Console Logbook

Tuesday, January 17, 2023 2:47 PM

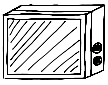
RETRO GAME CONSOLE LOGBOOK

Design Considerations:
- Must use parts provided in the kit
- Does the controller need to detach?
- Controls must work with game
- Digital input reads for joystick
- 3D-printed parts must be under 15 hours in printing time
- TFT display

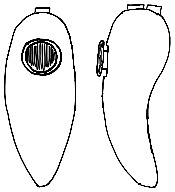
CHOICE OF GAME: PONG

Required Functions and Code Blocks:
- Allow ball to move at various angles through y=mx+b
- Ball must bounce off top and bottom planes of screen
- Movement of paddle
- Score system
- Computer controlled player 2

DESIGN:

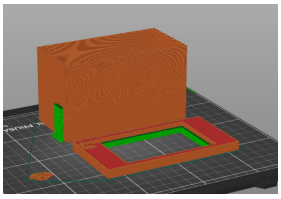


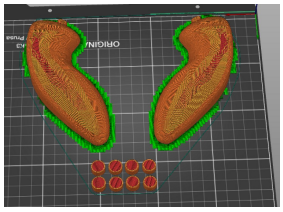
Screen Enclosure:
- Small hollow box with lid for screen
- Arduino enclosed inside alongside battery and wiring
- Holes for controller connection and USB access



Controller:
- Wii "nunchuck" like design with 2 buttons at top and joystick
- Wiring extends out from top of controller and attaches to screen enclosure
- Breadboard housed inside controller for joystick and buttons
- Ergonomic design for comfortability
- Printed in 2 halves that will be glued together

3D-MODELS:

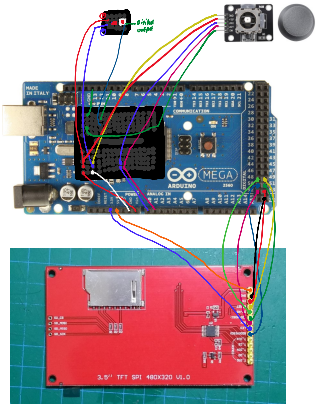




MATERIALS AND COMPONENTS REQUIRED:

- TFT display
- Button x1
- Small Breadboard
- Big Breadboard
- Arduino Mega
- Jumper Cables
- Wires
- Resistor
- 9V battery

THE CIRCUIT:



THE CODE:

```
#include <TFT_eSPI.h>
#include <stdbool.h>

// initialize TFT display
TFT_eSPI tft = TFT_eSPI();

#define BLACK 0x0000
#define WHITE 0xFFFF
#define joystickx A0
#define joysticky A1

// set paddle variables
int paddleWidth = 5;
int paddleSize = 50;
int paddlePositiony = tft.height()/2;
int paddlePositionx = 5;

// set ball variables
int ballPositionX = 60;
int ballPositionY = 60;
int ballDirectionX = 5;
int ballDirectionY = 5;
int ballSize = 3;

// set score variables
int playerScore = 0;
int computerScore = 0;

// other variables

int16_t rpaddle_x = tft.width() - 15;

int16_t rpaddle_y = tft.height()/2;

int16_t rpaddle_d = -5;

int16_t lpaddle_ball_t = tft.width() - tft.width() / 4;
int16_t rpaddle_ball_t = tft.height() / 4;

int16_t target_y = 0;


void setup() {
  // initialize TFT display
  tft.init();
  tft.setRotation(1);
  tft.fillScreen(TFT_BLACK);

  // initialize push buttons
  pinMode(joystickx, INPUT);
  pinMode(joysticky, INPUT);
}

void loop() {

  int oldpositiony = paddlePositiony;
  // move paddles
  if(analogRead(joystickx) < 450 && paddlePositiony < tft.height() - 13 - paddleSize) {
    paddlePositiony = paddlePositiony +5;
  }
  if(analogRead(joystickx) > 540 && paddlePositiony > 18) {
    paddlePositiony = paddlePositiony -5;
  }

  // move ball
  tft.fillCircle(ballPositionX, ballPositionY, ballSize, TFT_BLACK);
  ballPositionX += ballDirectionX;
  ballPositionY += ballDirection;
  tft.fillCircle(ballPositionX, ballPositionY, ballSize, TFT_WHITE);

  // check if ball hits paddle USE CHECKCOLLISION
  //if(ballPositionX + ballSize >= tft.width() - 10 && ballPositionX + ballSize >= paddlePosition && ballPositionX - ballSize <= paddlePosition + paddleSize) {
  //  ballDirectionX = -random(2, 15);
  //}
  if(checkCollision(paddlePositionx, paddlePositiony, paddleWidth, paddleSize, ballPositionX, ballPositionY, ballSize, ballSize) == true) {
    ballDirectionX = -ballDirectionX;
    ballDirectionX = -ballDirectionX;
  }
  if(checkCollision(rpaddle_x, rpaddle_y, paddleWidth, paddleSize, ballPositionX, ballPositionY, ballSize, ballSize) == true) {
    ballDirectionY = -ballDirectionY;
    ballDirectionX = -ballDirectionX;
  }

  // check if ball hits paddle USE CHECKCOLLISION
  //if(ballPositionX - ballSize <= 10 || ballPositionX + ballSize >= tft.width() - 10) {
  //  ballDirectionX = -random(2, 15);
  //}

  // check if ball hits top or bottom
  if(ballPositionY - ballSize <= 15 || ballPositionY + ballSize >= tft.height() - 15) {
    ballDirectionY = -ballDirectionY;
  }
  int nerf = random(0,10);
  if(nerf>7) {
    calc_target_y();
  }

  rpaddle();

// check if ball hits left or right
  if(ballPositionX - ballSize <= 0) {
    computerScore++;
  }
  if(ballPositionX + ballSize >= tft.width()) {
    playerScore++;
  }
  if(ballPositionX - ballSize <= 0 || ballPositionX + ballSize >= tft.width()) {
    tft.fillCircle(ballPositionX, ballPositionY, ballSize, TFT_BLACK);
    ballPositionX = tft.width() / 2;
    ballPositionY = tft.height() / 2;
    ballDirectionX = random(5, 15);
    ballDirectionY = random(-7, 7);
    tft.fillRect(0, 0, tft.width(), 15, TFT_BLACK);
    tft.setCursor(5, 0);
    tft.setTextColor(TFT_WHITE);
    tft.setTextSize(2);
    tft.print("Player: ");
    tft.print(playerScore);
    tft.setCursor(tft.width() - 150, 0);
    tft.print("Computer: ");
    tft.print(computerScore);
  }

  // update TFT display
  tft.fillRect(0, tft.height() - 15, tft.width(), 5, TFT_WHITE);
  tft.fillRect(0, 15, tft.width(), 5, TFT_WHITE);
  tft.fillRect(0, tft.height() - 10, tft.width(), 10, TFT_BLACK);
  tft.fillRect(paddlePositionx, oldpositiony, paddleWidth, paddleSize, TFT_BLACK);
  tft.fillRect(paddlePositionx, paddlePositiony, paddleWidth, paddleSize, TFT_WHITE);
  tft.setCursor(5, 0);
  tft.setTextColor(TFT_WHITE);
  tft.setTextSize(2);
  tft.print("Player: ");
  tft.setCursor(tft.width() - 150, 0);
  tft.print("Computer: ");

  //if(computerScore == 5) {
    //tft.fillRect(0, 0, tft.width(), tft.height(), TFT_BLACK);
    //tft.setCursor(tft.width()/2 - 50, tft.height()/2);
    //tft.setTextColor(TFT_WHITE);
    //tft.setTextSize(10);
    //tft.print("COMPUTER WINS");
  //}
  //if(playerScore >= 5) {
    //tft.fillRect(0, 0, tft.width(), tft.height(), TFT_BLACK);
    //tft.setCursor(tft.width()/2 - 50, tft.height()/2);
    //tft.setTextColor(TFT_WHITE);
    //tft.setTextSize(10);
    //tft.print("Player WINS");
  //}
}

void rpaddle() {

  if (rpaddle_d == 5) {
    tft.fillRect(tft.width() - 15, rpaddle_y, paddleWidth, 1, BLACK);
  }
  else if (rpaddle_d == -5) {
    tft.fillRect(tft.width() - 15, rpaddle_y + paddleSize - 5, paddleWidth, 1, BLACK);
  }
  tft.fillRect(tft.width() - 15, rpaddle_y, paddleWidth, paddleSize, BLACK);
  rpaddle_y = rpaddle_y + rpaddle_d;

  if (ballDirectionX < 0) rpaddle_d = 0;
  else {
    if (rpaddle_y + paddleSize / 2 == target_y) rpaddle_d = 0;
    else if (rpaddle_y + paddleSize / 2 > target_y) rpaddle_d = -5;
    else rpaddle_d = 5;
  }

  if (rpaddle_y + paddleSize >= tft.height()-10 && rpaddle_d == 5) rpaddle_d = 0;
  else if (rpaddle_y <= 18 && rpaddle_d == -5) rpaddle_d = 0;

  tft.fillRect(tft.width() - 15, rpaddle_y, paddleWidth, paddleSize, WHITE);
}

void calc_target_y() {
  int16_t target_x;
  int16_t reflections;
  int16_t y;

  if (ballDirectionX > 0) {
    target_x = tft.width() - ballSize;
  }
  else {
    target_x = -1 * (tft.width() - ballSize);
  }

  y = abs(target_x * (ballDirectionY / ballDirectionX) + ballPositionY);

  reflections = floor(y / tft.height());
```

paddle variables

ball variables

score variables

initializing display and buttons

movement of the ball

ball movement after hitting paddle

collision detection

ball position determination - does it go left or right?

updating display after movement

printing winner after reaching score limit

right paddle logic

calculate y position of ball for computer paddle

```
  if (ballDirectionX > 0) {
    target_x = tft.width() - ballSize;
  }
  else {
    target_x = -1 * (tft.width() - ballSize);
  }

  y = abs(target_x * (ballDirectionY / ballDirectionX) + ballPositionY);

  reflections = floor(y / tft.height());

  if (reflections % 2 == 0) {
    target_y = y % tft.height();
  }
  else {
    target_y = tft.height() - (y % tft.height());
  }
}
// bounding box collision detection
boolean checkCollision(int x1, int y1, int width1, int height1, int x2, int y2, int width2,
int height2){
  boolean hit = false;
  if ((((x2 + width2) >= x1) && (x2 <= (x1 + width1))) && (((y2 + height2) >= y1) && (y2 <=
(y1 + height1)))) {
      hit = true;
    }
  return hit;
}
```

calculate y position of ball for computer to move paddle