ECE 177 – Programming I: From C Foundations to Hardware Interaction Lecture 26

Vince Weaver

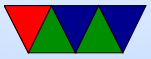
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

6 April 2026

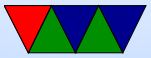
Announcements

- Lab#8 this week
- HW#6 due
- Working on getting HW#7 posted
- Reminder midterm the 10th
- We will review for midterm#2 in class on Wednesday
- There will be an ECE review for midterm#2 (with pizza) Barrows Hall 130, Wednesday, April 8th from 5-7 pm



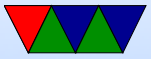
Lab8 – Keypad Movement

- This lab we will re-use your graphics code from Lab#7 and combine it with the keypad code from Lab#6
- Note: because of this be sure you follow the directions in the lab handout, there won't be any comments in the code about this because you'll be re-using Lab#7



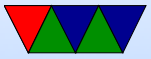
Lab8 – Copying Your Code Over

- You'll need to copy your `bounce.c` file over from Lab#7 and have it over-write `paddle.c`
- How do you do that?
- Can just copy with the GUI on windows / mac
- If you're on Linux at the command line can use `cp`
- Be careful on Windows, you might end up with a file called `paddle.c.c` and vs code won't be able to find it. Make sure it only has one `.c`



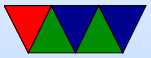
Lab8 – Adding a Life Count

- Add a variable to hold it
- If go off bottom of screen, lose a life
- How tell off bottom? Check if `ball.y` is greater than `YMAX - YSIZE` (you already had code for this in Lab#7 to bounce off the bottom)



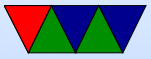
Lab8 – Off Bottom of Screen, 0 Lives

- Check ball count. If zero do the following
- Print "Game Over" using `printf()`
- Exit the game
  - Calling `return 0;` might be your best bet
  - Can also maybe call `exit(0);`
  - Pi Pico has no operating system, so if you exit like this it will just hang forever



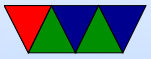
Lab8 – Off Bottom of Screen, Some Lives

- If have some lives left, do the following
- Decrement the ball count
- Update ball count on screen (see followup slide)
- Reset `ball.x` and `ball.y`
  - Put `ball.y` center of screen, YMAX divided by two
  - Pick `ball.x` randomly (see later slide)
  - Set `ball.yadd` to minus 1.
  - TODO: should we make ball.xadd random?

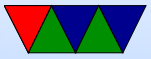


Lab8 – Random Numbers

- Random: will need to include `stdlib.h`
- Can use the call `rand()`
- Returns int from 0 to MAXINT. How can we make it between 0 and `XMAX`? Modulus / remainder `%`
- It's a pseudo-random number generator, but good enough for a game
- You can use `srand()` to seed it to make it more random but you'd have to find some Pi Pico source of randomness (often you use time, but Pico doesn't have a RTC? Get

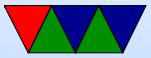


it from timing keypad presses?)



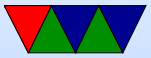
Lab8 – Random Numbers Aside

- Why use `rand()` and not `random()`
- Both in C library
- In old days `random()` often had better randomness
- Modern Linux they both use the same code?



Lab8 – Writing Life Count

- Create an `update_lives()` function
- Predeclare it with a prototype at top, put code at bottom
- We don't want to use `printf()` as this clears the screen
- Create a string that contains "LIFE: X" where instead of X you have the number of lives

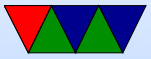


Lab8 – Creating Custom String with sprintf()

- Can create custom string with strcpy() and such
- Better way is to use `sprintf()`. It's like printf but prints to a string. Something like:

```
char string[100];
sprintf(string,"LIFE: %d",lives);
```

- Note to use `sprintf()` you might need to include `string.h`



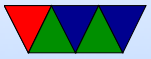
Lab8 – Writing Text Direct to Screen

- Use the function

```
size_t graphics_drawText(char *s,
                         uint16_t x, uint16_t y);
```

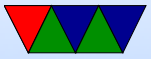
  to draw this string at location 0,0
- Note: that's just a function prototype. When using it take out the variable types and substitute your parameters, something like

```
graphics_drawText(my_string, 100,100);
```



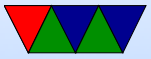
Lab8 – Erase Text before Drawing It

- By default the `drawText` routine draws transparent text, meaning it lets the background show through if not part of a letter
- This is a problem if we try to write our text on top of existing text as the previous number bleeds through
- Use the draw rectangle routine to over-write the area in black before writing to clear that out



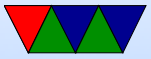
Lab8 – Calling update_lives()

- Be sure to call `update_lives()` after you decrement the life count
- Also be sure to call it once before the main game `while()` loop to print lives left at start of game



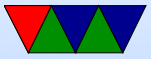
Lab8 – Aside on How Fonts Work

- If we have time maybe talk about that



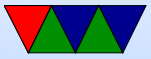
Lab8 – Initializing the Paddle

- Create a new struct type paddle with fields `x`, `y`, `xsize`, `ysize`, and `color`
- For now color white
- x is halfway across screen, y is `PADDLE_Y`
- You should `#define PADDLE_Y 300` near the top of the code
- paddle xsize 64, ysize 10 (arbitrary)
- Should we have co-ordinates on left edge of paddle or center? Just different math to handle that

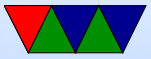


Lab8 – Drawing the Paddle

- Should we only draw if paddle moved?
  Would be less flicker, but more complex
- Use `Adafruit_ST7735_fillRect()` to draw the paddle
- Put the code just before the end of the `while()` loop
- We will actually do two things
  - Erase the old paddle by calling fillRect with `paddle.x`, `paddle.y`, `paddle.xsize`, `paddle.ysize` and the color 0 (black)
  - Draw the new paddle by calling fillRect again but with

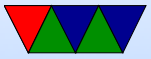


the color being `paddle.color`



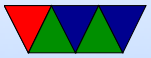
Lab8 – Collision Detecting the Paddle

- Check if `ball.y+YSIZE` is equal to `paddle.y`
- If that's true, then check if `ball.x` is greater than `paddle.x`
- If that's true, then check if `ball.x` is less than `paddle.x` plus `paddle.xsize`
- You can do this with nested ifs, or you can use one big statement like `if ( (x) && (y) && (z) )`
- If these all true, hit paddle, so flip direction and play sound just like if hitting wall



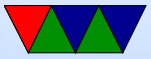
Lab8 – Advanced Paddle Handling

- Ideally you'd check even more things here
  - What if it somehow hits the left / right side of the paddle?
  - What if you move the paddle so it's inside the ball?
  - Should you make it maybe go faster or add spin if you hit a glancing blow?
  - What if ball gets stuck?



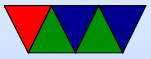
Lab8 – Adding in the Keypad Code

- First, remove the `jump_to_MSD()` function from the end of the while loop



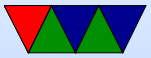
Lab8 – Create init_keypad()

- Next create an `init_keypad()` function at the end of the file, and add a prototype to the top
- For this we will cut and paste code from Lab#6
- Copy into it the keypad GPIO init code from Lab#6
- Make sure this `init_keypad()` function is called at the start of your main() function



Lab8 – Create read_keypad()

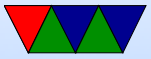
- Create a `read_keypad()` function at the end of the file, and add a prototype to the top
- This should return an int but take no arguments
- Copy into it the code from Lab#6 Section D part B.
- In that Lab this was inside a while loop, but in our case we don't want the while, we just want the code that does the scan so each time we call `read_keypad()` it does one scan
- At the start of the function have an int called `result`



which you initialize to -1 (meaning no key pressed)

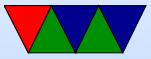
- In your code, instead or printf() row/column like Lab6, instead generate a value in result that is `(row<<4)|col`
- At the top add the following defines that will map this result to KEYPAD values

```
/* row/column */
#define KEYPAD_1 ((3<<4)|3)
#define KEYPAD_A ((3<<4)|0)
#define KEYPAD_C ((1<<4)|3)
#define KEYPAD_D ((0<<4)|3)
```



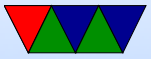
Lab8 – Paddle Left/Right

- Add a switch/case between the paddle erase and paddle draw calls
- Case `KEYPAD_1`, paddle left. In this case check if `paddle.x` is greater than 0. If so, decrement `paddle.x`
- Case`KEYPAD_A`, paddle right. In this case check if `paddle.x` plus `paddle.xsize` is less than `XMAX`. If so, increment `paddle.x`



Lab8 – Disabling / Enabling Sound

- Create a variable `sound_enabled`. Global. Starts as 1.
- If C pressed, turn off
- If 7 pressed, turn on
- Originally had you toggle it if C pressed, but we read the keypad so often that it would just toggle it many times when key pressed

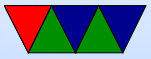


Lab8 – Entering Program Mode if 'D' Pressed

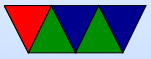
- This just calls the code used by the MSD function provided by Sheaff

```
printf("Reboot to MSD\n");
sleep_ms(2000);
reset_usb_boot(0,0);
```

- Why not *D like before? Tricky to return two keys pressed?
- One way to do that is have a bitmap. 16-bit value with currently all that are being held down. We couldn't use



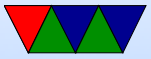
a case/switch in this case anymore, we'd have to use masking



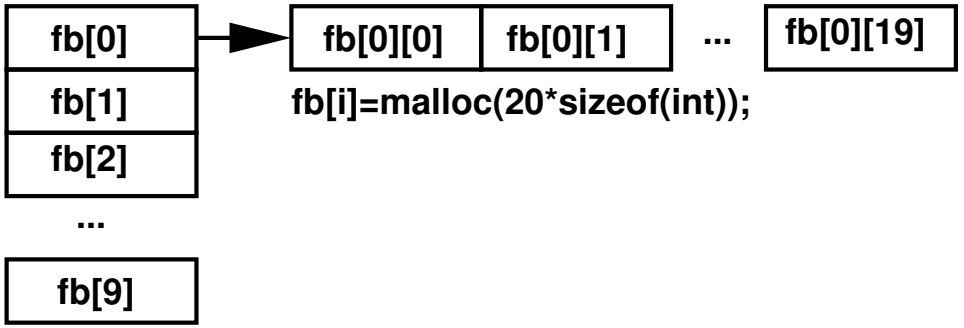
Multidimensional Arrays with Malloc

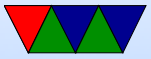
- Can you create multi-dimensional arrays with `malloc()`?
- Yes but it is a huge pain and not recommended by me at least
- `int fb[10][20];` is sort of intuitive and easy to access

```
int **fb;
// pointers to pointers
fb=malloc(10*sizeof(int *));
for(i=0;i<10;i++) {
   fb[i]=malloc(20*sizeof(int));
}
and accessing not that great either, lookup
```



```
int **fb;
fb=malloc(10*sizeof(int *));
```





Using 1-dimensional instead

- Can just have

```
int *fb=malloc(10*20*sizeof(int))
```

- This will give same layout as an `int fb[10][20];`
- To access it though you'd have to do `fb[(y*xsize)+x];` which is a bit of a pain
- You can have a macro to make this easier