Computing Machinery II
Assignment 4

Maze Game: SNES Controller and Frame Buffer Programming

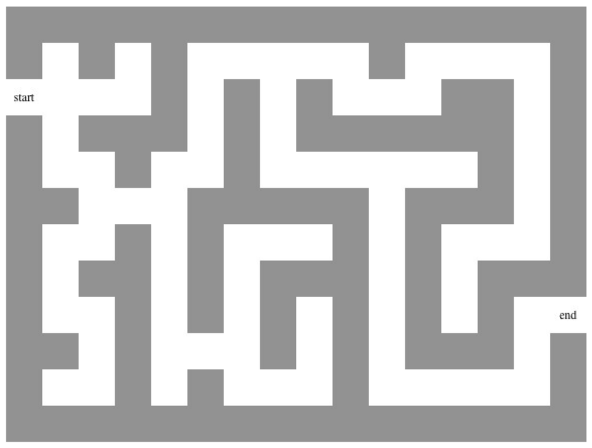
Create a program for the Raspberry Pi that presents a maze on the HDMI monitor. Use a dark color for the walls of the maze, a lighter color for the pathways through the maze, and red for the player that moves through the maze on the pathways. When the player reaches the exit of the maze, redisplay the player in green to indicate that the maze has been solved and the game is over.

The maze should be constructed on a grid with 12 rows and 16 columns, where each element of the grid is a square that is 64 x 64 pixels in size. Use a display resolution of 1024 x 768 pixels, where each pixel is a 32-bit integer containing the RGB encoding for the color. Use these squares to construct the walls and pathways of the maze, and to represent the player as it moves through the maze.

You should use a two-dimensional array as the data structure to represent the maze. For example, in the following, 0 represents a pathway, 1 a wall, 2 the entrance, and 3 the exit:

```
int maze[12][16] = {
    {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
    {1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1},
    {2, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1},
    {1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1},
    {1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1},
    {1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1},
    {1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1},
    {1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1},
    {1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 3},
    {1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1},
    {1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1},
    {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}};
```

This could be displayed as follows:



Your program will use the SNES controller connected to the Raspberry Pi to control the movement of the player through the maze. When your program starts up, it will display the maze, but not the position of the player. When the user presses the controller start button, show the player in red in the starting position. Use the joypad arrow buttons to move the player one position to the left, right, up, or down. The player can only move on the pathways, and is not allowed to go through any walls or leave the playing area. Display the player in red as it moves through the maze, and be sure to recolor the pathway its original color once the player has moved onto a new position. When the player reaches the exit of the maze, display the player using a green color to indicate that the game is over. At this point, the player can press the start button to restart the game.

You may write your program in C or in A64 assembly language, or a combination of both. You can freely use the following files located on D2L as part of your code base: gpio.h, link.ld, mailbox.s, mailbox.h, Makefile, start.s or startV2.s, systimer.c, systimer.h, uart.c, and uart.h.

Demonstration

You will demo your program to your TA shortly after the due date. Your TA will set up a schedule for this in class before the due date.

Bonus 1 (5% if fully implemented)

Use a pale red color to display the pathway that the player has followed to go from the entrance to the exit of the maze. Only show the direct pathway. In other words, do not show any side trips the player has taken. This means that you will have to recolor a pathway if a player backs up after encountering a dead end.

Bonus 2: (10% if fully implemented)

Allow the player to use "dissolving acid" on a selected wall square to create a new pathway through the maze. If the player presses both the X button and one of the arrow buttons simultaneously, then the wall square in the direction of the arrow will be dissolved. It will take 5 applications of the acid to fully dissolve the wall. Show this by displaying successively lighter wall colors on each application. Once fully dissolved, the square is then considered a regular pathway, and the player can move through this square as usual.

New Skills Needed for this Assignment:

- Ability to read data from the SNES controller
- Ability to do frame buffer programming

Submit the following on D2L:

1. All files needed to compile your program. This will include the files listed above, plus any custom code that you have created. Your TA will compile and run your program to make sure it is working. If you are doing the bonus parts of the assignment, then also include a README file that indicates this.

Computing Machinery II

Assignment 4 Grading

Student:____

Responds to 5 SNES button presses	5	______	
Maze displayed correctly (walls, pathways)	4	______	
Correct player movement (4 directions, stays inbounds, stays on pathway)	6	______	
Player movement properly displayed	2	______	
End of game recognized/displayed	2	______	
Game restart works properly	2	______	
Code formatting and documentation	2	______	
Design quality	2	______	
Total	**25**	______	_____%
Bonus 1 (5% if fully implemented)			_____%
Bonus 2 (10% if fully implemented)			_____%
Assignment Grade			_____%