Interactive Adventure Game

Greg Luthman
Akash Shah
Overview

• Inspiration: Super Mario Brothers
• Goal: create a side scroll adventure game that puts the player into the game world.
  – A live action, side scroll adventure game
  – Instead of playing with a controller and seeing a character move on screen, everything is controlled by the player's actions in front of a camera.
  – Use the video of the player to determine the proper commands to send to the game
  – The player will be able to duck objects, jump over objects, move forward or backward in the game world.
Video Subsystem Breakdown

• Video in: Handles the video that is being input from the camera into the labkit.

• Green Screen Module
  – Handles detection of the green screen background
  – Allows for an overlay that replaces the background color with the virtual game world.
  – The module will allow for handling of variations in intensity and color
Character Module
- Utilizes the green screen module
- Determines the location of the character on the captured image.

Gesture Recognition
- Recognizes simple gestures that the player may choose to execute.
- A group of control signals denote which actions have been made.
Video Subsystem Cont’d

• Overlay
  – Overlays incoming filtered video feed with game environment

• VGA Output
  – Outputs the VGA signal to the monitor
Game Subsystem

• Frame Buffer
  – 240 x 256 resolution, 9-bit pixel data (fits onto BRAM)
  – Can only write when vcount is off screen to avoid glitchy looking graphics

• Background Generator
  – Updates the frame buffer using data from the level rom and the tile memory
  – Uses “left_pixel” from FSM to determine where the screen is in the game world
Game Subsystem cont’d

• Tile Memory
  – Holds 64 tiles
  – Each tile is 16 x 16

• Level ROM
  – Game world is 15 tiles high by 256 tiles long
  – Given a row and a column, will return which type of tile is in that spot
Game Subsystem cont’d

• Sprite RAM
  – Holds data for 16 different sprites
  – X-coordinate, y-coordinate, and data about the sprite (tile type, sprite state, etc.)

• FSM
  – Updates and draws the sprites into the Frame Buffer after the Background generator
  – Detects collisions between sprites, the player, and the background
  – Does all physics calculations
Timeline

• Last Week
  – Camera Working
  – Chroma Key
  – Frame Buffer 100% finished
  – Background Generator working

• This Week
  – Character Recognition
  – Background Generator 100% finished
  – Level ROM, Tile memory working
Timeline cont’d

• Next Week (before Thanksgiving)
  – Simple Gestures
  – Sprite RAM working

• First Week of December
  – Gestures finished
  – FSM 100% finished
  – Background tiles finished (.coe 50% finished)
Timeline cont’d

• 2\textsuperscript{nd} Week of December
  – Sprite tiles finished (.coe files 100 \% finished)
  – Testing integrated system

• Just in Case Weekend
  – Debugging
  – Extras (if time)
Questions?