ECE177: Programming I: From C...

Lab #7 — Bouncing Square

Week of 30 March 2026

1 Objectives

- Complete all of the steps outlined below while working on `bounce.c`
- Program the Pico breadboard with the modified `bounce.c` file and confirm working output

2 Materials

- Assembled and working Pico breadboard
- USB cable
- Laptop and Laptop charger

3 Procedure

1. Make sure to read all of the directions carefully before getting started.
2. Download the `lab07_code.zip` file here:
   `https://web.eece.maine.edu/~vweaver/classes/ece177/labs/lab07_code.zip`
3. Extract the zip file in your `Documents/ece177/labs/` directory that you created in lab01.
4. In VS Code, use the Raspberry Pi Pico Extension to import the `lab07_code` folder as a project.

4 Modify the Provided Code

1. Edit the provided `bounce.c` file
2. This document describes the tasks you need to do. There are also code comments in the `bounce.c` file but they might not be as detailed as the ones in this document.
3. Make sure to read all of the directions carefully before getting started so as to not miss any of the instructions.
4. Be sure to comment your code!
5. Test your code often! Don't try to write it all at once!
6. As with previous labs ideally you can just press the VS Code "Run" button and it will compile and upload your program to the Pico. If it doesn't you might have to find the `bounce.uf2` file in the `build` subdirectory and copy it over to the Pi Pico virtual drive.

Section A: Update the Code Comments

(a) Update the code comments at the top of the file with your name, the date, and a brief description of the lab.

Section B : Init Sound Code

(a) Add code that calls `init_pwm()`

Section C: Setup initial conditions

(a) Setup a struct to hold the ball's state. The struct type should be `ball_state` and declare a `struct ball_state` named `ball`. This struct should contain elements `x`, `y`, `xadd`, and `yadd` that are all signed integers, and `color` that's a short integer

(b) Clear the screen to black using the function
`Adafruit_ST7735_fillScreen(int color);`
Remember, the color black is a 16-bit value that's all zero bits.

(c) Initialize the `ball` structure with the following settings:

i. `x` should be the middle of the screen (make it half the value of the provided `XMAX`)

ii. `y` should be the middle of the screen (make it half the value of the provided `YMAX`)

iii. `xadd` and `yadd` should start out as 1

iv. Color should start out as white (this is a 16-bit value that's all 1 bits)

Section D: Test Ball Drawing

(a) Create an infinite loop
Inside this loop do the following things:

(b) Use `Adafruit_ST7735_fillRect(x,y,xsize,ysize,color);` to draw a rectangle at `ball.x`, `ball.y` of tXSIZE, `YSIZE` and `ball.color`

(c) At the end of the loop put `jump_to_MSD();` so that pressing `*+D` will re-enter BOOTSEL mode

(d) Compile and run your code to make sure it clears the screen and puts a white rectangle in the middle. If not, debug your code until it does

Section E: Move Ball Horizontally

(a) After the draw code, delay a proper amount of time for a 60fps frame rate. Use `sleep_ms()` for this delay.
We discussed in class, for example to have a 50fps rate you'd take 1/50 which is 0.020s, or 20ms. Calculate what 60fps would be and use it.

(b) After the delay, erase the ball using `Adafruit_ST7735_fillRect()` as from before, but making the color black.

(c) Add code to move the ball back and forth horizontally by taking `ball.x` and adding ball.xadd to it

(d) Add collision detection. If `ball.x` is less than 0, then negate `ball.xadd` so it moves the other way

(e) Likewise if `ball.x` is greater than `XMAX-XSIZE` flip the direction

(f) Test your code and make sure the ball moves properly

Section F: Move Ball Vertically

(a) Add code to move the ball back and forth vertically by duplicating your code, but make sure it modifies `ball.y` instead and uses `YMAX-YSIZE` for the range

(b) Test your code and make sure the ball moves properly

Section G: Play Sound on Wall Collision

(a) Now create a function called `play_tone()` after the end of the main function. It should take one argument, `length`, that is the time in ms to play

(b) This function should do the following:
`pwm_set_enabled(slice_number,true);`
sleep by the number of ms specified in length
`pwm_set_enabled(slice_number,false);`

(c) Add calls to this function so that it gets called

Section H: Change Color/Size of the Ball

(a) Change the color of the ball to be a color of your choice other than white

(b) You can find HEX color pickers online that will tell you the HEX rgb values for any color you like

(c) You can try to convert these to the 565 color scheme as described in class manually

(d) Alternately you can use the function

```
uint16_t Adafruit_ST7735_Color565(uint8_t r, uint8_t g, uint8_t b)
```

to convert it for you

(e) Also double the size of the ball by finding the pre-processor define for `XSIZE` and `YSIZE` and changing them to 16 instead of 8.
This is the point of using pre-processor defines, if you used them consistently throughout then changing it in this one place should be all that's needed and you don't have to find hard-coded constants everywhere and change them.

5 Grading/Checkoff

1. Upload your `bounce.c` file to BrightSpace.
2. Upload a picture of your board running and showing the bounce code running.
3. TAs will check the output of your final program running on your hardware. They will ask you to demonstrate the lab requirements and will enter grades after the requested information has been uploaded on the assignment rubric.
4. TAs: Enter grades only after verifying full functionality of the code running, the requested information has been uploaded, and the assignment has been submitted.