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

\_\_\_\_\_ \_ \_ \_\_\_ \_ \_\_ \_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_ \_\_

/ \_\_\_|| \ | | / \_ \ | | / /| \_\_\_| / /| \_\_\_ \| \_\_\_||\_ \_|/ \_ \\ \

\ `--. | \| |/ /\_\ \| |/ / | |\_\_ | | | |\_/ /| |\_\_ | | / /\_\ \| |

`--. \| . ` || \_ || \ | \_\_| | | | \_\_\_ \| \_\_| | | | \_ || |

/\\_\_/ /| |\ || | | || |\ \| |\_\_\_ | | | |\_/ /| |\_\_\_ | | | | | || |

\\_\_\_\_/ \\_| \\_/\\_| |\_/\\_| \\_/\\_\_\_\_/ | | \\_\_\_\_/ \\_\_\_\_/ \\_/ \\_| |\_/| |

\\_\ /\_/

\_ \_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_ \_\_\_\_\_\_\_ \_ \_ \_

/ )/ )( \_\_\_\_ \\\_\_ \_\_/( \_\_\_\_ \( \_\_\_\_ )( ( /|( \_\_\_ )( \ ( \( \

\_\_\_ / // / | ( \/ ) ( | ( \/| ( )|| \ ( || ( ) || ( \ \\ \ \_\_\_

(\_\_\_)/ // / | (\_\_ | | | (\_\_ | (\_\_\_\_)|| \ | || (\_\_\_) || | \ \\ \(\_\_\_)

\_\_\_( (( ( | \_\_) | | | \_\_) | \_\_)| (\ \) || \_\_\_ || | ) )) )\_\_\_

(\_\_\_)\ \\ \ | ( | | | ( | (\ ( | | \ || ( ) || | / // /(\_\_\_)

\ \\ \ | (\_\_\_\_/\ | | | (\_\_\_\_/\| ) \ \\_\_| ) \ || ) ( || (\_\_\_\_/\ / // /

\\_)\\_)(\_\_\_\_\_\_\_/ )\_( (\_\_\_\_\_\_\_/|/ \\_\_/|/ )\_)|/ \|(\_\_\_\_\_\_\_/(\_/(\_/

Written by: Zachariah Weber,

Alixander Richards-Thompson,

and Clayton Wahlstrom

Designed for: Tiva TM4C123GXL with Adafruit ST7789 Display

Version: Beta(0.07)

Note: The goal of this version is to finalize the code.

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

/\* DISPLAY Libraries, defines, and pins.\*/

#include <Adafruit\_GFX.h> // Core graphics library

#include <Adafruit\_ST7789.h> // Hardware-specific library for ST7789

#include <spiE.h>

#define TFT\_CS 12 // PA\_3 // Chip Select

#define TFT\_RST 24 // PD\_1 // Reset pin

#define TFT\_DC 25 // PD\_2 // Data/Command or Register Select

#define TFT\_BL 23 // PD\_0 // Backlight

Adafruit\_ST7789 tft **=** Adafruit\_ST7789**(&**SPI0**,** TFT\_CS**,** TFT\_DC**,** TFT\_RST**);** //Create screen object

/\* Game Defines \*/

/\* Some of these may get removed when the MENU is implemented \*/

#define ROT 0 // Screen rotation: 0 = Vertical, 1 = Horizontal, 4 = Mirrored V, 5 = Mirrored H

#if ROT == 1 || ROT == 5

#define DISWIDTH 320 // Max Width, 320

#define DISHEIGHT 192 // Top and bottom border of 24 Pixels, so max of 192

#endif

#if ROT == 0 || ROT == 4

#define DISWIDTH 240 // Max Width, 240

#define DISHEIGHT 272 // Top an bottom border of 24 pixels, so max of 272

#endif

const uint8\_t JOYSTICKS**[**4**][**2**]** **=** **{** //Player Joystick pins (X,Y)

**{**29**,** 28**},** **{**27**,** 26**},** **{**2**,** 5**},** **{**6**,** 7**}**

**};**

/\* Optimization Macros: Optimize by using uint32\_t arrays as 2bit arrays \*/

//Write a 2 bit element to a uint32 array.

#define WRITE2ARRAY(A,k,m) ((A[k / 16] &= ~(3 << ((k%16) \* 2))) |= (m << (k%16) \* 2))

//Read a 2 bit element from a uint32 array.

#define READ2ARRAY(A,k) (A[k / 16] & (3 << ((k%16) \* 2))) >> ((k%16) \* 2)

/\*Global Enumerations\*/

enum **{** JoystickUP **=** 0**,** JoystickLEFT**,** JoystickDOWN**,** JoystickRIGHT**,** JoystickNULL **};**

enum **{** Player1 **=** 0**,** Player2**,** Player3**,** Player4 **};**

enum **{** No **=** 0**,** Yes **};**

/\* Function and Class Declarations: Necessary for Snake class \*/

void setPixel**(**int**,** int**,** uint16\_t**);**

void FoodGen**();**

uint8\_t controller**(**uint8\_t**);**

class Snake**;**

/\* Global Variables \*/

volatile bool button **=** 0**;**

bool gameRunning **=** 0**;**

uint8\_t players **=** 0**;**

uint8\_t food**[**2**];**

uint8\_t scale**;**

uint8\_t**\*\*** displayMatrix **=** **{** **NULL** **};**

Snake**\*** snakePtrs**[**4**]** **=** **{** **NULL** **};** //Up to 4 players

/\* Snake class: Easy snake command.

As game board sizes increase, more ram is needed to run this game.

Since we wanted this to be a multiplayer game, we determined that a class

would be an easy way to dynamically generate snakes for each game instance.

This also makes controlling the snakes easier.

\*/

class Snake **{**

public**:**

uint16\_t color**;** // Snake color (https://www.demmel.com/ilcd/help/16BitColorValues.htm)

uint8\_t dir**;** // Snake movement direction (Can't be smaller)

uint16\_t ends**[**2**][**2**];** /\* The locations of the ends of the snake. (320 greater than 255, so

16bit) ends[0][\*] is the head of the snake. ends[1][\*] is the tail\*/

int8\_t alive **=** 1**;** // Whether this snake is alive. (defaults to 1)

uint32\_t snakeLength**;** // Current snake length (At a scale of 1, 320\*240, this must be 32bit)

/\* Constructor: Snake Setup

Take an initial tail position, length, direction, and

snake color to generate that snake on the displaymatrix.

\*/

Snake**(**uint8\_t x**,** uint8\_t y**,** uint8\_t sLength**,** uint8\_t sDir**,** uint8\_t i**,** uint16\_t sColor **=** ST77XX\_GREEN**)**

**:** color**(**sColor**),** dir**(**sDir**),** direction**(**sDir**),** index**(**i**)** **{**

//Generate Segments: Each segment is 2bits; round up.

segments **=** **new** uint32\_t**[(((**DISWIDTH **/** scale**)** **\*** **(**DISHEIGHT **/** scale**)** **+** 15**)** **/** 16**)];**

snakeLength **=** sLength **-** 1**;** //Number of segments after the head

ends**[**1**][**0**]** **=** x**;** //Tail Start

ends**[**1**][**1**]** **=** y**;**

**switch** **(**dir**)** **{** //Head Start

**case** JoystickUP**:**

ends**[**0**][**0**]** **=** ends**[**1**][**0**];**

ends**[**0**][**1**]** **=** ends**[**1**][**1**]** **-** snakeLength**;**

**break;**

**case** JoystickLEFT**:**

ends**[**0**][**0**]** **=** ends**[**1**][**0**]** **-** snakeLength**;**

ends**[**0**][**1**]** **=** ends**[**1**][**1**];**

**break;**

**case** JoystickDOWN**:**

ends**[**0**][**0**]** **=** ends**[**1**][**0**];**

ends**[**0**][**1**]** **=** ends**[**1**][**1**]** **+** snakeLength**;**

**break;**

**case** JoystickRIGHT**:**

ends**[**0**][**0**]** **=** ends**[**1**][**0**]** **+** snakeLength**;**

ends**[**0**][**1**]** **=** ends**[**1**][**1**];**

**break;**

**}**

//Draw snakes on displayMatrix & write segment relationships

**for** **(**uint16\_t i **=** 0**;** i **<** snakeLength**;** i**++)** **{**

**switch** **(**dir**)** **{**

**case** JoystickUP**:**

displayMatrix**[**ends**[**1**][**0**]][**ends**[**1**][**1**]** **-** i**]** **=** 2**;**

setPixel**(**ends**[**1**][**0**],** ends**[**1**][**1**]** **-** i**,** color**);**

**break;**

**case** JoystickLEFT**:**

displayMatrix**[**ends**[**1**][**0**]** **-** i**][**ends**[**1**][**1**]]** **=** 2**;**

setPixel**(**ends**[**1**][**0**]** **-** i**,** ends**[**1**][**1**],** color**);**

**break;**

**case** JoystickDOWN**:**

displayMatrix**[**ends**[**1**][**0**]][**ends**[**1**][**1**]** **+** i**]** **=** 2**;**

setPixel**(**ends**[**1**][**0**],** ends**[**1**][**1**]** **+** i**,** color**);**

**break;**

**case** JoystickRIGHT**:**

displayMatrix**[**ends**[**1**][**0**]** **+** i**][**ends**[**1**][**1**]]** **=** 2**;**

setPixel**(**ends**[**1**][**0**]** **+** i**,** ends**[**1**][**1**],** color**);**

**break;**

**}**

WRITE2ARRAY**(**segments**,** i**,** dir**);**

**}**

//Draw the head of the snake

setPixel**(**ends**[**0**][**0**],** ends**[**0**][**1**],** color**);**

PrintScore**();**

**}**

/\* Deconstructor: Snake Teardown

For now, simply free the memory reserved for "segments".

\*/

**~**Snake**()** **{**

**delete** segments**;**

**}**

/\* SnakeMove:

Move the snake in the direction of its "dir" variable.

If the snake eats, increase length and update score.

\*/

void SnakeMove**()** **{**

displayMatrix**[**ends**[**0**][**0**]][**ends**[**0**][**1**]]** **=** 2**;** // The current head position is not displayed on

// displayMatrix until subsequent move

**if** **(((**dir **+** 2**)** **%** 4 **!=** direction**)** **&&** dir **!=** JoystickNULL**)** // Restrict players from flipping around

direction **=** dir**;**

**}**

//Move snake head in a direction

**switch** **(**direction**)** **{**

**case** JoystickUP**:**

ends**[**0**][**1**]** **-=** 1**;**

**break;**

**case** JoystickLEFT**:**

ends**[**0**][**0**]** **-=** 1**;**

**break;**

**case** JoystickDOWN**:**

ends**[**0**][**1**]** **+=** 1**;**

**break;**

**case** JoystickRIGHT**:**

ends**[**0**][**0**]** **+=** 1**;**

**break;**

**}**

//Update "segments" with newest direction

uint8\_t saved2bit **=** direction**;**

**for** **(**uint16\_t m **=** 0**;** m **<** **(**snakeLength **/** 16**)** **+** 1**;** m**++)** **{**

uint8\_t read2bit **=** READ2ARRAY**(**segments**,** **(**m **\*** 16 **+** 15**));**

segments**[**m**]** **<<=** 2**;**

WRITE2ARRAY**(**segments**,** **(**m **\*** 16**),** saved2bit**);**

saved2bit **=** read2bit**;**

**}**

// Did you eat food?

**if** **((**ends**[**0**][**0**]** **==** food**[**0**]** **&&** ends**[**0**][**1**]** **==** food**[**1**])** **||**

**(**displayMatrix**[**ends**[**0**][**0**]][**ends**[**0**][**1**]]** **==** 3**))** **{**

snakeLength **+=** 1**;** // Grow Snake

PrintScore**();** // Update Score

Serial**.**println**(**"Nom"**);**

**}**

**else** **{**

// Only "move" the end of the snake if it didn't eat.

setPixel**(**ends**[**1**][**0**],** ends**[**1**][**1**],** ST77XX\_BLACK**);**

displayMatrix**[**ends**[**1**][**0**]][**ends**[**1**][**1**]]** **=** 0**;**

//Update Tail Position

**switch** **(**READ2ARRAY**(**segments**,** snakeLength**))** **{**

**case** JoystickUP**:**

ends**[**1**][**1**]** **-=** 1**;**

**break;**

**case** JoystickLEFT**:**

ends**[**1**][**0**]** **-=** 1**;**

**break;**

**case** JoystickDOWN**:**

ends**[**1**][**1**]** **+=** 1**;**

**break;**

**case** JoystickRIGHT**:**

ends**[**1**][**0**]** **+=** 1**;**

**break;**

**}**

**}**

**}**

/\* SnakeCheck:

Check to see if the snake has eaten food or died.

\*/

uint8\_t SnakeCheck**()** **{**

// Did the snake eat regenerating food?

**if** **(**ends**[**0**][**0**]** **==** food**[**0**]** **&&** ends**[**0**][**1**]** **==** food**[**1**])** **{**

FoodGen**();**

**}**

// Did the snake run into an obstacle?

**if** **(**displayMatrix**[**ends**[**0**][**0**]][**ends**[**0**][**1**]]** **==** 1 **||** displayMatrix**[**ends**[**0**][**0**]][**ends**[**0**][**1**]]** **==** 2**)** **{**

alive **=** 0**;**

**return** 1**;** //This snake has died

**}**

// Did the snake run into another snake's head?

// Check head location against every other snake excluding itself.

**for** **(**uint8\_t i **=** **(**index **+** 1**);** **(**i **!=** index**)** **&&** **(**players **>** 1**);** **)** **{**

**if** **((**ends**[**0**][**0**]** **==** snakePtrs**[**i**]->**ends**[**0**][**0**])** **&&** **(**ends**[**0**][**1**]** **==** snakePtrs**[**i**]->**ends**[**0**][**1**]))** **{**

// If it did, and that snake was dead, clear the head pixel.

**if** **(**snakePtrs**[**i**]->**alive **==** 0 **&&** players **>** 2**)** **{**

setPixel**(**ends**[**0**][**0**],** ends**[**0**][**1**],** ST77XX\_BLACK**);**

**}**

alive **=** 0**;**

**return** 1**;** //This snake has died

**}**

**if** **(++**i **>** **(**players **-** 1**))** **{**

i **=** 0**;**

**}**

**}**

// If the snake isn't dead, place the head pixel. (Graphics work around)

setPixel**(**ends**[**0**][**0**],** ends**[**0**][**1**],** color**);**

**return** 0**;**

**}**

/\* SetDir:

Set the current moving direction of the snake

\*/

void SetDir**(**uint8\_t nDir**)** **{**

dir **=** nDir**;**

**}**

/\* Turn2Food: Multiplayer Afterlife \*/

void Turn2Food**()** **{**

uint16\_t tail**[**2**]** **=** **{** ends**[**1**][**0**],** ends**[**1**][**1**]** **};** // Make temp variable

// Turn every other snake segment into food from the tail to the head

**for** **(**uint16\_t i **=** snakeLength**;** i **>** 0**;** i**--)** **{**

**if** **(**i **%** 2 **==** 0**)** **{**

setPixel**(**tail**[**0**],** tail**[**1**],** ST77XX\_BLACK**);**

displayMatrix**[**tail**[**0**]][**tail**[**1**]]** **=** 0**;**

**}**

**else** **{**

setPixel**(**tail**[**0**],** tail**[**1**],** ST77XX\_YELLOW**);**

displayMatrix**[**tail**[**0**]][**tail**[**1**]]** **=** 3**;**

**}**

**switch** **(**READ2ARRAY**(**segments**,** **(**i **-** 1**)))** **{**

**case** JoystickUP**:**

tail**[**1**]** **-=** 1**;**

**break;**

**case** JoystickLEFT**:**

tail**[**0**]** **-=** 1**;**

**break;**

**case** JoystickDOWN**:**

tail**[**1**]** **+=** 1**;**

**break;**

**case** JoystickRIGHT**:**

tail**[**0**]** **+=** 1**;**

**break;**

**}**

**}**

ends**[**0**][**0**]** **=** ends**[**0**][**1**]** **=** ends**[**1**][**0**]** **=** ends**[**1**][**1**]** **=** 0**;** // Move snake off play area

alive **=** **-**1**;** // Snake has already been processed

**}**

/\* PrintScore: Prints score \*/

void PrintScore**()** **{**

// Print score with black background if it is a single player game

**if** **(**players **==** 1**)** **{**

tft**.**fillRect**(**0**,** 0**,** DISWIDTH **/** 2**,** 24**,** ST77XX\_BLACK**);**

tft**.**setTextColor**(**ST77XX\_WHITE**);**

tft**.**setCursor**(**5**,** 4**);**

tft**.**setTextSize**(**2**);**

tft**.**print**(**snakeLength **\*** 10**);** tft**.**print**(**"pts"**);**

**}**

//Otherwise, set score background to snake color

**else** **{**

**switch** **(**index**)** **{**

**case** Player1**:**

tft**.**fillRect**(**0**,** 0**,** DISWIDTH **/** 2**,** 24**,** color**);**

tft**.**setTextColor**(**ST77XX\_BLACK**);**

tft**.**setCursor**(**5**,** 4**);**

tft**.**setTextSize**(**2**);**

tft**.**print**(**snakeLength **\*** 10**);** tft**.**print**(**"pts"**);**

**break;**

**case** Player2**:**

tft**.**fillRect**(**DISWIDTH **/** 2**,** 0**,** DISWIDTH **/** 2**,** 24**,** color**);**

tft**.**setTextColor**(**ST77XX\_BLACK**);**

tft**.**setCursor**(**DISWIDTH **/** 2 **+** 5**,** 4**);**

tft**.**setTextSize**(**2**);**

tft**.**print**(**snakeLength **\*** 10**);** tft**.**print**(**"pts"**);**

**break;**

**case** Player3**:**

tft**.**fillRect**(**0**,** DISHEIGHT **+** 24**,** DISWIDTH **/** 2**,** 24**,** color**);**

tft**.**setTextColor**(**ST77XX\_BLACK**);**

tft**.**setCursor**(**5**,** DISHEIGHT **+** 28**);**

tft**.**setTextSize**(**2**);**

tft**.**print**(**snakeLength **\*** 10**);** tft**.**print**(**"pts"**);**

**break;**

**case** Player4**:**

tft**.**fillRect**(**DISWIDTH **/** 2**,** DISHEIGHT **+** 24**,** DISWIDTH **/** 2**,** 24**,** color**);**

tft**.**setTextColor**(**ST77XX\_BLACK**);**

tft**.**setCursor**(**DISWIDTH **/** 2 **+** 5**,** DISHEIGHT **+** 28**);**

tft**.**setTextSize**(**2**);**

tft**.**print**(**snakeLength **\*** 10**);** tft**.**print**(**"pts"**);**

**break;**

**}**

**}**

**}**

private**:**

uint8\_t direction**;** // Previous snake direction (Fixes moving up as start direction)

uint8\_t index**;** // Snake ID number

uint32\_t**\*** segments**;** // Dynamically generated snake segment array. (32bit for efficiency)

**};**

void setup**()** **{**

analogWrite**(**TFT\_BL**,** 255**);** // Set Backlight Intensity set to "10" for presentation

tft**.**init**(**240**,** 320**);** // Init and clear screen

tft**.**setRotation**(**ROT**);** // Set rotation/ horizontally flip screen for presentation with "4"

randomSeed**(**analogRead**(**0**));** // Randomize food Seed

Serial**.**begin**(**9600**);**

pinMode**(**PUSH2**,** INPUT**);** //Create our button interrupt

attachInterrupt**(**digitalPinToInterrupt**(**PUSH2**),** pressedButton**,** FALLING**);**

**}**

void loop**()** **{**

//Initial Clear screen (Remove static)

tft**.**fillScreen**(**ST77XX\_BLACK**);**

// Fill variables

static uint8\_t alive**;**

static unsigned long referenceScreen **=** 0**;**

static unsigned long referenceDir **=** 0**;**

title**();**

players **=** playerCount**();** // Set player count

alive **=** players**;**

scale **=** gameScale**();** // Set screen size, Resolution Downscaling Multiplayer: 1,2,4,8,16

replay**:** // Replay Game goto

// Generate displayMatrix

displayMatrix **=** **new** uint8\_t **\*** **[**DISWIDTH **/** scale**];**

displayMatrix**[**0**]** **=** **new** uint8\_t**[(**DISWIDTH **/** scale**)** **\*** **(**DISHEIGHT **/** scale**)];**

**for** **(**int i **=** 1**;** i **<** **(**DISWIDTH **/** scale**);** i**++)** **{**

displayMatrix**[**i**]** **=** displayMatrix**[**i **-** 1**]** **+** **(**DISHEIGHT **/** scale**);**

**}**

// Fill Display Matrix; 1 is wall, 0 is open

**for** **(**uint8\_t h **=** 0**;** h **<** **(**DISHEIGHT **/** scale**);** h**++)** **{**

**for** **(**uint8\_t w **=** 0**;** w **<** **(**DISWIDTH **/** scale**);** w**++)** **{**

**if** **(**h **==** 0 **||** w **==** 0 **||** h **==** **((**DISHEIGHT **/** scale**)** **-** 1**)** **||** w **==** **((**DISWIDTH **/** scale**)** **-** 1**))** **{**

displayMatrix**[**w**][**h**]** **=** 1**;**

setPixel**(**w**,** h**,** ST77XX\_WHITE**);**

**}**

**else** **{**

displayMatrix**[**w**][**h**]** **=** 0**;**

**}**

**}**

**}**

// Generate Snakes

**if** **(**players **>** 1**)** **{** //Multiplayer Snake colors and settings.

**for** **(**uint8\_t i **=** 0**;** i **<** players**;** i**++)** **{**

**switch** **(**i**)** **{**

**case** Player1**:**

snakePtrs**[**Player1**]** **=** **new** Snake**(**2**,** 4**,** 3**,** JoystickRIGHT**,** Player1**,** ST77XX\_GREEN**);**

**break;**

**case** Player2**:**

snakePtrs**[**Player2**]** **=** **new** Snake**((**DISWIDTH **/** scale**)** **-** 3**,** 4**,** 3**,** JoystickLEFT**,**

Player2**,** ST77XX\_CYAN**);**

**break;**

**case** Player3**:**

snakePtrs**[**Player3**]** **=** **new** Snake**(**2**,** **(**DISHEIGHT **/** scale**)** **-** 5**,** 3**,** JoystickRIGHT**,**

Player3**,** ST77XX\_MAGENTA**);**

**break;**

**case** Player4**:**

snakePtrs**[**Player4**]** **=** **new** Snake**((**DISWIDTH **/** scale**)** **-** 3**,** **(**DISHEIGHT **/** scale**)** **-** 5**,** 3**,**

JoystickLEFT**,** Player4**,** ST77XX\_RED**);**

**break;**

**}**

**}**

**}**

**else** **{** //Single player Snake settings

snakePtrs**[**Player1**]** **=** **new** Snake**(**2**,** 4**,** 3**,** JoystickRIGHT**,** Player1**);**

**}**

FoodGen**();**

gameRunning **=** 1**;**

//Snake Game

**while** **(true)** **{**

//Pause game if the button was pressed

**if** **(**button **==** 1**)** **{**

button **=** 0**;**

paused**();**

referenceScreen **=** millis**();**

referenceDir **=** millis**();**

**}**

unsigned long current **=** millis**();**

// Update controller input every 10ms

**if** **(**current **-** referenceDir **>=** 10**)** **{**

referenceDir **=** current**;**

**for** **(**int i **=** 0**;** i **<** players**;** i**++)** **{**

snakePtrs**[**i**]->**SetDir**(**controller**(**i**));**

**}**

**}**

// Update the snakes every .5 seconds

current **=** millis**();**

**if** **(**current **-** referenceScreen **>=** 500**)** **{**

referenceScreen **=** current**;**

**for** **(**int i **=** 0**;** i **<** players**;** i**++)** **{** // Move all alive snakes

**if** **(**snakePtrs**[**i**]->**alive **==** 1**)** **{**

snakePtrs**[**i**]->**SnakeMove**();**

**}**

**}**

**for** **(**int i **=** 0**;** i **<** players**;** i**++)** **{** // Check all alive snakes

**if** **(**snakePtrs**[**i**]->**alive **==** 1**)** **{**

alive **-=** snakePtrs**[**i**]->**SnakeCheck**();**

**}**

**}**

**for** **(**int i **=** 0**;** i **<** players**;** i**++)** **{** // Delete dead snakes

**if** **(**snakePtrs**[**i**]->**alive **==** 0 **&&** players **>** 2**)** **{**

snakePtrs**[**i**]->**Turn2Food**();**

**}**

**}**

//End screen conditions & screens

**if** **(**alive **==** 1 **&&** players **!=** 1**)** **{** // If a multiplayer game has ended

**for** **(**int i **=** 0**;** i **<** players**;** i**++)** **{**

**if** **(**snakePtrs**[**i**]->**alive **==** 1**)** **{**

// Math used to center messages in either orientation

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 73**,** **(**DISHEIGHT**)** **/** 2 **-** 28**,** 146**,** 106**,**

ST77XX\_WHITE**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 70**,** **(**DISHEIGHT**)** **/** 2 **-** 25**,** 140**,** 100**,**

ST77XX\_BLACK**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 60**,** **(**DISHEIGHT**)** **/** 2 **-** 15**);**

tft**.**setTextColor**(**snakePtrs**[**i**]->**color**);**

tft**.**setTextSize**(**3**);**

tft**.**print**(**"Player"**);** tft**.**println**(**i **+** 1**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 53**,** **(**DISHEIGHT**)** **/** 2 **+** 12**);**

tft**.**setTextSize**(**4**);**

tft**.**println**(**"WINS!"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 53**,** **(**DISHEIGHT**)** **/** 2 **+** 50**);**

tft**.**setTextSize**(**2**);**

tft**.**print**((**snakePtrs**[**i**]->**snakeLength**)** **\*** 10**);** tft**.**print**(**"pts"**);**

gameRunning **=** 0**;**

**}**

**}**

**}**

**else** **if** **(**alive **==** 0 **&&** players **==** 1**)** **{** // If a single player game has ended

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 73**,** **(**DISHEIGHT**)** **/** 2 **-** 28**,** 146**,** 106**,** ST77XX\_WHITE**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 70**,** **(**DISHEIGHT**)** **/** 2 **-** 25**,** 140**,** 100**,** ST77XX\_BLACK**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 35**,** **(**DISHEIGHT**)** **/** 2 **-** 13**);**

tft**.**setTextColor**(**ST77XX\_RED**);**

tft**.**setTextSize**(**3**);**

tft**.**print**(**"Game"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 53**,** **(**DISHEIGHT**)** **/** 2 **+** 12**);**

tft**.**setTextSize**(**4**);**

tft**.**println**(**"OVER!"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 53**,** **(**DISHEIGHT**)** **/** 2 **+** 50**);**

tft**.**setTextSize**(**2**);**

tft**.**print**((**snakePtrs**[**0**]->**snakeLength**)** **\*** 10**);** tft**.**print**(**"pts"**);**

gameRunning **=** 0**;**

**}**

**else** **if** **(**alive **==** 0**)** **{** // If there is a tie in a multiplayer game

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 73**,** **(**DISHEIGHT**)** **/** 2 **-** 3**,** 146**,** 56**,** ST77XX\_WHITE**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 70**,** **(**DISHEIGHT**)** **/** 2**,** 140**,** 50**,** ST77XX\_BLACK**);**

tft**.**setTextColor**(**ST77XX\_YELLOW**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 42**,** **(**DISHEIGHT**)** **/** 2 **+** 10**);**

tft**.**setTextSize**(**4**);**

tft**.**println**(**"TIE!"**);**

gameRunning **=** No**;**

**}**

**}**

//Continue Screen if game has ended

**if** **(**gameRunning **==** No**)** **{**

delay**(**800**);** // Artistic wait before the menu starts. Nothing happens here anyway.

gameRunning **=** playAgain**();**

//Delete Dynamic Variables.

**for** **(**uint8\_t i **=** 0**;** i **<** players**;** i**++)** **{**

**delete** snakePtrs**[**i**];**

**}**

**delete** **[]** displayMatrix**;**

**if** **(**gameRunning **==** Yes**)** **{**

alive **=** players**;**

//Jump to after the menus

tft**.**fillScreen**(**ST77XX\_BLACK**);**

**goto** replay**;**

**}**

**else** **{**

//Jump back to Title Screen

**break;**

**}**

**}**

**}**

**}**

/\*\* Game Functions \*\*/

/\* Draw scaled "Pixel" at (x,y) location in "color" \*/

void setPixel**(**int x**,** int y**,** uint16\_t color**)** **{**

tft**.**fillRect**(**x **\*** scale**,** **(**y **\*** scale**)** **+** 24**,** scale**,** scale**,** color**);**

**}**

/\* Generate Food position \*/

void FoodGen**()** **{**

uint8\_t overlap**;**

**do** **{**

overlap **=** 0**;**

//Generate food positions until it doesn't overlap with an existing element.

food**[**0**]** **=** random**((**DISWIDTH **/** scale**));**

food**[**1**]** **=** random**((**DISHEIGHT **/** scale**));**

**for** **(**uint8\_t i **=** 0**;** i **<** players**;** i**++)** **{**

**if** **((**food**[**0**]** **==** snakePtrs**[**i**]->**ends**[**0**][**0**]**

**&&** food**[**1**]** **==** snakePtrs**[**i**]->**ends**[**0**][**1**])**

**||** displayMatrix**[**food**[**0**]][**food**[**1**]]** **!=** 0**)** **{**

overlap **=** 1**;**

**}**

**}**

**}** **while** **(**overlap**);**

// Draw food

displayMatrix**[**food**[**0**]][**food**[**1**]]** **=** 3**;**

setPixel**(**food**[**0**],** food**[**1**],** ST77XX\_YELLOW**);**

**}**

/\* Take controller direction for snake "index"\*/

uint8\_t controller**(**uint8\_t index**)** **{**

uint8\_t direct **=** JoystickNULL**;**

int16\_t Y **=** map**(**analogRead**(**JOYSTICKS**[**index**][**1**]),** 0**,** 4095**,** **-**2048**,** 2048**);**

int16\_t X **=** map**(**analogRead**(**JOYSTICKS**[**index**][**0**]),** 0**,** 4095**,** 2048**,** **-**2048**);**

//Dead zone for a radius of 300 out of 4096

**if** **((**X **\*** X **+** Y **\*** Y**)** **<=** 90000**)** **{**

X **=** 0**;**

Y **=** 0**;**

**}**

//If the X direction is bigger than Y, then is it left or right?

**else** **if** **(**X **\*** X **>** Y**\*** Y**)** **{**

**if** **(**X **>** 0**)** **{**

direct **=** JoystickLEFT**;**

**}**

**else** **{**

direct **=** JoystickRIGHT**;**

**}**

**}**

**else** **{**

**if** **(**Y **>** 0**)** **{**

direct **=** JoystickUP**;**

**}**

**else** **{**

direct **=** JoystickDOWN**;**

**}**

**}**

//Report last direction if in dead zone.

**return** direct**;**

**}**

/\* INTERRUPT: BUTTON PRESSED \*/

void pressedButton**()** **{**

static unsigned long referencePress **=** 0**;**

**if** **(**millis**()** **-** referencePress **>=** 200**)** **{** // Added to reduce double presses

button **=** 1**;**

referencePress **=** millis**();**

**}**

**}**

/\*\* Menu Screens \*\*/

/\*Title Screen\*/

void title**()** **{**

// Print "snake" in multiple colors

tft**.**setTextSize**(**3**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 117**,** DISHEIGHT**/**2 **-** 25**);**

tft**.**setTextColor**(**ST77XX\_GREEN**);**

tft**.**print**(**"S"**);**

tft**.**setTextColor**(**ST77XX\_CYAN**);**

tft**.**print**(**"n"**);**

tft**.**setTextColor**(**ST77XX\_RED**);**

tft**.**print**(**"a"**);**

tft**.**setTextColor**(**ST77XX\_ORANGE**);**

tft**.**print**(**"k"**);**

tft**.**setTextColor**(**ST77XX\_MAGENTA**);**

tft**.**print**(**"e"**);**

// End of "snake nonsense; Print "Eternal"

tft**.**setTextColor**(**ST77XX\_WHITE**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 105**,** DISHEIGHT **/** 2**);**

tft**.**setTextSize**(**5**);**

tft**.**println**(**"ETERNAL"**);**

**while** **(true)** **{**

unsigned long current **=** millis**();**

static unsigned long referenceUpdate **=** current**;**

static bool toggle **=** 0**;**

// Every Second, flash the (PUSH2) message

**if** **(**current **-** referenceUpdate **>=** 1000**)** **{**

**switch** **(**toggle**)** **{**

**case** 0**:**

tft**.**setTextColor**(**ST77XX\_WHITE**);**

**break;**

**case** 1**:**

tft**.**setTextColor**(**ST77XX\_BLACK**);**

**break;**

**}**

tft**.**setTextSize**(**2**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 98**,** **(**DISHEIGHT**)+**5**);**

tft**.**print**(**"(PUSH2) to Play"**);**

toggle **=** **!**toggle**;**

referenceUpdate **=** millis**();**

**}**

**if** **(**button **==** 1**)** **{**

button **=** 0**;**

//Quickly erase title screen by only rewriting text; faster than "fillScreen()"

tft**.**setTextColor**(**ST77XX\_BLACK**);**

tft**.**setTextSize**(**3**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 117**,** DISHEIGHT **/** 2 **-** 25**);**

tft**.**print**(**"Snake"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 105**,** DISHEIGHT **/** 2**);**

tft**.**setTextSize**(**5**);**

tft**.**println**(**"ETERNAL"**);**

tft**.**setTextSize**(**2**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 98**,** **(**DISHEIGHT**)+**5**);**

tft**.**print**(**"(PUSH2) to Play"**);**

**return;**

**}**

**}**

**}**

/\* Continue Screen: Replay?\*/

uint8\_t playAgain**()** **{**

// Update Timers

static unsigned long referenceUpdate **=** millis**();**

static unsigned long referenceDir **=** millis**();**

static unsigned long referenceTick **=** millis**();**

unsigned long current **=** millis**();**

// Set Local variables

uint8\_t dir **=** JoystickNULL**;**

uint8\_t selected **=** 0**;**

uint8\_t timer **=** 9**;**

// Display menu over "Game Over" stuff

tft**.**setTextColor**(**ST77XX\_WHITE**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 100**,** **(**DISHEIGHT**)** **/** 2 **-** 56**,** 200**,** 28**,** ST77XX\_BLACK**);**

tft**.**setTextSize**(**3**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 98**,** **(**DISHEIGHT**)** **/** 2 **-** 54**);**

tft**.**println**(**"Play Again?"**);**

tft**.**setTextSize**(**2**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 10**,** **(**DISHEIGHT**)** **/** 2 **+** 80**,** 20**,** 20**,** ST77XX\_BLACK**);**

tft**.**setTextColor**(**ST77XX\_RED**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 5**,** **(**DISHEIGHT**)** **/** 2 **+** 83**);**

tft**.**print**(**timer**);**

tft**.**setTextColor**(**ST77XX\_WHITE**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 70**,** **(**DISHEIGHT**)** **/** 2 **+** 80**,** 50**,** 22**,** ST77XX\_BLACK**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **+** 20**,** **(**DISHEIGHT**)** **/** 2 **+** 80**,** 50**,** 22**,** ST77XX\_BLACK**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 55**,** **(**DISHEIGHT**)** **/** 2 **+** 83**);**

tft**.**print**(**"No"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **+** 27**,** **(**DISHEIGHT**)** **/** 2 **+** 83**);**

tft**.**print**(**"Yes"**);**

// Display just the outline of a rectangle 3 pixels thick; faster than fillRect twice.

**for** **(**uint8\_t i **=** 0**;** i **<** 3**;** i**++)** **{**

tft**.**drawRect**((**DISWIDTH**)** **/** 2 **-** 70 **+** i**,** **(**DISHEIGHT**)** **/** 2 **+** 80 **+** i**,** 50 **-** **(**2 **\*** i**),** 22 **-** **(**2 **\*** i**),**

ST77XX\_WHITE**);**

**}**

**while** **(true)** **{**

unsigned long current **=** millis**();**

// Update controller input every 10ms

**if** **(**current **-** referenceDir **>=** 10**)** **{**

dir **=** controller**(**Player1**);**

referenceDir **=** current**;**

**}**

// Update display every half-second based on controller input

**if** **(**current **-** referenceUpdate **>=** 500 **&&** dir **!=** JoystickNULL**)** **{**

**if** **(**dir **==** JoystickLEFT **&&** selected **!=** No**)** **{**

selected **-=** 1**;**

**}**

**else** **if** **(**dir **==** JoystickRIGHT **&&** selected **!=** Yes**)** **{**

selected **+=** 1**;**

**}**

//tft.fillRect((DISWIDTH) / 2 - 70, (DISHEIGHT) / 2 + 75, 140, 40, ST77XX\_BLACK);

**switch** **(**selected**)** **{**

**case** No**:** //Move border to "no"

**for** **(**uint8\_t i **=** 0**;** i **<** 3**;** i**++)** **{**

tft**.**drawRect**((**DISWIDTH**)** **/** 2 **-** 70**+**i**,** **(**DISHEIGHT**)** **/** 2 **+** 80**+**i**,** 50**-(**2**\***i**),**

22**-(**2**\***i**),** ST77XX\_WHITE**);**

tft**.**drawRect**((**DISWIDTH**)** **/** 2 **+** 20**+**i**,** **(**DISHEIGHT**)** **/** 2 **+** 80**+**i**,** 50**-(**2**\***i**),**

22**-(**2**\***i**),** ST77XX\_BLACK**);**

**}**

**break;**

**case** Yes**:** //Move border to "yes"

**for** **(**uint8\_t i **=** 0**;** i **<** 3**;** i**++)** **{**

tft**.**drawRect**((**DISWIDTH**)** **/** 2 **-** 70 **+** i**,** **(**DISHEIGHT**)** **/** 2 **+** 80 **+** i**,** 50 **-** **(**2**\***i**),**

22 **-** **(**2 **\*** i**),** ST77XX\_BLACK**);**

tft**.**drawRect**((**DISWIDTH**)** **/** 2 **+** 20 **+** i**,** **(**DISHEIGHT**)** **/** 2 **+** 80 **+** i**,** 50 **-** **(**2**\***i**),**

22 **-** **(**2 **\*** i**),** ST77XX\_WHITE**);**

**}**

**break;**

**}**

referenceUpdate **=** millis**();**

**}**

// Increment timer. automatically select "no" if time runs out. 9 "ticks"

**if** **(**current **-** referenceTick **>=** 1250**)** **{**

**if** **(**timer **==** 0**)** **{**

**return** 0**;**

**}**

**else** **{**

// Erase Previous Number, increment timer

tft**.**setTextColor**(**ST77XX\_BLACK**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 5**,** **(**DISHEIGHT**)** **/** 2 **+** 83**);**

tft**.**print**(**timer**--);**

// Print new timer number

tft**.**setTextColor**(**ST77XX\_RED**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 5**,** **(**DISHEIGHT**)** **/** 2 **+** 83**);**

tft**.**print**(**timer**);**

**}**

referenceTick **=** millis**();**

**}**

**if** **(**button **==** 1**)** **{**

button **=** 0**;**

**return** selected**;**

**}**

**}**

**}**

/\* Pause Screen \*/

void paused**()** **{**

// Update Timers

unsigned long referenceTime **=** 0**;**

unsigned long current**;**

enum pause **{** Displayed **=** 0**,** NotDisplayed **};**

bool toggle **=** Displayed**;** //Displayed or no?

**while** **(true)** **{**

current **=** millis**();**

**if** **(**current **-** referenceTime **>=** 1250**)** **{** //Toggle "paused" message every 1.25 seconds

**switch** **(**toggle**)** **{**

**case** Displayed**:**

//Displayed "Paused" in player 1 score position

tft**.**fillRect**(**0**,** 0**,** DISWIDTH **/** 2**,** 24**,** ST77XX\_ORANGE**);**

tft**.**setTextColor**(**ST77XX\_BLACK**);**

tft**.**setCursor**(**22**,** 4**);**

tft**.**setTextSize**(**2**);**

tft**.**print**(**"Paused"**);**

**break;**

**case** NotDisplayed**:**

snakePtrs**[**Player1**]->**PrintScore**();** //Re-print top left score (Player 1)

**break;**

**}**

toggle **=** **!**toggle**;**

referenceTime **=** current**;**

**}**

**if** **(**button **==** 1**)** **{** //If button is pressed again, reprint player 1's score and resume game

button **=** 0**;**

snakePtrs**[**Player1**]->**PrintScore**();**

**return;**

**}**

**}**

**}**

/\* Game Size Select \*/

uint8\_t gameScale**()** **{**

// Update Timers

static unsigned long referenceUpdate **=** 0**;**

static unsigned long referenceDir **=** 0**;**

const uint8\_t opt**[**3**]** **=** **{** 16**,** 8**,** 4 **};** //Scale options, 16, 8, or 4

uint8\_t dir **=** JoystickNULL**;**

enum diff **{** Normal **=** 0**,** Hard**,** Extreme **};**

uint8\_t selected **=** Normal**;**

//Draw menu on the screen

tft**.**setTextColor**(**ST77XX\_WHITE**);**

tft**.**setTextSize**(**3**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 105**,** 25**);**

tft**.**println**(**"Select Board"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 35**,** 50**);**

tft**.**println**(**"Size"**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 73**,** **(**DISHEIGHT**)** **/** 2 **+** 27**,** 146**,** 46**,** ST77XX\_WHITE**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 70**,** **(**DISHEIGHT**)** **/** 2 **+** 30**,** 140**,** 40**,** ST77XX\_BLACK**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 53**,** **(**DISHEIGHT**)** **/** 2**);**

tft**.**print**(**"Normal"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 63**,** **(**DISHEIGHT**)** **/** 2 **+** 38**);**

tft**.**print**(**"15 x 17"**);**

tft**.**setTextSize**(**2**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 98**,** **(**DISHEIGHT**)-**15**);**

tft**.**print**(**"Player 1 Selects"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 98**,** **(**DISHEIGHT**)+**5**);**

tft**.**print**(**"(PUSH2) to Enter"**);**

tft**.**setTextSize**(**3**);**

**while** **(true)** **{**

unsigned long current **=** millis**();**

// Update controller input every 10ms

**if** **(**current **-** referenceDir **>=** 10**)** **{**

dir **=** controller**(**0**);**

referenceDir **=** current**;**

**}**

// Change difficulty level every half second if contoller is moved.

**if** **(**current **-** referenceUpdate **>=** 500 **&&** dir **!=** JoystickNULL**)** **{**

**if** **(**dir **==** JoystickLEFT **&&** selected **!=** Normal**)** **{**

selected **-=** 1**;**

**}**

**else** **if** **(**dir **==** JoystickRIGHT **&&** selected **!=** Extreme**)** **{**

selected **+=** 1**;**

**}**

// Redraw diffculty box and text

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 70**,** **(**DISHEIGHT**)** **/** 2 **-** 17**,** 140**,** 40**,** ST77XX\_BLACK**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 70**,** **(**DISHEIGHT**)** **/** 2 **+** 30**,** 140**,** 40**,** ST77XX\_BLACK**);**

**switch** **(**selected**)** **{**

**case** Normal**:**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 53**,** **(**DISHEIGHT**)** **/** 2**);**

tft**.**print**(**"Normal"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 63**,** **(**DISHEIGHT**)** **/** 2 **+** 38**);**

tft**.**print**(**"15 x 17"**);**

**break;**

**case** Hard**:**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 43**,** **(**DISHEIGHT**)** **/** 2**);**

tft**.**print**(**"Hard"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 63**,** **(**DISHEIGHT**)** **/** 2 **+** 38**);**

tft**.**print**(**"30 x 34"**);**

**break;**

**case** Extreme**:**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 62**,** **(**DISHEIGHT**)** **/** 2**);**

tft**.**print**(**"Extreme"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 63**,** **(**DISHEIGHT**)** **/** 2 **+** 38**);**

tft**.**print**(**"60 x 68"**);**

**break;**

**}**

referenceUpdate **=** millis**();**

**}**

// Break and return when button is pressed

**if** **(**button **==** 1**)** **{**

button **=** 0**;**

// Erase Screen by overwriting text, faster than fillscreen

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 73**,** **(**DISHEIGHT**)** **/** 2 **+** 27**,** 146**,** 46**,** ST77XX\_BLACK**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 70**,** **(**DISHEIGHT**)** **/** 2 **-** 17**,** 140**,** 40**,** ST77XX\_BLACK**);**

tft**.**setTextColor**(**ST77XX\_BLACK**);**

tft**.**setTextSize**(**3**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 105**,** 25**);**

tft**.**println**(**"Select Board"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 35**,** 50**);**

tft**.**println**(**"Size"**);**

tft**.**setTextSize**(**2**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 98**,** **(**DISHEIGHT**)-**15**);**

tft**.**print**(**"Player 1 Selects"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 98**,** **(**DISHEIGHT**)+**5**);**

tft**.**print**(**"(PUSH2) to Enter"**);**

**return** opt**[**selected**];**

**}**

**}**

**}**

/\* Game player count Select \*/

uint8\_t playerCount**()** **{**

static unsigned long referenceUpdate **=** 0**;**

static unsigned long referenceDir **=** 0**;**

uint8\_t dir **=** JoystickNULL**;**

uint8\_t selected **=** 0**;**

//Draw menu on the screen

tft**.**setTextColor**(**ST77XX\_WHITE**);**

tft**.**setTextSize**(**3**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 117**,** 25**);**

tft**.**println**(**"Select Player"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 45**,** 50**);**

tft**.**println**(**"Count"**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 23**,** **(**DISHEIGHT**)** **/** 2 **+** 27**,** 46**,** 46**,** ST77XX\_WHITE**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 20**,** **(**DISHEIGHT**)** **/** 2 **+** 30**,** 40**,** 40**,** ST77XX\_BLACK**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 7**,** **(**DISHEIGHT**)** **/** 2 **+** 38**);**

tft**.**print**(**1**);**

tft**.**setTextSize**(**2**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 98**,** **(**DISHEIGHT**)-**15**);**

tft**.**print**(**"Player 1 Selects"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 98**,** **(**DISHEIGHT**)+**5**);**

tft**.**print**(**"(PUSH2) to Enter"**);**

tft**.**setTextSize**(**3**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 7**,** **(**DISHEIGHT**)** **/** 2 **+** 63**);**

**while** **(true)** **{**

unsigned long current **=** millis**();**

// Update controller input every 10ms

**if** **(**current **-** referenceDir **>=** 10**)** **{**

dir **=** controller**(**0**);**

referenceDir **=** current**;**

**}**

// Update player count with half second delays.

**if** **(**current **-** referenceUpdate **>=** 500 **&&** dir **!=** JoystickNULL**)** **{**

// Erase Current number

tft**.**setTextColor**(**ST77XX\_BLACK**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 7**,** **(**DISHEIGHT**)** **/** 2 **+** 38**);**

tft**.**print**(**selected **+** 1**);**

// Update Number

**if** **(**dir **==** 1 **&&** selected **!=** Player1**)** **{**

selected **-=** 1**;**

**}**

**else** **if** **(**dir **==** 3 **&&** selected **!=** Player4**)** **{**

selected **+=** 1**;**

**}**

// Redraw text

tft**.**setTextColor**(**ST77XX\_WHITE**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 7**,** **(**DISHEIGHT**)** **/** 2 **+** 38**);**

tft**.**print**(**selected **+** 1**);**

referenceUpdate **=** millis**();**

**}**

// Break and return when button is pressed

**if** **(**button **==** 1**)** **{**

button **=** 0**;**

// Erase screen by directly overwriting text: Faster than "fillscreen"

tft**.**setTextColor**(**ST77XX\_BLACK**);**

tft**.**setTextSize**(**3**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 117**,** 25**);**

tft**.**println**(**"Select Player"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 45**,** 50**);**

tft**.**println**(**"Count"**);**

tft**.**fillRect**((**DISWIDTH**)** **/** 2 **-** 23**,** **(**DISHEIGHT**)** **/** 2 **+** 27**,** 46**,** 46**,** ST77XX\_BLACK**);**

tft**.**setTextSize**(**2**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 98**,** **(**DISHEIGHT**)-**15**);**

tft**.**print**(**"Player 1 Selects"**);**

tft**.**setCursor**((**DISWIDTH**)** **/** 2 **-** 98**,** **(**DISHEIGHT**)+**5**);**

tft**.**print**(**"(PUSH2) to Enter"**);**

**return** **(**selected**+**1**);**

**}**

**}**

**}**