includes graphics rendering acceleration, physics calculations, and collision detection.

2. **Software Prototype (HPS)**: Develop an initial software prototype running on the DE1-SoC's Hard Processor System (HPS), handling game logic, state management, and preliminary interaction.

3. **FPGA Implementation**: Design and implement specialized hardware modules on the FPGA (using SystemVerilog) responsible for accelerated graphics rendering, physics simulation, and collision detection.

4. **Hardware Verification**: Establish a Verilator-based simulation testbench to verify and validate FPGA modules individually, ensuring correctness in rendering algorithms, physics accuracy, and collision handling.

5. **User Interface and Input**: Develop a responsive user interface that supports interaction through a USB keyboard, ensuring smooth and intuitive player control.

6. **System Integration and Testing**: Integrate the FPGA hardware modules with the HPS software components, optimizing communication and data flow between hardware and software to achieve a stable game experience at 60 frames per second.

6 Conclusion

This project will demonstrate the capability of the Intel DE1-SoC platform in real-time game rendering and physics simulation. By developing a simplified game titled "Forest Fire and Ice," we will investigate the advantages, challenges, and potential performance improvements provided by combining FPGA hardware acceleration with software running on the embedded ARM-based HPS.