/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Orion Serup

CS M10A

Dr. Alnaji

Moorpark College

Spring 2019

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <TFT\_ILI9341.h> // Display driver library

#include <SPI.h> // Communication protocol library

#include <EEPROM.h> // ROM storage library

#include <avr/pgmspace.h> // PROGMEM use library

#define up 2

#define down 3 // directional button pins

#define left 4

#define right 5

#define WIDTH 320

#define HEIGHT 240

#define MAZEHEIGHT 24

#define MAZEWIDTH 32

#define RATIO 10

#define BLACK 0x0000 //all of the colors in 565 form for ease of access

#define BLUE 0x001F

#define RED 0xF800

#define GREEN 0x07E0

#define CYAN 0x07FF

#define MAGENTA 0xF81F

#define YELLOW 0xFFE0

#define WHITE 0xFFFF

#define ORANGE 0xFC00

#define PINK 0xFC18

#define LIGHTBLUE 0x87FF

TFT\_ILI9341 disp = TFT\_ILI9341(HEIGHT, WIDTH); //create a display instance with dimensions

bool wall(uint16\_t x, uint16\_t y);

void drawmenu();

void about();

void drawmap();

void drawplayer(uint16\_t x, uint16\_t y, uint16\_t color);

void endgame(int score, int moves);

void initialize();

void game();

uint16\_t px = 10, py = 239, playerx = 10, playery = 239;

bool gameover = false;

void setup() { // runs once

initialize();

game();

}

void loop() {} // while(1); // we dont need this since it is all done in setup

void drawmap() { // scales and prints the maze

const byte PROGMEM maze1[MAZEHEIGHT][MAZEWIDTH] = // maze from paint

{

{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1},

{1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1},

{1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1},

{1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1},

{1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1},

{1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1},

{1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},

{1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1},

{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1},

{1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1},

{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1},

{1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1},

{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},

{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1},

{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1},

{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}

};

disp.fillScreen(BLACK);

Serial.println("MAP PRINTING");

uint8\_t counterx, countery;

uint16\_t realx, realy;

for (countery = 0; countery < MAZEHEIGHT; countery++) {

for (counterx = 0; counterx < MAZEWIDTH; counterx++) {

realx = RATIO \* counterx;

realy = RATIO \* countery;

if ((pgm\_read\_byte(&(maze1[countery][counterx]))) == 1) // create

disp.fillRect(realx, realy, 10, 10, WHITE);

else if ((pgm\_read\_byte(&(maze1[countery][counterx]))) == 2)

disp.fillRect(realx, realy, 10, 10, GREEN);

else if ((pgm\_read\_byte(&(maze1[countery][counterx]))) == 3)

disp.fillRect(realx, realy, 10, 10, RED);

}

}

Serial.println("MAP PRINTED");

}

bool wall(uint16\_t x, uint16\_t y) { //if there is a block in the way return true

int acx, acy;

acx = x / 10;

acy = y / 10;

if (pgm\_read\_byte(&(maze1[acy][acx])) == 1)

return true;

else

return false;

}

void drawplayer(uint16\_t x, uint16\_t y, uint16\_t color) { //draws the player at a point with (x,y) and color color

const bool player[RATIO][RATIO] =

{

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 1, 1, 1, 1, 1, 1, 1, 1, 0},

{0, 1, 0, 1, 1, 1, 1, 1, 1, 0},

{0, 1, 1, 1, 1, 1, 1, 1, 1, 0},

{0, 1, 1, 1, 0, 1, 1, 1, 1, 0},

{0, 1, 1, 1, 1, 1, 1, 1, 1, 1},

{0, 1, 1, 1, 1, 1, 0, 1, 1, 0},

{0, 1, 1, 1, 1, 1, 1, 1, 1, 1},

{0, 1, 1, 1, 1, 1, 1, 1, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

}; // character looks like a square

for (int i = 0; i < RATIO; i++) {

for (int j = 0; j < RATIO; j++) {

if (player[i][j] == 1)

disp.drawPixel(x + i, y - j, color);

}

}

px = playerx; // save pos as old

py = playery;

playerx = x; // update position

playery = y;

delay(5); // wait

// update screen

}

void drawmenu() { // menu

const byte PROGMEM menu[5][55] = {

{ 1, 0, 0, 0, 1, 0, 0, 0, 2, 2, 0, 0, 3, 3, 3, 3, 0, 4, 4, 4, 4, 0, 0, 0, 5, 5, 5, 0, 0, 6, 0, 0, 6, 0, 7, 0, 0, 0, 7, 0, 8, 0, 0, 0, 8, 0, 9, 9, 9, 9, 0, 10, 10, 10, 0},

{ 1, 1, 0, 1, 1, 0, 0, 2, 0, 2, 0, 0, 0, 0, 3, 0, 0, 4, 0, 0, 0, 0, 0, 0, 5, 0, 0, 5, 0, 6, 0, 0, 6, 0, 7, 7, 0, 0, 7, 0, 8, 8, 0, 0, 8, 0, 0, 0, 0, 9, 0, 10, 0, 0, 10},

{ 1, 0, 1, 0, 1, 0, 2, 0, 0, 2, 0, 0, 0, 3, 0, 0, 0, 4, 4, 4, 4, 0, 0, 0, 5, 0, 0, 5, 0, 6, 0, 0, 6, 0, 7, 0, 7, 0, 7, 0, 8, 0, 8, 0, 8, 0, 0, 9, 9, 9, 0, 10, 0, 0, 10},

{ 1, 0, 0, 0, 1, 0, 2, 2, 2, 2, 2, 0, 3, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 5, 5, 5, 0, 0, 6, 0, 0, 6, 0, 7, 0, 0, 7, 7, 0, 8, 0, 0, 8, 8, 0, 0, 0, 0, 9, 0, 10, 10, 10, 0},

{ 1, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 3, 3, 3, 3, 0, 4, 4, 4, 4, 0, 0, 0, 5, 0, 0, 5, 0, 0, 6, 6, 0, 0, 7, 0, 0, 0, 7, 0, 8, 0, 0, 0, 8, 9, 9, 9, 9, 0, 0, 10, 0, 0, 10}

};

int color;

byte val;

for (int i = 0; i < 5; i++) { // run through all of the positions in the menu array

for (int j = 0; j < 55; j++) {

val = pgm\_read\_byte(&(menu[i][j])); // read the value at the location

switch (val) { // correlate a byte to a 16 bit color

case 0:

color = BLACK;

break;

case 1:

color = RED;

break;

case 2:

color = ORANGE;

break;

case 3:

color = YELLOW;

break;

case 4:

color = GREEN;

break;

case 5:

color = CYAN;

break;

case 6:

color = LIGHTBLUE;

break;

case 7:

color = BLUE;

break;

case 8:

color = MAGENTA;

break;

case 9:

color = PINK;

break;

default:

color = WHITE;

}

disp.drawRect(104 + 2 \* j, 154 + 2 \* i, 2, 2, color); // draw a 2 unit wide square for every array location cented in the screen

delay(5); // a delay so that it loads in for effect;

}

}

disp.setTextSize(2);

disp.setTextColor(WHITE);

delay(200);

disp.setCursor(150, 200);

disp.println("Press Up for Game"); // prompt game

disp.setCursor(150, 220);

delay(200);

disp.println("Press Down for About"); // go to about

Serial.println("MENU IS PRINTED");

delay(500); // wait a half second

while (1) { // loop

if (!digitalRead(down)) // pushing down?

about(); // if so , go to about screen

if (!digitalRead(up)) // pushing up?

break; // leave while

delay(50);

}

}

void about() { // disps general information about me and the game

disp.fillScreen(BLACK);

disp.setCursor(10, 30);

disp.setTextColor(WHITE);

disp.setTextSize(1);

disp.println("Made By Orion Serup CS M10A Moorpark College Spring 2019 \n");

disp.println("HOW TO:");

disp.println("Use Buttons to Move");

disp.println("You Have 1 Minute");

disp.println("Go Fast \n");

disp.println("Press UP to return \n");

while (1) { // loop forever

if (!digitalRead(up)) { // pushing up?

drawmenu(); // disp menu

break; // leave loop

}

}

}

void endgame(int score, int moves) {

byte val;

disp.fillScreen(WHITE);

disp.setCursor(1, 10);

disp.print("Your Score: "); // display score and number of moves

disp.print(score);

disp.print(" IN ");

disp.print(moves);

disp.println(" Moves \n");

disp.print("HIGH SCORE: ");

val = EEPROM.read(0); // retrieve highscore value from rom

if (score > val) {

EEPROM.write(0, score);

disp.println(score);

disp.println("NEW HIGHSCORE");

}

else

disp.println(val);

}

void initialize() {

pinMode(up, INPUT\_PULLUP); // the buttons are for reading

pinMode(down, INPUT\_PULLUP);

pinMode(left, INPUT\_PULLUP);

pinMode(right, INPUT\_PULLUP);

Serial.begin(9600); // start serial communication

Serial.println("Serial is to simply give feedback to the computer"); // begin Serial as debugger

disp.init(); // start the screen and fill it black

disp.setRotation(3);

}

void game() {

while (1) {

int finalscore = 100, counter = 0;

bool gameover = false;

drawmenu();

disp.fillScreen(BLACK);

drawmap(); // draw map function

delay(50); // wait

drawplayer(10, 239, RED);

int cx, cy;

while (!gameover) { // while the player is still playing

cx = playerx;

cy = playery;

if (!digitalRead(up)) {

if (!wall(playerx, playery - 10)) { // check adjacent square

py = playery; // save pos as old

playery -= 10;

if (playery > 239 || playery < 0) // bounds check

cy = playery;

Serial.println("PLAYER UP"); // Serial Debug Message

}

}

if (!digitalRead(down)) {

if (!wall(playerx, playery + 1)) {

py = playery;

playery += 10;

if (playerx > 239 || playery < 0)

cy = playery;

Serial.println("PLAYER DOWN");

}

}

if (!digitalRead(left)) {

if (!wall(playerx - 1, playery)) {

px = playerx;

playerx -= 10;

if ((playerx > 319 ) || (playerx < 0))

cx = playerx;

Serial.println("PLAYER LEFT");

}

}

if (!digitalRead(right)) {

if (!wall(playerx + 10, playery)) {

px = playerx;

playerx += 10;

if ((playerx > 319 ) || (playerx < 0))

cx = playerx;

Serial.println("PLAYER RIGHT");

}

}

if ((cx != playerx) || (cy != playery)) { // if a button was pushed

disp.fillRect(px, py - 9, 10, 10, BLACK); // place a black square where the character was

drawplayer(playerx, playery, RED); // move the character to the new location

counter++;

if (counter == 1) //

disp.fillRect(0, 229, 10, 10, WHITE); // fix empty left box

}

finalscore--; // take away from the score

delay(200); // game speed, the lower the delay the faster the game

if ( ((playerx / 10 == 30) && (playery / 10 == 0)) || (finalscore == 0))

gameover = true;

}

endgame(finalscore, counter);

disp.println("PRESS UP TO RESTART");

while (1) {

if (!digitalRead(up))

break;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Orion Serup

CS M10A

Dr. Alnaji

Moorpark College

Spring 2019

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <TFT\_ILI9341.h> // Display driver library

#include <SPI.h> // Communication protocol library

#include <EEPROM.h> // ROM storage library

#include <avr/pgmspace.h> // PROGMEM use library

#define up 2

#define down 3 // directional button pins

#define left 4

#define right 5

#define WIDTH 320

#define HEIGHT 240

#define MAZEHEIGHT 24

#define MAZEWIDTH 32

#define RATIO 10

#define BLACK 0x0000 //all of the colors in 565 form for ease of access

#define BLUE 0x001F

#define RED 0xF800

#define GREEN 0x07E0

#define CYAN 0x07FF

#define MAGENTA 0xF81F

#define YELLOW 0xFFE0

#define WHITE 0xFFFF

#define ORANGE 0xFC00

#define PINK 0xFC18

#define LIGHTBLUE 0x87FF

TFT\_ILI9341 disp = TFT\_ILI9341(HEIGHT, WIDTH); //create a display instance with dimensions

bool wall(uint16\_t x, uint16\_t y);

void drawmenu();

void about();

void drawmap();

void drawplayer(uint16\_t x, uint16\_t y, uint16\_t color);

void endgame(int score, int moves);

void initialize();

void game();

uint16\_t px = 10, py = 239, playerx = 10, playery = 239;

bool gameover = false;

void setup() { // runs once

initialize();

game();

}

void loop() {} // while(1); // we dont need this since it is all done in setup

void drawmap() { // scales and prints the maze

const byte PROGMEM maze1[MAZEHEIGHT][MAZEWIDTH] = // maze from paint

{

{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1},

{1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1},

{1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1},

{1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1},

{1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1},

{1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1},

{1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},

{1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1},

{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1},

{1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1},

{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1},

{1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1},

{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},

{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1},

{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1},

{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}

};

disp.fillScreen(BLACK);

Serial.println("MAP PRINTING");

uint8\_t counterx, countery;

uint16\_t realx, realy;

for (countery = 0; countery < MAZEHEIGHT; countery++) {

for (counterx = 0; counterx < MAZEWIDTH; counterx++) {

realx = RATIO \* counterx;

realy = RATIO \* countery;

if ((pgm\_read\_byte(&(maze1[countery][counterx]))) == 1) // create

disp.fillRect(realx, realy, 10, 10, WHITE);

else if ((pgm\_read\_byte(&(maze1[countery][counterx]))) == 2)

disp.fillRect(realx, realy, 10, 10, GREEN);

else if ((pgm\_read\_byte(&(maze1[countery][counterx]))) == 3)

disp.fillRect(realx, realy, 10, 10, RED);

}

}

Serial.println("MAP PRINTED");

}

bool wall(uint16\_t x, uint16\_t y) { //if there is a block in the way return true

int acx, acy;

acx = x / 10;

acy = y / 10;

if (pgm\_read\_byte(&(maze1[acy][acx])) == 1)

return true;

else

return false;

}

void drawplayer(uint16\_t x, uint16\_t y, uint16\_t color) { //draws the player at a point with (x,y) and color color

const bool player[RATIO][RATIO] =

{

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{0, 1, 1, 1, 1, 1, 1, 1, 1, 0},

{0, 1, 0, 1, 1, 1, 1, 1, 1, 0},

{0, 1, 1, 1, 1, 1, 1, 1, 1, 0},

{0, 1, 1, 1, 0, 1, 1, 1, 1, 0},

{0, 1, 1, 1, 1, 1, 1, 1, 1, 1},

{0, 1, 1, 1, 1, 1, 0, 1, 1, 0},

{0, 1, 1, 1, 1, 1, 1, 1, 1, 1},

{0, 1, 1, 1, 1, 1, 1, 1, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

}; // character looks like a square

for (int i = 0; i < RATIO; i++) {

for (int j = 0; j < RATIO; j++) {

if (player[i][j] == 1)

disp.drawPixel(x + i, y - j, color);

}

}

px = playerx; // save pos as old

py = playery;

playerx = x; // update position

playery = y;

delay(5); // wait

// update screen

}

void drawmenu() { // menu

const byte PROGMEM menu[5][55] = {

{ 1, 0, 0, 0, 1, 0, 0, 0, 2, 2, 0, 0, 3, 3, 3, 3, 0, 4, 4, 4, 4, 0, 0, 0, 5, 5, 5, 0, 0, 6, 0, 0, 6, 0, 7, 0, 0, 0, 7, 0, 8, 0, 0, 0, 8, 0, 9, 9, 9, 9, 0, 10, 10, 10, 0},

{ 1, 1, 0, 1, 1, 0, 0, 2, 0, 2, 0, 0, 0, 0, 3, 0, 0, 4, 0, 0, 0, 0, 0, 0, 5, 0, 0, 5, 0, 6, 0, 0, 6, 0, 7, 7, 0, 0, 7, 0, 8, 8, 0, 0, 8, 0, 0, 0, 0, 9, 0, 10, 0, 0, 10},

{ 1, 0, 1, 0, 1, 0, 2, 0, 0, 2, 0, 0, 0, 3, 0, 0, 0, 4, 4, 4, 4, 0, 0, 0, 5, 0, 0, 5, 0, 6, 0, 0, 6, 0, 7, 0, 7, 0, 7, 0, 8, 0, 8, 0, 8, 0, 0, 9, 9, 9, 0, 10, 0, 0, 10},

{ 1, 0, 0, 0, 1, 0, 2, 2, 2, 2, 2, 0, 3, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 5, 5, 5, 0, 0, 6, 0, 0, 6, 0, 7, 0, 0, 7, 7, 0, 8, 0, 0, 8, 8, 0, 0, 0, 0, 9, 0, 10, 10, 10, 0},

{ 1, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 3, 3, 3, 3, 0, 4, 4, 4, 4, 0, 0, 0, 5, 0, 0, 5, 0, 0, 6, 6, 0, 0, 7, 0, 0, 0, 7, 0, 8, 0, 0, 0, 8, 9, 9, 9, 9, 0, 0, 10, 0, 0, 10}

};

int color;

byte val;

for (int i = 0; i < 5; i++) { // run through all of the positions in the menu array

for (int j = 0; j < 55; j++) {

val = pgm\_read\_byte(&(menu[i][j])); // read the value at the location

switch (val) { // correlate a byte to a 16 bit color

case 0:

color = BLACK;

break;

case 1:

color = RED;

break;

case 2:

color = ORANGE;

break;

case 3:

color = YELLOW;

break;

case 4:

color = GREEN;

break;

case 5:

color = CYAN;

break;

case 6:

color = LIGHTBLUE;

break;

case 7:

color = BLUE;

break;

case 8:

color = MAGENTA;

break;

case 9:

color = PINK;

break;

default:

color = WHITE;

}

disp.drawRect(104 + 2 \* j, 154 + 2 \* i, 2, 2, color); // draw a 2 unit wide square for every array location cented in the screen

delay(5); // a delay so that it loads in for effect;

}

}

disp.setTextSize(2);

disp.setTextColor(WHITE);

delay(200);

disp.setCursor(150, 200);

disp.println("Press Up for Game"); // prompt game

disp.setCursor(150, 220);

delay(200);

disp.println("Press Down for About"); // go to about

Serial.println("MENU IS PRINTED");

delay(500); // wait a half second

while (1) { // loop

if (!digitalRead(down)) // pushing down?

about(); // if so , go to about screen

if (!digitalRead(up)) // pushing up?

break; // leave while

delay(50);

}

}

void about() { // disps general information about me and the game

disp.fillScreen(BLACK);

disp.setCursor(10, 30);

disp.setTextColor(WHITE);

disp.setTextSize(1);

disp.println("Made By Orion Serup CS M10A Moorpark College Spring 2019 \n");

disp.println("HOW TO:");

disp.println("Use Buttons to Move");

disp.println("You Have 1 Minute");

disp.println("Go Fast \n");

disp.println("Press UP to return \n");

while (1) { // loop forever

if (!digitalRead(up)) { // pushing up?

drawmenu(); // disp menu

break; // leave loop

}

}

}

void endgame(int score, int moves) {

byte val;

disp.fillScreen(WHITE);

disp.setCursor(1, 10);

disp.print("Your Score: "); // display score and number of moves

disp.print(score);

disp.print(" IN ");

disp.print(moves);

disp.println(" Moves \n");

disp.print("HIGH SCORE: ");

val = EEPROM.read(0); // retrieve highscore value from rom

if (score > val) {

EEPROM.write(0, score);

disp.println(score);

disp.println("NEW HIGHSCORE");

}

else

disp.println(val);

}

void initialize() {

pinMode(up, INPUT\_PULLUP); // the buttons are for reading

pinMode(down, INPUT\_PULLUP);

pinMode(left, INPUT\_PULLUP);

pinMode(right, INPUT\_PULLUP);

Serial.begin(9600); // start serial communication

Serial.println("Serial is to simply give feedback to the computer"); // begin Serial as debugger

disp.init(); // start the screen and fill it black

disp.setRotation(3);

}

void game() {

while (1) {

int finalscore = 100, counter = 0;

bool gameover = false;

drawmenu();

disp.fillScreen(BLACK);

drawmap(); // draw map function

delay(50); // wait

drawplayer(10, 239, RED);

int cx, cy;

while (!gameover) { // while the player is still playing

cx = playerx;

cy = playery;

if (!digitalRead(up)) {

if (!wall(playerx, playery - 10)) { // check adjacent square

py = playery; // save pos as old

playery -= 10;

if (playery > 239 || playery < 0) // bounds check

cy = playery;

Serial.println("PLAYER UP"); // Serial Debug Message

}

}

if (!digitalRead(down)) {

if (!wall(playerx, playery + 1)) {

py = playery;

playery += 10;

if (playerx > 239 || playery < 0)

cy = playery;

Serial.println("PLAYER DOWN");

}

}

if (!digitalRead(left)) {

if (!wall(playerx - 1, playery)) {

px = playerx;

playerx -= 10;

if ((playerx > 319 ) || (playerx < 0))

cx = playerx;

Serial.println("PLAYER LEFT");

}

}

if (!digitalRead(right)) {

if (!wall(playerx + 10, playery)) {

px = playerx;

playerx += 10;

if ((playerx > 319 ) || (playerx < 0))

cx = playerx;

Serial.println("PLAYER RIGHT");

}

}

if ((cx != playerx) || (cy != playery)) { // if a button was pushed

disp.fillRect(px, py - 9, 10, 10, BLACK); // place a black square where the character was

drawplayer(playerx, playery, RED); // move the character to the new location

counter++;

if (counter == 1) //

disp.fillRect(0, 229, 10, 10, WHITE); // fix empty left box

}

finalscore--; // take away from the score

delay(200); // game speed, the lower the delay the faster the game

if ( ((playerx / 10 == 30) && (playery / 10 == 0)) || (finalscore == 0))

gameover = true;

}

endgame(finalscore, counter);

disp.println("PRESS UP TO RESTART");

while (1) {

if (!digitalRead(up))

break;

}

}

}