Labyrinth

6.111 Final Project

Laplie Anderson and Mihalis Papalampros

Project Proposal

The goal of this project is to implement a two-person 3D maze game. In the game, the players attempt to reach the center of the map as fast possible. The first player to reach the center wins. Each of the two players controls their own character and has an individual view of their surroundings. The characters will be place at disperse locations at the start of every game.

The map itself will be randomly generated at the start of each game. To make sure each map is playable, there will be a set of invariants that each map obeys. Some example invariants is that there is always at least one path to the center, and there is a path from every space to every other space. To make the maps fair, the generator will assign a difficulty to each player and ensure that the difficulties for each player is similar.

In order to make the game playable, a mini-map will be displayed showing only icons representing the players and where they are in relation to the entire map. The mini-map will help the players reach the center of the map without telling them how the map actually looks.

The game is broken in to 3 major modules. The map generation module is in charge of creating the maps. This module may take a long time, but the length of time it takes to generate is not very important because map generation is done before the game (players aren't waiting for the map generation). The map that is output is a 1-dimentional array of bits where later logic will group the bits to create lines of the map. A 3x3 map will use a 9 bit array where every 3 bits corresponds to a line on the map. A wall will be a 1, and an open space will be a 0.

This module will be tested by using it to generate maps and ensuring that none of the generated maps break any invariants or constraints.

The next major module is the Game Logic module. This module keeps track of where players are in the map, and the angle they are looking at. This module takes inputs (such as moving up or turning left), ensures that the move is valid, and then updates values accordingly.

This module will be tested by giving the game logic module valid and invalid actions (walk into an open space, walk into a wall, etc.) and making sure the game prohibits the invalid actions and allows the valid actions.

The last major module is the Video Renderer. This module takes as input the player positions and the angle they are looking at along with the map uses them to render the view of the world for each player. The module then outputs the appropriate hsync, vsync, blank, hcount, vcount, to the labkit to be displayed on the screen. The major difficulty here is efficiently rendering of the world (or prerendering and displaying in a non-choppy manner).

This module will be tested by giving it valid inputs and have it generate a map of the world. The inputs will be changed and the map of the world should be changed accordingly.

Laplie Anderson will be doing the Video render modules and any sub modules that are required. Mihalis Papalampros will be implementing the map generator module and the game logic modules.

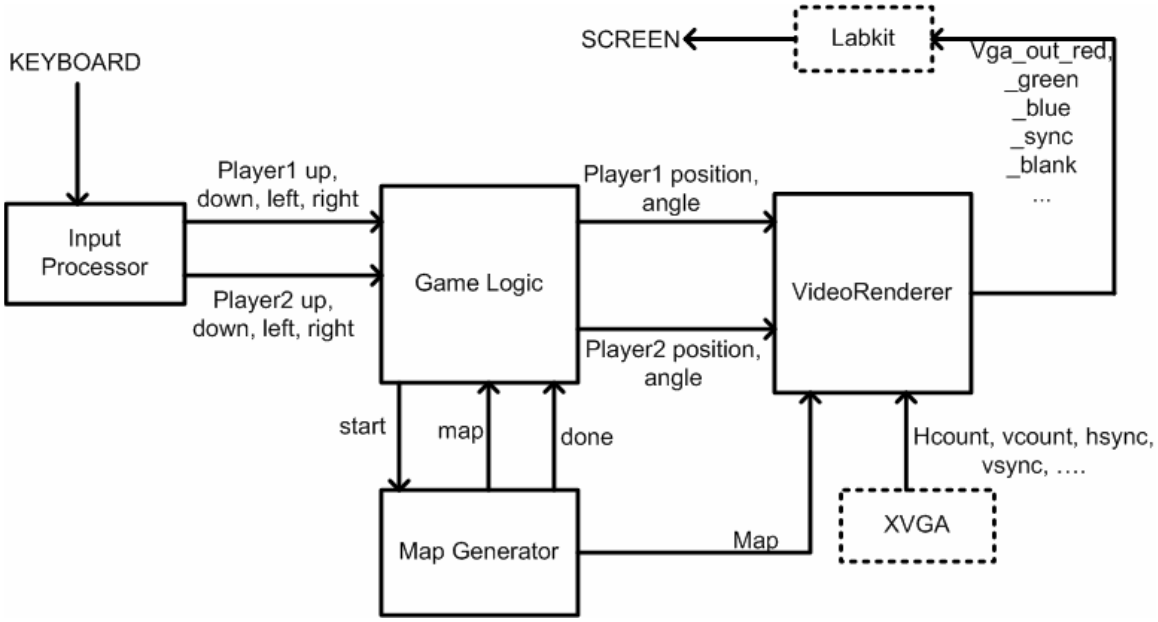

Figure 1: Block Diagram showing major modules of Labyrinth

Input

Description: Takes in keyboard input and debounces it. Then translates those inputs into player commands.

Inputs: Keyboard
Outputs: Player1_up, player1_down, player1_left, player1_right, Player2_up, player2_down, player2_left, player2_right

Game Logic

Description: Takes player commands and updates the game state accordingly.
Inputs: Player1_up, player1_down, player1_left, player1_right, Player2_up, player2_down, player2_left, player2_right, map

Outputs: player1_vposition, player1_hposition, player1_angle, player2_vposition, player2_hposition, player2_angle

Map Generator

Inputs: start
Outputs: done, map

Video Renderer

Description: Take in player attributes and map and display a view to each player
Inputs: Player1_up, player1_down, player1_left, player1_right, Player2_up, player2_down, player2_left, player2_right, map
Outputs: vga_out_red, vga_out_green, vga_out_blue, vga_out_hsync, vga_out_vsync, vga_out_blank, vga_out_hcount, vga_out_vcount

XVGA (same one used in Lab5)
Description: Generate XVGA signals

Labkit (probably default labkit.v)
Description: Interface with Labkit and screen