CRT 360: Advanced Prototyping

GamePad - Game Setup

Part I: Getting Started

1. Graphic Resources

Open Teensyduino and create a new sketch and save it as [FirstName]_GamePad. Create an appropriate header comment with information about your game. For example:

```
// HackBerry Attack!
// Zane Cochran || 01 Jan 2021
// CRT 360 Advanced Prototyping || GamePad
```

Export all pixel sprites from Piskel by going to Export -> Others -> Download C File. Make sure your scale is set to 1.0x to ensure proper sizing.

Then use the Piskel2LCD application to convert each Piskel file into an Arduino-friendly .h files. Each C file from Piskel will result in two Arduino files - an XXX_MASK.h and an XXX_PIX.h file. The MASK file contains all the transparency information for the images whereas the PIX file contains all the color information. The Arduino will use both of these files to draw the graphics you created.

Move these files into the folder created by Arduino when you saved your new sketch.

2. Frame Rate Sampling

Begin a Serial connection (9600 baud rate) in the setup( ) function.

Create a new tab called *frame.h* and include the Metro library. Create a new timer using the following format (replace *timerName* with a descriptive name for your timer and *timerDuration* for the number of milliseconds to delay):

```
#include <Metro.h>                          // This adds the Metro library to your sketch
Metro timerName = Metro(timerDuration);     // A prototype for a Metro timer
```

Create a function called checkFrame that will print how many times the Arduino has gone around the loop every 1 second. You can use the .check( ) function in Metro like this:

```
if(timerName.check( ) == true){
        // do something
}
```

Or even simpler - since the .check( ) function returns a true or false value, it is not necessary to include the == true part of the if statement:

```
if(timerName.check( )){
        // do something
}
```

void checkFrame( ) Copy/Paste your function in the space below

void checkFrame() {
  frames++;
  if (sampleTimer.check()) {
  Serial.println(frames);
  frames = 0;
  }
}

Include the frame.h file in the header of your program's main tab and call the checkFrame function in the loop( ). Record an example of the types of frame rates you observe below:

Observed Frame Rate
11985713 13635640 13635641

3. Screen Setup:

Connect your Teensy to your LCD screen using the following configuration:

Teensy / ILI9341 LCD Screen Pinouts			
ILI9341 Pin	**Teensy 4.0 Pin**	**ILI9341 Pin**	**Teensy 4.0 Pin**
VCC	VIN	GND	GND
CS	10	RESET	VCC
D/C	9	SDI (MOSI)	11
SCK	13	LED	VIN (w/ 100Ω Resistor)
SDO (MISO)	12	Source: https://www.pjrc.com/store/display_ili9341_touch.html	

Create a screen.h tab and include the SPI and ILI9341 libraries, screen pin definitions, and create the screen object in the header of the tab:

```
#include "SPI.h"
#include "ILI9341_t3n.h"

#define TFT_DC  9
#define TFT_CS  10
```

```
#define TFT_RST 8

ILI9341_t3n tft = ILI9341_t3n(TFT_CS, TFT_DC, TFT_RST);
```

Also define the screen width and height and initialize the screen buffer in the header as well:

```
#define screenW  320
#define screenH  240
DMAMEM uint16_t screenBuffer[screenW * screenH];          // Screen Buffer
```

Create the function void initScreen( ) to begin communicating with the screen. Connect to the LCD and rotate the screen using the .setRotation( ) command. Initialize and activate the frameBuffer and fill the screen with black:

```
tft.begin();                          // Connect to LCD Screen
tft.setRotation(1);                   // Rotate Screen 90 Degrees

tft.setFrameBuffer(screenBuffer);     // Initialize Frame Buffer
tft.useFrameBuffer(1);                // Use Frame Buffer

tft.fillScreen(ILI9341_BLACK);        // Clear Screen
```

Create another function called screenTest( ). This function should change the entire screen from black to white over and over again every 2 seconds. Use the Metro library, a boolean, and the resources from the [Adafruit GFX Documentation] to help you.

Include the screen.h file on the program's main tab and call the initScreen( ) function in the main program's setup( ). Call the screenTest( ) function in the loop( ) and test. After testing, remove the screenTest( ) from the loop.

void screenTest( ) Copy/Paste your function and any helper variables.

```
Metro screenTestTimer = Metro(2000);
boolean flip;

void screenTest(){
 if(screenTestTimer.check()){
   flip = !flip;
 }

 if(flip){
   tft.fillScreen(ILI9341_BLACK);
 } else {
   tft.fillScreen(ILI9341_WHITE);
 }
}
```

4. Drawing Level Tiles

Create a new tab called *tile.h* and include all the pixel sprite .h files you added in Step 2 into its header. For example:

```
#include "tiles_MASK.h"
#include "tiles_PIX.h"
```

Define some parameters for the level sprite tiles. For example:

```
#define tileW 16        // 16 Tiles Across
#define tileH 12        // 12 Tiles Down
#define tileSize 20     // Tile Width (in pixels)
#define numLevels 6     // Number of Levels
```

Create a 2-dimensional array called levels to store each level's tile layout. Each value in the array will correspond to a specific bitmap tile you created in Piskel. You can reference the tiles using their hex values you recorded in the previous assignment. If you have some parts of the level that will not have a tile, indicate this by using the hex value 0xFF. Use your level maps and game resources from the concept development assignment to help you. For example:

```
int levels[numLevels][tileW * tileH]{
 // Level 0 - Front
 {
  0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01,
  0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
  0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x1C, 0x1D, 0x00, 0x00, 0x1E, 0x1F, 0x00, 0x00, 0x00,
  0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xFF, 0xFF, 0xFF, 0xFF, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xFF, 0xFF, 0xFF, 0xFF, 0x0F, 0x04, 0x04, 0x04, 0x04, 0x05, 0x05, 0x04, 0x04, 0x04, 0x04, 0x0C,
 },
 // Level 1 - Design Studio
 {
  0xFF, 0xFF, 0xFF, 0xFF, 0x0F, 0x04, 0x04, 0x04, 0x04, 0x05, 0x05, 0x04, 0x04, 0x04, 0x04, 0x0C,
  0xFF, 0xFF, 0xFF, 0xFF, 0x0A, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x06,
  0xFF, 0xFF, 0xFF, 0xFF, 0x0A, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x06,
  0xFF, 0xFF, 0xFF, 0xFF, 0x0A, 0x03, 0x03, 0x17, 0x14, 0x03, 0x03, 0x17, 0x14, 0x03, 0x03, 0x06,
  0xFF, 0xFF, 0xFF, 0xFF, 0x0A, 0x03, 0x03, 0x16, 0x15, 0x03, 0x03, 0x16, 0x15, 0x03, 0x03, 0x06,
  0xFF, 0xFF, 0xFF, 0xFF, 0x0A, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x07,
  0xFF, 0xFF, 0xFF, 0xFF, 0x0A, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x07,
  0xFF, 0xFF, 0xFF, 0xFF, 0x0A, 0x03, 0x03, 0x17, 0x14, 0x03, 0x03, 0x17, 0x14, 0x03, 0x03, 0x06,
  0xFF, 0xFF, 0xFF, 0xFF, 0x0A, 0x03, 0x03, 0x16, 0x15, 0x03, 0x03, 0x16, 0x15, 0x03, 0x03, 0x06,
  0xFF, 0xFF, 0xFF, 0xFF, 0x0A, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x06,
  0xFF, 0xFF, 0xFF, 0xFF, 0x0A, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x21, 0x06,
  0xFF, 0xFF, 0xFF, 0xFF, 0x0E, 0x08, 0x08, 0x08, 0x08, 0x09, 0x09, 0x08, 0x08, 0x08, 0x08, 0x0D,
 },
 // Etc..
};
```

Write a function called void drawLevel(int thisLevel) that takes an input of which level should be drawn and store each tile at the appropriate location in the screen buffer using the tft.drawRGBBitmap( ) function.

Test the drawLevel( ) function by calling it in the main loop sampling each of the levels you designed. You will need to call the tft.updateScreen( ) function to send the screen buffer to the LCD screen. Also, open the Serial monitor and make a note of the frame rate. Remove these functions from the main loop when you are done testing.

void drawLevel( ) Copy/Paste your function in the space below

```
void drawLevel(int thisLevel) {
 for (int y = 0; y < tileH; y++) {
  for (int x = 0; x < tileW; x++) {
    int index = x + (y * tileW);
    int whichTile = levels[thisLevel][index];

    int finalX = x * tileSize;
    int finalY = y * tileSize;
    tft.drawRGBBitmap(finalX, finalY, backgroundTiles_PIX[whichTile], tileSize, tileSize);
  }
 }
}
```