# LAB 01:
# Drawing Shapes

## Provided Files

- lab01.c

## Files to Add from Lab00

- Makefile
- .vscode
  - tasks.json

## Files to Edit

- lab01.c

---

## Instructions

Note: Make sure to copy over your Makefile and .vscode folder (including your tasks.json file) from Lab00.

In this lab, you will be completing TODOs to finish writing three functions:

**TODO 1.0-1.1**

```
void setPixel(int col, int row, unsigned short color)
```

- Complete this function that fills the pixel at the specified col and row with the specified color.
- Call the function in main to make sure that it works!

**TODO 2.0-2.2**

```
void drawRect(int col, int row, unsigned short color)
```

- Create two macros, `RECT_WIDTH` and `RECT_HEIGHT` to represent the width and height of your rectangles, respectively.
- Complete this function to draw a rectangle at the specified col and row.
  - Draws a rectangle at the specified column and row.
  - Must be filled in (i.e. a solid rectangle, not just lines).
  - May be any color.
  - May be any (GBA appropriate) size.
  - May NOT be hardcoded. You may not just call setPixel for every colored-in dot on your rectangle. You must use iteration to do so!

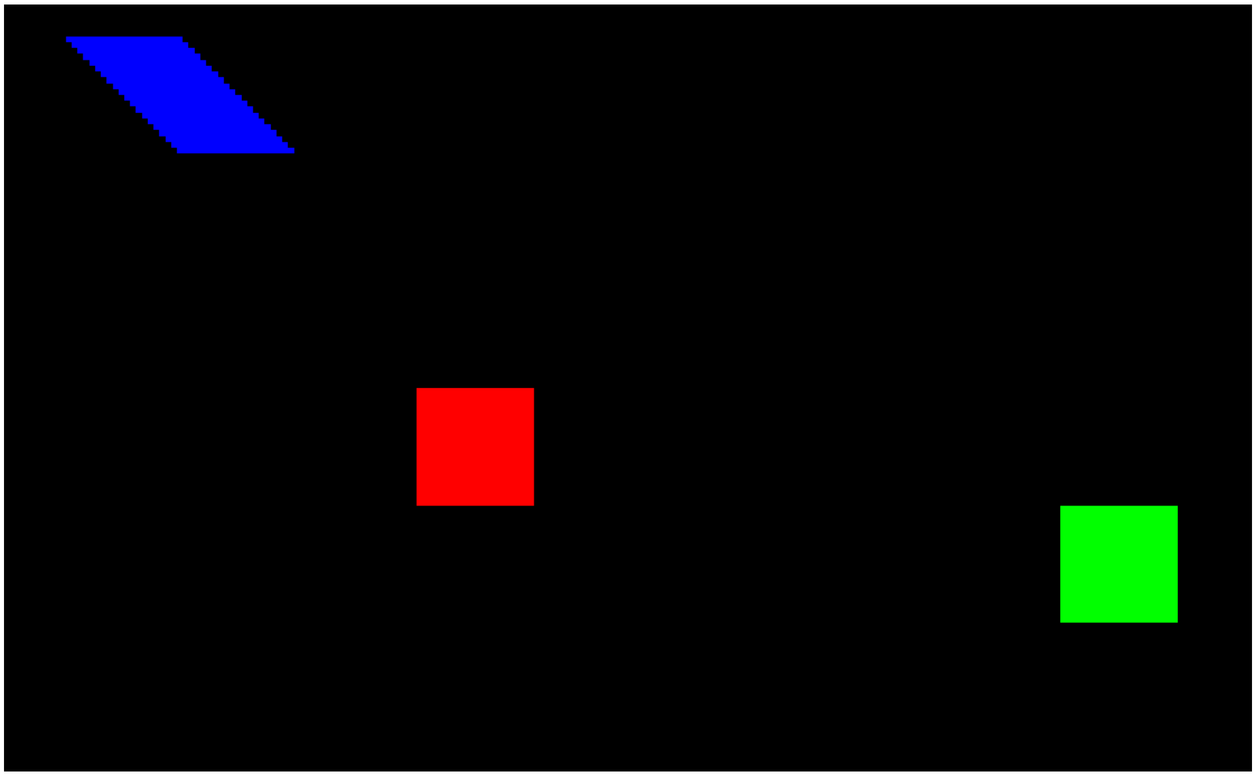
        ■ Must use your defined macros for the width and height

## TODO 3.0-3.2

`void drawParallelogram(int col, int row, unsigned short color)`

- Draws a parallelogram at the specified column and row.
- May be any color, either filled in or not (your choice).
- May be any (GBA appropriate) size.
- May NOT be hardcoded. You may not just call setPixel individually for every pixel. You must use **iteration** to do so!
- Hint: Think of a parallelogram as a series of stacked lines with each one shifted slightly left or right compared to the one above.

After writing the two functions, **call the drawRectangle function two times and call the drawParallelogram function one time in your main function,** with different parameters. The output of your program should look something like the following.



Your parallelogram and your rectangle may be any size, shape, color, rotation, or position you want, as long as we can see that you drew two filled in rectangles and one parallelogram. You may also create helper functions if you wish. After you have completed all the TODOs and your output looks correct, you are finished with the lab after you submit it.

## Tips

- For-loops are really great for drawing lines.
- The GBA screen size is 240 pixels wide and 160 pixels tall.

---

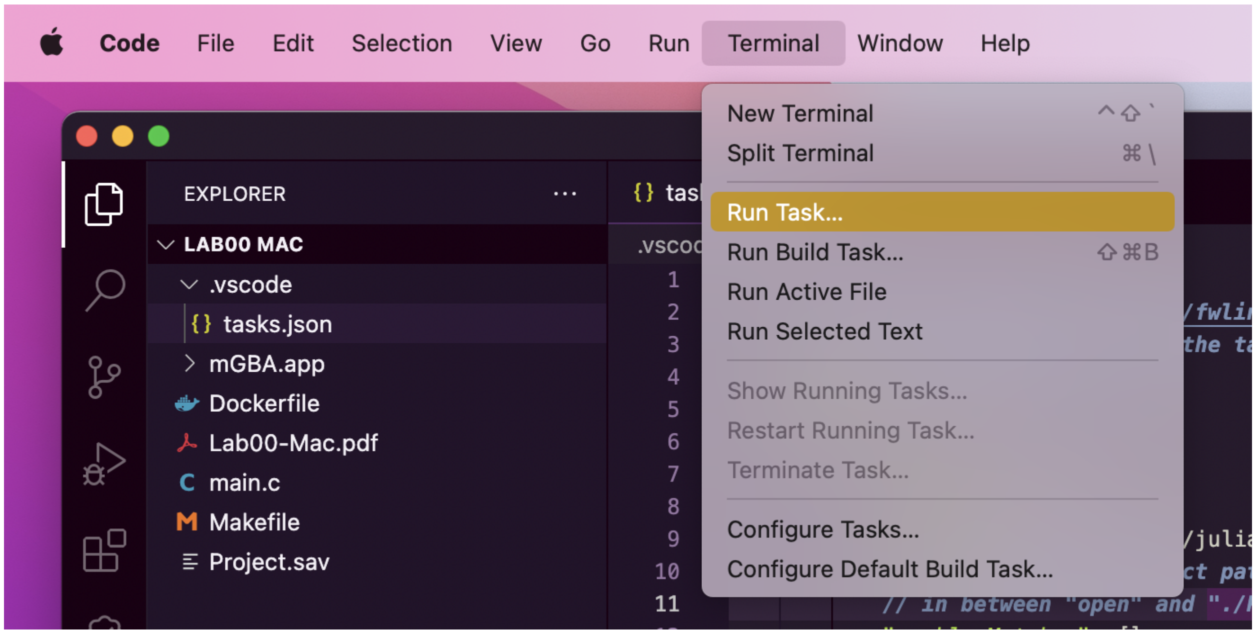
## Submission Instructions

Ensure that cleaning and building/running your project still gives the expected results. Please reference the last page of this document for instructions on how to perform a "clean" command.

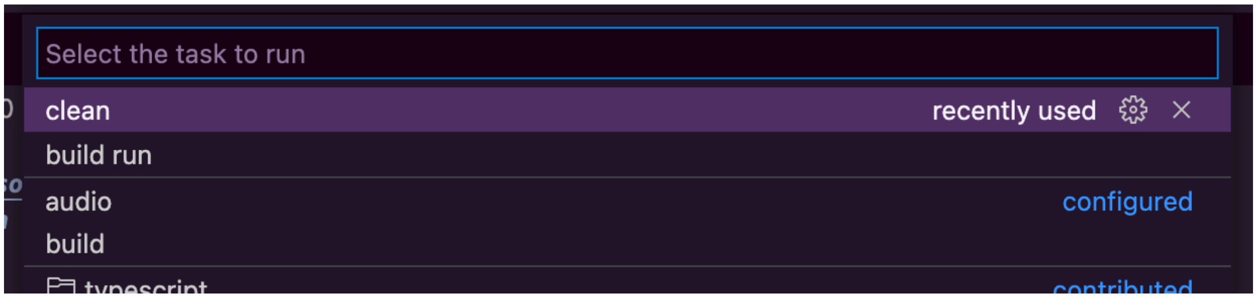
Zip up your entire project folder, including all source files, the Makefile, and everything produced during compilation **(including the .gba file)**. Submit this zip on Canvas. Name your submission Lab01_FirstameLastname, for example: "Lab01_GeorgeBurdell.zip". It is your responsibility to ensure that all the appropriate files have been submitted, and that your submitted zip can be opened and everything cleans, builds, and runs as expected.

## How to Clean

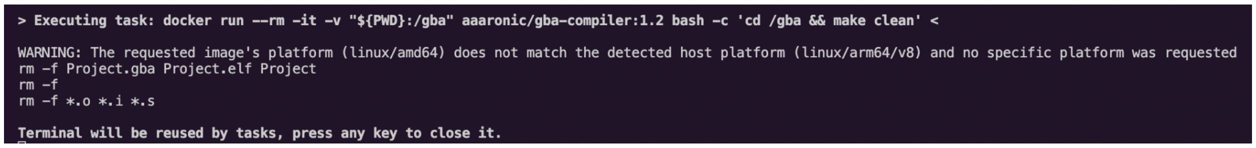
Navigate to the **Terminal** option at the top of your screen. Click on it, and then select **Run Task…**, as shown in the image below.



A dropdown menu will appear with the possible tasks to perform. Next, select **clean** from the options. Note that yours might be in a different order from mine, but the task should be there.



After selecting clean, something similar to the following should appear in your terminal tab (bottom of the screen).



```
> Executing task: docker run --rm -it -v "${PWD}:/gba" aaaronic/gba-compiler:1.2 bash -c 'cd /gba && make clean' <

WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
rm -f Project.gba Project.elf Project
rm -f
rm -f *.o *.i *.s

Terminal will be reused by tasks, press any key to close it.
```

If so, success! You have cleaned. You can now do **cmd/ctrl + shift + B** to build run (or click **Terminal**, then **Run Build Task…** or **Run Task…** and then **build run**).