**Bitmap Project Report**

By: Kevin Boudreaux

11/4/2021

**Overview**

For my Bitmap Project, I made a maze where that the user must traverse and get to the other end. I marked the walls as black pixels, the user as a red pixel, and the destination as a green pixel. There is more than one correct path to the destination since I am unexperienced in creating mazes. The program makes the maze by printing out white spots where a wall is not located. I did this by manually counting where each wall section is in a X, Y coordinate system with 0,0 being the top left of the map.

I then created a simple JAVA program that goes through the coordinates I put in a text document and converted those coordinates into a location based off one coordinate also known as a pointer. The program outputs this new list of coordinates into the console which I copied over into the MIPS program (this does not need to be repeated by the user, but the program and files are left in, if someone wishes to reenact what I did).

The MIPS program then goes into a loop going from 0 to 4069 (the size of the bitmap), and for each location that is not in the list of walls is turned into a white pixel to represent the path a player can go down. Once it has finished printing the layout, it prints a green pixel at the destination and a red pixel that represents the player. The program then loops and waits for a player’s input and based on the input will go into one of five parts of the program. All are similar except for one which ends the program; they check for each possible direction the user could go and checks to see if it is a valid input, if not it will not move the player but if it is it will update the bitmap and move the player. Once the player reaches the green pixel a message will appear congratulating the player on beating the maze and then end the program.

**Flowchart/Pseudocode**

1. Loop 0 to 4096
   1. Check to see if current location is at 4096, if so, jump to 2.
   2. Check to see if current location is a wall
      1. If it is skip too next
      2. If it is not, then print a white pixel at current location and decrease array pointer by 4. (To keep array pointing at correct location)
   3. Increment bitmap location by 1
   4. Increment array pointer by 4 (for 4 bits)
2. Print exit / objective as green pixel
3. Print player as red pixel
4. Wait for player input
   1. If input is a space, go to 5
   2. If input is a “w”, go to 6
   3. If input is a “d”, go to 7
   4. If input is a “s”, go to 8
   5. If input is a “a”, go to 9
   6. If no input, go to 4
5. End program
6. Get location North of current
   1. Check to see if it is less than zero for out of bounds
   2. Check to see if the pixel at that location is black
      1. If it is, go to 4
      2. If it is not, erase current pixel and make pixel above red, then go to 4
7. Get location East of current
   1. Check to see if the pixel at that location is black
      1. If it is, go to 4
      2. If it is not, erase current pixel and make pixel above red, then go to 4
8. Get location South of current
   1. Check to see if the pixel at that location is green
      1. If it is print out winner message and go to 5
   2. Check to see if the pixel at that location is black
      1. If it is, go to 4
      2. If it is not, erase current pixel and make pixel above red, then go to 4
9. Get location West
   1. Check to see if the pixel at that location is black
      1. If it is, go to 4
      2. If it is not, erase current pixel and make pixel above red, then go to 4

**Sample Runs**

Diagram

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application, Word

Description automatically generated