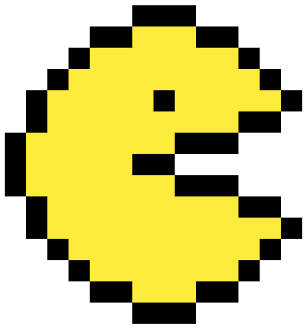
HOMEWORK 03:
Extended Mode 3 Game



Purpose: To build a more complex game in Mode 3 to further your understanding of: structs, arrays, and pooling.

Instructions

In this homework, you will be making a more complex game in Mode 3. **This must be something different than what you implemented for HW02 or any of the labs.** Examples of games you can use for inspiration are at the bottom of this PDF.

The design of this game must be more complex than "catching/dodging falling boxes" or basic "pong". We are leaving this one more open ended so that you can push yourself creatively and see what fun game you can come up with! You are free to choose one of the examples we provide, but you are also free to create your own original game (if you choose this option, *please speak with a TA first* so that we can ensure it is on the expected difficulty level).

Requirements:

Your *game* must use the following:

- At least **one struct**
- At least **one array**
- **Object pooling**
- A **state machine** including at least the following states:

- Start
- Pause
- Game
- Win (and/or lose)
  - It is okay if your game is a survival based game, and therefore only has a lose state
- You must be able to **navigate between the states** in some way (e.g. pressing button START while on the Start state takes you to the Game state, beating the game while on the Game state takes you to the Win state, etc.)

- At least **four moving objects**
- At least **four buttons** used for input
- **Collision that matters** (i.e. *something* must happen whenever two different objects hit each other)
- A **readme.txt** file
  - An instruction manual (of sorts) that tells a player how to play your game
- Only a **minimal amount of flicker**

Your *code* must have the following:

- At least **three .c** files (this time more than just main.c and myLib.c)
- At least **two .h** files
- Good organization (see tips below)
- Meaningful comments

Examples of games at the complexity level we expect:

- Tanks
- Simple flash games (e.g. https://thesimplearcade.com/)
- Simple Neopets games (http://neopets.com/games/)
- Old Atari games, like Asteroids (http://freeasteroids.org/)

Tips

- **Start early**. Never underestimate how long it takes to make a game!
- When splitting code between multiple files, put code that will be useful in multiple games in myLib.c, and code specific to this game in main.c or other files. Those other files should be specific to a concept (response to collision, a specific state of the state machine, etc.).
- Organize your code into functions specific to what that code does. **Your main method should not be very long at all!**

- Having update() and draw() functions that you either call directly in main() or call in another function that is called in main() is helpful.
- Make sure the order of calling your functions takes into account waiting for vBlank at the correct times to minimize flicker.
- Build upon the myLib.c and myLib.h files from previous assignments.
- Feel free to reference the Recitations files on Canvas to review concepts.
- Feel free to reach out to the TAs if you have any questions!

Submission Instructions:

Compress your entire project folder, including all source files, the Makefile, and everything produced during compilation (including the .gba file) into a single .zip file. Submit this .zip on Canvas. Name your submission HW03_FirstnameLastname, for example: “HW03_PrincessDaisy.zip”.