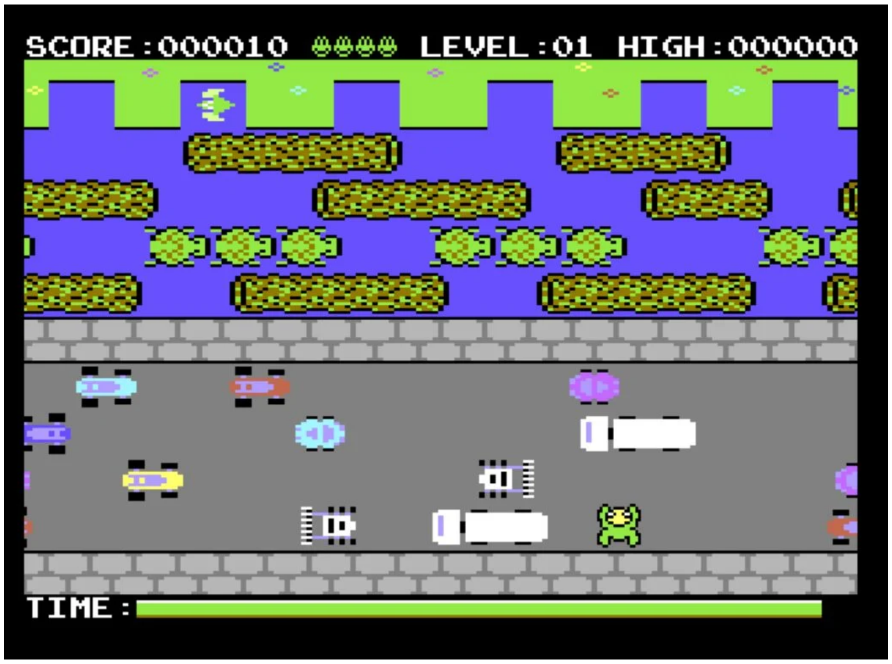
HOMEWORK 02:

Frogger



Purpose: To build a simple game in Mode 3 to further your understanding of: inputs to the GBA, collision detection and reaction, C basics, and drawing in Mode 3.

Instructions

For this homework, you will create a simplified version of the game *Frogger* in Mode 3. If you are unfamiliar with the game, you can watch this playthrough of Frogger. Review the **Requirements** section of this document for an explicit list of what we expect as the base requirements. The basic idea of our version of *Frogger* is as follows:

The player starts the game at the bottom of the screen and will try to traverse to the top. The player has three lives and can jump up, down, left and right by a set distance. The screen is divided into two main sections: the road and the river. On the road, cars move horizontally and will cause the player to lose a life if they are hit. In the river, logs move horizontally, and the player will lose a life if they enter the river without touching a log. When the player is touching a log, they move at the same rate as that log until they jump off. The player wins if they reach the top without being hit by a car or falling in the river, and they lose if they run out of lives.

Moreover, you are encouraged to be creative! Go outside of the requirements to add some flair to your game. **You will receive a maximum of 95 out of 100 points for meeting all of the base requirements. By adding your own flair, you *may* be awarded the 5 additional points needed to receive 100 points.**

Requirements

Gameplay

Your *game* must have the following:

- The player can move meaningfully with *intuitive* user input controls.
  - The player rectangle should be no smaller than 4x4 and should jump in the direction of movement by a set amount every time a movement button is pressed.
  - If a movement button is held, the player should only jump *once*.
- Rows should be spaced evenly and given a height so that the player's jumps cause them to move up and down by exactly one row.
  - The road should contain at least 2 rows, each of which must contain at least one car that moves horizontally. The cars in subsequent rows should move in opposite directions.
  - The river should contain at least 2 rows, each of which must contain at least one log that moves horizontally. The logs in subsequent should move in opposite directions.
  - There should be a safe row (no hazards) before the road, between the road and the river, and after the river.
- If the player collides with a car, they should lose a life.
- If the player collides with a log, they should move at the same speed as the log until they are no longer colliding with it.
  - The player should lose a life if they are in the river section without colliding with a log.
- The player should have 3 lives, visually indicated by a tally on the top-left corner of the screen. When the player loses all three lives, they should lose the game.
  - There should be a visual representation of when a life is lost, meaning that one of the boxes/rectangles in the tally should disappear.
  - Whenever the player loses a life, their position should be reset to the starting point.

- The player should win the game when they reach the safe zone after the river, and they should lose once they run out of lives.
- The game should enter an end state once the player wins or loses.
    - It must be clear that the game has ended; the player should not be able to play indefinitely. Preventing the player and atom from moving is okay here.
- Only a minimal amount of flicker.

Code/files

Your *code* must have the following:

- **You must use the provided HW02 scaffold!**
- Multiple `.c` files.
- At least one `.h` file.
- A **`readme.txt`** file.
    - This should contain an **instruction manual** that tells a player (but most importantly, your grading TA!) how to play your game.
- Good organization (see **Tips** section below).
- Meaningful comments.

Flair

Add *flair* to your game in order to receive the 5 points mentioned earlier. Some ideas:

- Create an animated player character!
- Change the end state to do something fun when a player wins.
- Add lots more logs and cars (without introducing flicker).
- Make your game look extra nice!

If you have questions about whether a specific flair will earn 5 points, please ask Luke about it!

Tips

- **Start early.** Never underestimate how long it takes to make a game!
- For collision code, draw pictures. Graph paper is your friend.
- When splitting code between multiple files, put code that will be useful in multiple games in your lib file (e.g. `gba.h`), and code specific to this game in `main.c` or other files. Those other files should be specific to a concept (collision, movement, etc.).
- Organize your code into functions specific to what that code does. **Your `main()` should not be very long.** Use helper functions!
- Having `update()` and `draw()` functions that you call in `main()` is helpful.

- Make sure the order in which you call your functions takes into account waiting for VBlank at the correct times. This will help you minimize flicker.

Submission Instructions:

Ensure that **cleaning** and building/running your project still gives the expected results. **Please reference the last page of previous assignments 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 HW02_LastnameFirstname, for example:

“HW02_SmithMichael.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.