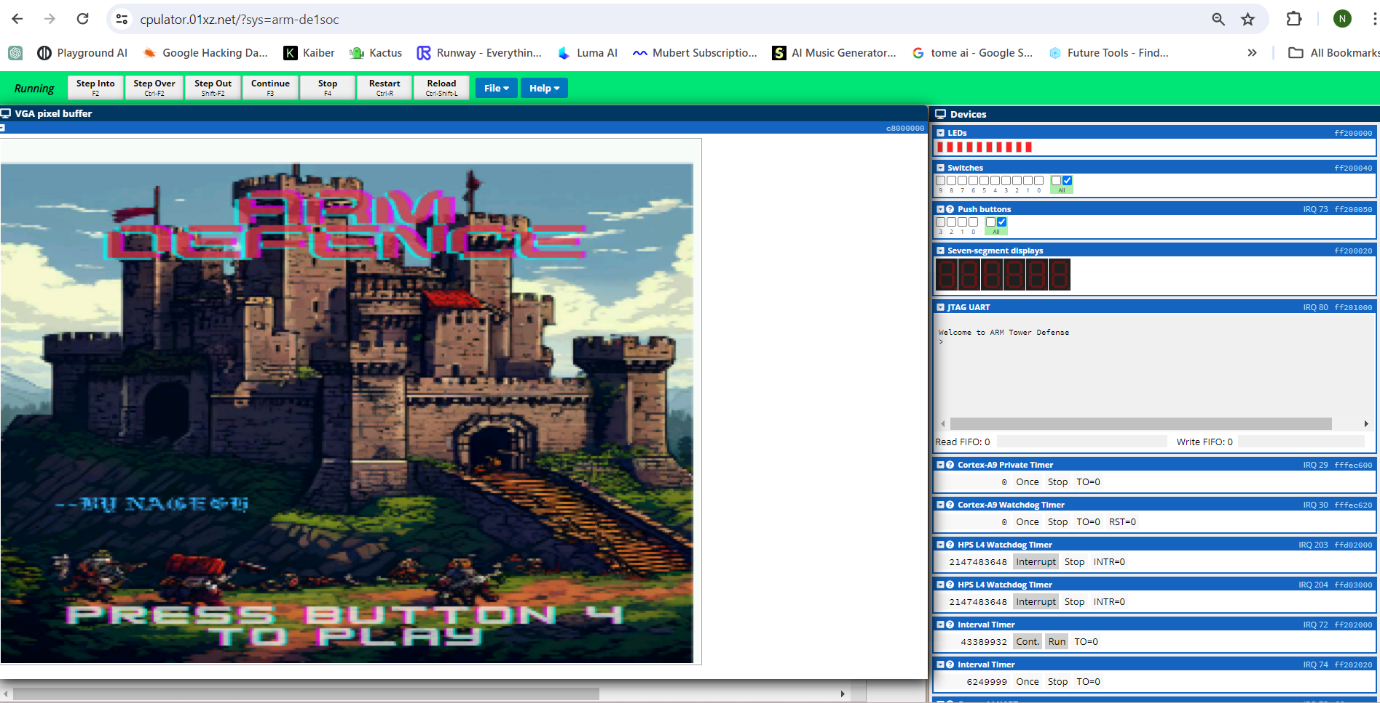
CSE 210 Mini\_Project\_II Report :-

Arm Defense



Introduction :

This report presents the development and implementation of a tower defense game for the DE1-SoC board in CS210. Drawing inspiration from classic titles like "Plants vs. Zombies" and 1-bit tower defense games, our project focuses on strategic turret placement to repel preprogrammed waves of enemies.

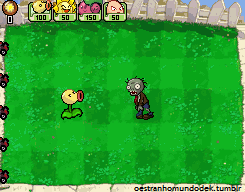
Game Overview

The game tasks players with utilizing points as resources to strategically position turrets, each with unique abilities, against incoming enemy waves. This blend of resource management and strategic planning forms the core gameplay experience.

Design Inspiration

Influenced by "Plants vs. Zombies" and 1-bit retro aesthetics, our game integrates these inspirations with original mechanics and visual design. The objective is to create an engaging and challenging tower defense experience for the DE1-SoC platform.

A screenshot of a video game

Description automatically generated 

Report Objectives

This report details the development process, challenges faced, design decisions made, and technical aspects of our tower defense game. It aims to provide a comprehensive overview of the project's creation, serving as a documentation of methodologies used in game development for the DE1-SoC board.

Graphics Development

Double Buffering for Smooth Rendering

To achieve smooth rendering, we implemented double buffering for the VGA display. This involved utilizing both the front and back buffers, with the front buffer displaying the current frame while the back buffer was prepared for the next frame. Once rendering for the next frame was complete, the buffers were swapped. This technique ensured seamless transitions between frames, enhancing the visual experience of the game.

Frame Storage for Transition Effects

In addition to double buffering, we stored the last two frames of the game. This storage method enabled smooth transitions between frames, particularly noticeable during animations or scene changes. By storing the previous frames, we ensured that new frames were written to the back buffer without abrupt changes, contributing to a more polished gameplay experience.

VSync for Synchronization

To synchronize the game's framerate with the monitor's refresh rate, we implemented VSync (Vertical Synchronization). VSync aligns the framerate of our game with the monitor's refresh rate, preventing tearing and ensuring smoother gameplay visuals. This synchronization technique helps avoid visual artifacts and provides a more consistent and enjoyable display for players.

Image Conversion for User Interface

For rendering the start, game over, and game win pages, a simple Java code was employed. This code converted 32-bit images to 16-bit images, reducing memory usage and optimizing image processing. The RGB values of these 16-bit images were then stored in a short integer array. This approach maintained visual quality while efficiently handling the user interface elements, enhancing the overall aesthetic and responsiveness of these pages.

A screenshot of a computer

Description automatically generated

Controls

Start/Pause Game

Push Button 3: This button serves a dual purpose, allowing players to both start and pause the game. Upon pressing Push Button 3, the game begins, initializing the waves of enemies. Pressing it again during gameplay will pause the game, halting enemy movement and turret placement until resumed.

Turret Placement

Push Button 3, 2, 1: Players can strategically place specific turret types using Push Buttons 3, 2, and 1. Each button corresponds to a different turret type, enabling quick selection and placement during gameplay.

Push Button 3: Places a turret of Type 1.

Push Button 2: Places a turret of Type 2.

Push Button 1: Places a turret of Type 3.

Player Cursor Movement

JTAG-UART ARM0 Interface: The player cursor movement is controlled using the JTAG-UART ARM0 interface. This interface allows players to navigate the game grid using the WASD keys:

W: Move the cursor upwards.

A: Move the cursor to the left.

S: Move the cursor downwards.

D: Move the cursor to the right.

Game Mechanics and Game Loop

Grid-Based Gameplay

The game operates on a grid-based system, dividing the screen into tiles where various entities reside, including game tiles, enemies, and turrets. This grid also determines the paths that enemies will take through the game environment. The strategic placement of turrets along these paths is crucial for defending against incoming waves of enemies.

Turret Mechanics

Turret Shooting: Turrets will automatically shoot at enemies within their range.

Turret Range Calculation: The range of each turret is calculated using Bresenham's algorithm for circle printing. This algorithm determines the cells within the circular range of a turret, allowing for precise targeting of enemies.

Shot Delay: After each shot, there is a certain delay before the turret can fire again. This delay adds a strategic element, requiring players to consider timing and placement for optimal defense.

Enemy Health and Damage

Enemy Health Display: The health of enemies is represented visually as a line above them. This health line decreases as enemies sustain damage.

Damage Mechanism: When a turret successfully hits an enemy, the enemy's health decreases accordingly. This visual feedback allows players to assess the effectiveness of their turret placements and adjust strategies as needed.

Game Objective and Waves

Objective: Players aim to defend against waves of enemies and survive for as long as possible.

Health Points: The player starts with a total of 10 health points.

Wave System: The game features 4 waves of enemies, each increasing in difficulty and numbers. Players must strategically place turrets and manage resources to survive these waves.

A computer screen shot of a black and red screen

Description automatically generated

Output Medium

VGA Display

The game is rendered on the VGA display buffer, providing players with a visual representation of the game grid, turrets, enemies, and game elements.

The VGA display updates in real-time as the game progresses, showing animations, movements, and interactions between entities.

Grid-based cells, turrets, enemies, and pathing are all visually represented on the VGA screen, enhancing the player's understanding of the game environment.

LED Array for Health

Player health is displayed using an LED array.

The LED array consists of 10 LEDs, each representing one health point.

As the player's health decreases, LEDs turn off sequentially, providing a clear visual indication of remaining health.

When all LEDs are off, the player's health reaches 0, signaling the end of the game.

Seven-Segment LEDs for Points

Points accumulated by the player are displayed using seven-segment LEDs.

The seven-segment display provides a numerical representation of the player's current score.

As the player earns points by defeating enemies, the display updates in real-time to reflect the increasing score.

This visual feedback motivates players to strategize and maximize their points.

JTAG-UART Terminal Messages

Messages and game updates are printed to the JTAG-UART terminal.

Important game information, such as wave progression, turret placement, and game status, is communicated through text output.

Players can receive feedback and instructions through the terminal, enhancing their understanding of game events and objectives.

Debugging information and error messages are also displayed in the terminal, aiding in troubleshooting and development.