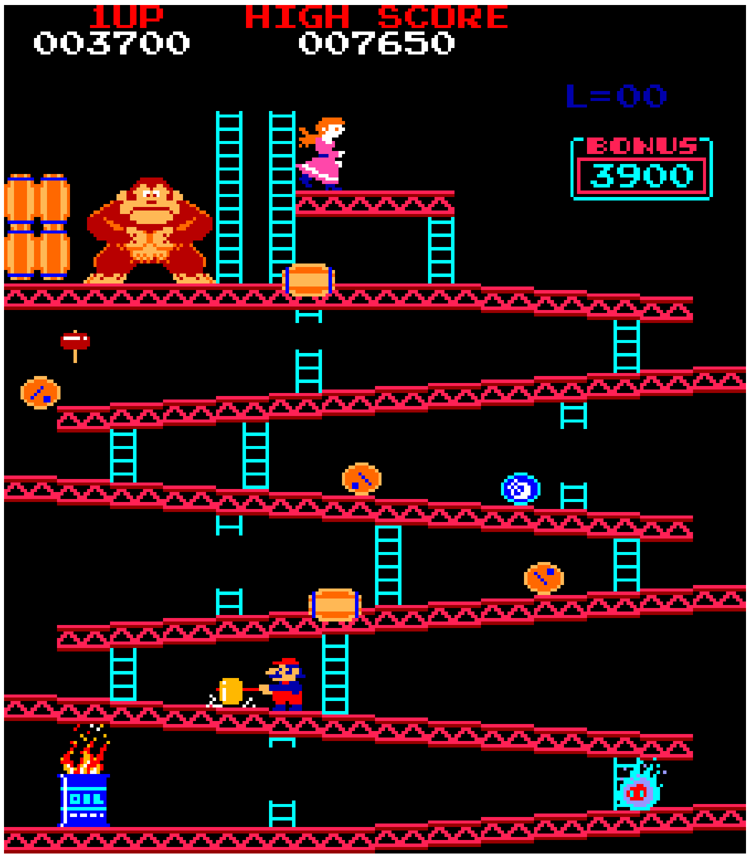
HOMEWORK 05

Donkey Kong - Mode 0 Game



Purpose: To build a Mode 0 game to further your understanding of tile modes, sprites, and backgrounds.

Instructions:

For this homework, you will recreate the classic 1980s arcade game, **"Donkey Kong"**, in **Mode 0**. To get a feel for the game, play the classic version here! You can also see some Donkey Kong play-throughs on Youtube, here and here for example. **Outside of the code in files provided in the Mode 0 scaffold, you must write all code yourself.**

You are free to expand upon or add your own twist to Donkey Kong, but the core components outlined in the requirements must be present.

Requirements:

Your *game* must have the following:

- **Core *Donkey-Kong* mechanics implemented**
  - **Player-controlled *Mario* character** that can move throughout an enclosed maze
    - *Mario* should be an animated sprite with at least 3 frames of animation
    - *Mario* should be able to walk right, walk left, jump
  - ***Donkey Kong* character** that sits on a platform, at the top of each level
    - *Donkey Kong* should be an animated sprite with at least 2 frames of animation
  - ***Pauline* character** that sits next to Donkey Kong
    - *Pauline* should be an animated sprite with at least 2 frames of animation
    - Once *Mario* collides with *Pauline*, the player is taken to the next level; after the last level, the player wins the game
  - Each level should be made up of **platforms and ladders**
    - *Mario* should be able to walk along platforms and climb up/down stairs
  - **Barrels thrown by *Donkey Kong***
    - These barrels should "roll" down the platforms and, occasionally, stairs
    - *Mario* should be able to ***jump over*** these barrels
  - *Mario* should have the opportunity to collect a **hammer "power-up"**
    - If it has been collected, *Mario* should change sprites to one holding a hammer, which he constantly swings
    - This hammer destroys barrels if *Mario* hits them while the power-up is active
    - After a few seconds, the power-up should deplete and *Mario* returns to his default sprite
  - A **"score"** counter displayed on the screen
    - The score should increase when *Mario* jumps over barrels, hits them with his hammer, or completes a level
  - A **"lives"** tracker displayed on the screen
    - Game should start with 3 lives
    - *Mario* should lose a life when *Mario* collides with *Donkey Kong* or barrels
    - Game ends when all lives are lost
- **Animated sprites** used for ***Mario***, ***Pauline***, and ***Donkey Kong***
- At least **two different levels with different design**

- ➔ **Transparency** used in a majority of the sprites
- ➔ A **state machine** with the following states: **Start, Pause, Game, Win, and Lose**
    - ◆ Your state screens besides your Game state can be static. You do not need to implement a timer in any state unless you so desire.

Your *code* must have the following:

- **Be entirely written in Mode 0**
    - Having the Mode 3 and Mode 4 functions alongside the others in HW05Lib.c is fine, as long as you **never** call them.
- **Multiple .c** files (more than just main.c and HW05Lib.c) and at least **two .h files**
- Good organization (see tips below)
- Meaningful comments

Tips:

- **Start early**. Never underestimate how long it takes to make a game!
- If you don't feel artistic, find the sprites for this game somewhere online, and copy them onto your spritesheet in Usenti.
    - Remember that your spritesheet is 256x256 pixels.
    - Tip: make sure your sprites are valic dimensions!
        - Check HW05Lib.h for the table with the dimensions!
- **Make use of the Mode 0 Scaffold! You can use this as a starting template to build your game.**
- Don't forget to enable sprites in the display control register and set up any backgrounds you've enabled!
- When splitting code between multiple files, put code that will be useful in multiple games in HW05Lib.c, and code specific to this game in main.c or other files. Those other files should be specific to a concept (collision, etc.).
- Organize your code into functions specific to what that code does. **Your main method should not be very long.**
    - Having update() and draw() functions that you call in main() is helpful.
    - Make sure the order takes into account waiting for vBlank at the correct times to minimize flicker, but also doesn't cause a drop in framerate.
- Feel free to reach out to 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 **HW05_LastnameFirstname**, for example:

“HW05_KongDonkey.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.