# Project Proposal: Oregon Trail Game

**Team Members:**

Xingcan Chen (xc2807)

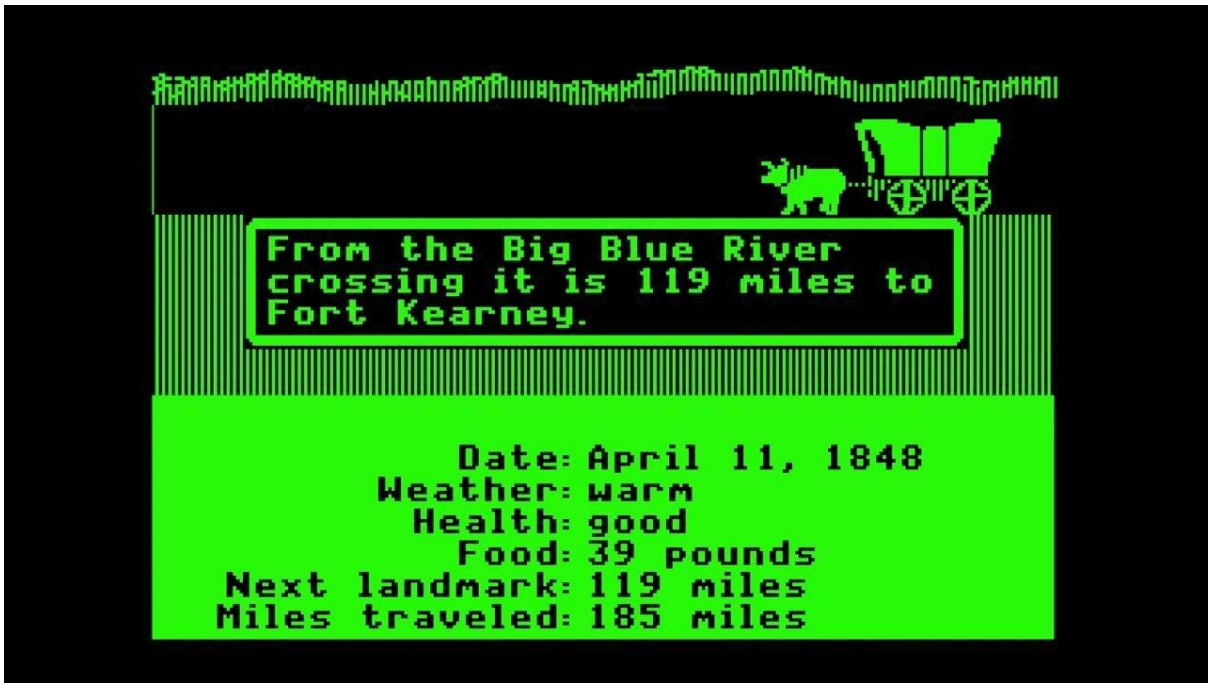
Adam Auer (aha2197)

Sunny Qi (sq2284)

## Overview:

In this project we will build a simple Oregon Trail-style game on the DE1-SoC board. The game will run partly in software and partly in hardware. A VGA monitor will show the game screen, and a USB keyboard will be used for player input.

The game simulates a trip across a map. During the trip the player will need to manage several resources such as food, health, money, and ammunition. Random events may occur while traveling, such as encountering a trader, or one of the party members getting sick. The player will choose different actions in response to these events, and the game state will change depending on the choices. The goal of the original Oregon Trail game is to survive long enough and reach the final destination by wagon. Our game will be similar in the sense that the player's objective is to survive and reach a destination, although the specific theme may not necessarily be that of the original Oregon Trail game.




## System Concept:

The system includes two parts: software running on the processor and hardware implemented on the FPGA.

The software will run on the processor under Linux. It will handle the main game logic, including

player input, event generation, and updates to the player's resources. The program will keep track of the game state and send display information to the FPGA.

The FPGA will be responsible for generating VGA video output. A video controller module will produce the VGA signals. Other hardware modules may draw text and simple graphics on the screen, such as background tiles or icons.

The processor and FPGA will communicate through memory-mapped registers. The processor will push data into shared registers and the FPGA side picks it up to drive the display.

Most of the software will be written in C and run in Linux. Existing framebuffer and USB keyboard libraries may be used to handle keyboard input and basic display setup during early testing. Hardware modules will be written in SystemVerilog.We are still deciding whether to use a framebuffer or direct register writes for the display interface.

## Expected Outcome

The result will be an interactive game running directly on the DE1-SoC board. The user will be able to play the game through a VGA monitor and keyboard without requiring a separate computer. The screen will display game information such as player status, resources, and event descriptions.

## Stretch Goals

After completing work on implementing the main game logic, graphics, and user interface, the next logical step would be to add audio to enhance the user experience. Whenever an event–such as a thunderstorm or animal attack–occurs, a sound should play to quickly suggest the severity of consequences to the player. If a user presses a button on a keyboard to take an action that is, at some particular moment, forbidden, a simple tone should indicate that the input has been received but rejected. Implementing these behaviors would require curating appropriate audio files and writing a hardware-based audio controller.

A further addition to make our edition of The Oregon Trail truly unique would be an online multiplayer mode. To make the experience of travelling across the American frontier more challenging and fun, two players could compete in a race, potentially sabotaging another along the way. For example, the “world” might have a limited supply of game to hunt, and one player’s hunting all the game might cause the other to starve. To implement this functionality, messages between two local instances of the game would be shared via TCP. Because only one FPGA is provided to each group, we would likely test our multiplayer mode using one “real” instance running on an FPGA and one “dummy” instance with a simple, text-based interface running in a terminal emulator on a Linux desktop.

## User Interface

- Input - Keyboard
- Video output - VGA
- Audio output (time-permitting) - 3.5mm analog jack

## Main Tasks

- Design the game (goal, mechanics, etc.)
- Implement core game logic
- Design the user interface
- Enable keyboard input

- Enable VGA display rendering