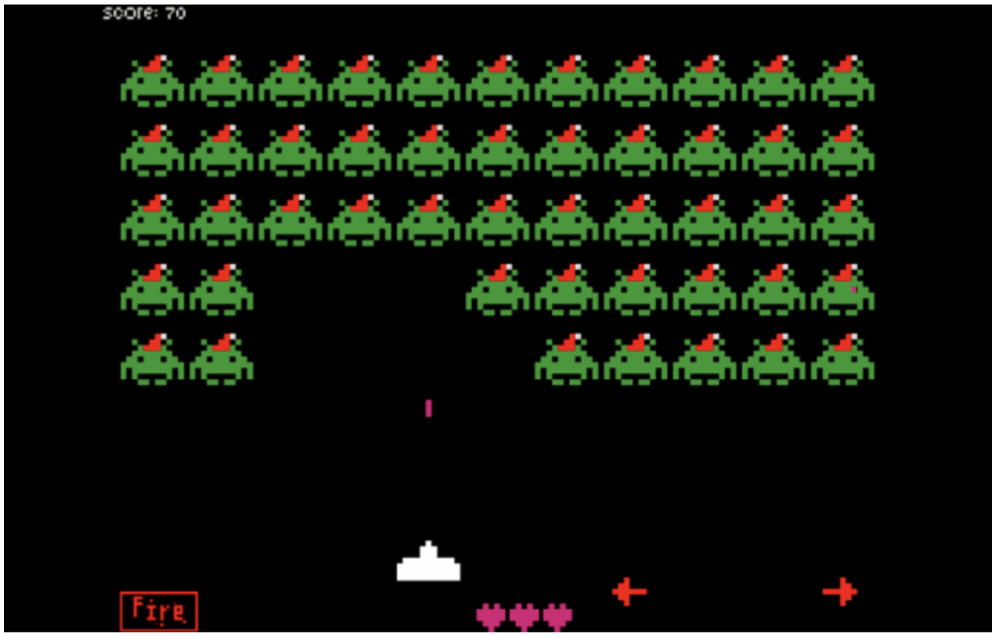
HOMEWORK 03
Extended Mode 3 Game



Purpose: To build a more complex game in Mode 3 to further your understanding of: analog sound, 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 <u>any of the labs</u>.** Also, **you must use the provided scaffold as the basis for your homework.**

The design of this game must be more complex than *catching/dodging falling boxes* or basic *Pong*. We are leaving this one somewhat open ended so that you can push yourself creatively and see what fun game you can come up with! You should adhere to the technical requirements and game mechanics listed below, but you can decide the theme and any unspecified mechanics.

Requirements:

Minimum Game Features

Your *game* must have the following:

- At least **two structs**.
- At least **one array.**
- **Object pooling.**
- A **state machine** including at least the following states:
  - Start,
  - Pause,
  - Game,
  - Win and/or Lose.
    - It is ok 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 the button START while on the Start state takes you to the Game state, winning the game while on Game states takes you to the Win state, etc.).
- At least **two types of moving objects**.
  - These must use two different structs. They must also *look* **and** *behave* differently (e.g. green player controlled by buttons and red enemy chasing player).
- **Collision that matters** (i.e. *something* must happen whenever two different objects touch each other -- think bullet+enemy, enemy+player, tomato+face, etc.).
  - This collision must be *something other than the player's collision with the collectable object or the exclusion area* (see Additional Required Mechanics section below). These collisions alone will not fulfill this requirement.
- At least **four buttons** used for input.
- At least **two different sound effects using analog sound**.
  - These sound effects should occur *when something happens* (e.g. collision with enemies, shooting bullets, losing a life, etc.).
- A **README.md** file.
  - An instruction manual (of sorts) that tells a player how to play your game, including things such as:
    - What each button does (controls),
    - How to navigate your state machine,
    - How to win and/or lose your game,
    - Anything that is buggy or not completed.
  - If you have little to no experience with Markdown syntax, here is a cheat

sheet and a more thorough explanation.
    - There are plenty of other resources online! Feel free to reference as many as you like.
  - You must use at **least one form of Markdown formatting** (header, bold, italics, etc.) and this use must make some sort of logical sense.
- A **minimal amount of flicker**.

Additional Required Mechanics

You must implement **at least two** of the following mechanics in your game:

- An **object that the player can collect** by colliding with the object and pressing a button at the same time. The player should also be able to both:
  - **Drop the object (returning it to the ground near the player's location)** by pressing the same button again (e.g. button A while colliding with it to collect, button A to drop it at the player's current location).
  - **Use the object (throw it, open a door with it, etc)** by pressing another button (continuing from above example, button B to open a door while holding the object).
- A **change in the player character's appearance** (shape, size, color, etc.) that occurs when something significant happens in the game. For example, if the player experiences damage, or if the player collects or drops an object.
- An **exclusion area** that the player cannot enter. The player can collide with the boundaries of this area, but cannot move any further into the area. This area must be at least 5 by 5 pixels, and **the location of the area must change once during gameplay.**

You may implement all three of these mechanics to receive **up to 5pts of extra credit** to your HW3 grade.

Code/files

Your *code* must have the following:

- At least **six .c files.**
  - The scaffold contains main.c, gba.c, font.c, text.c, and print.c. This means you have to create at least one more source file for this assignment.
- At least **five .h files.**
  - The scaffold contains gba.h, font.h, text.h, and print.h. This means you have to create at least one more header file for this assignment.
- Good organization (see tips below).
- Meaningful comments.

Tips:

- **Start early**. Never underestimate how long it takes to make a game! We give more time for this assignment compared to the previous two because ***we expect you to do a lot more work***.
- When splitting code between multiple files, put code that will be useful in multiple games in your gba.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 function should not be very long at all!**
- Having `update()` and `draw()` functions that you call directly in `main()` or within another function being 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.
- Use the HW03 scaffold (this is required!), and make sure that you create new .h and .c files for your game (also required!), e.g. `game.h` and `game.c`.
    - For better organization, we recommend you put all the game specific functions in this file, e.g. `updateGame()`.
- Reference the Recitations and Lectures files on Canvas to review concepts, and feel free to reach out to Aaron or the TAs if you have any questions!

Submission Instructions:

Ensure that **cleaning** and building/running your project still gives the expected results.

Please reference previous assignments for instructions on how to perform a "clean" command if you need clarification.

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 **HW03_LastnameFirstname**, for example:

“HW03_DaisyPrincess.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.