CS50's
Introduction to Game Development

OpenCourseWare

Colton Ogden (https://www.linkedin.com/in/colton-ogden-0514029b/)
cogden@cs50.harvard.edu

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan) (https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/) (https://orcid.org/0000-0001-5338-2522) (https://www.quora.com/profile/David-J-Malan) (https://www.reddit.com/user/davidjmalan) (https://www.tiktok.com/@davidjmalan) (https://davidjmalan.t.me/) (https://twitter.com/davidjmalan)

Flappy Bird

Objectives

- Read and understand all of the Flappy (Fifty!) Bird source code from Lecture 1.
- Influence the generation of pipes so as to bring about more complicated level generation.
- Give the player a medal for their performance, along with their score.
- Implement a pause feature, just in case life gets in the way of jumping through pipes!

Demo

by Edward Kang

Getting Started

Download the distro code for your game from cdn.cs50.net/games/2018/x/projects/1/flappy.zip (https://cdn.cs50.net/games/2018/x/projects/1/flappy.zip) and unzip `flappy.zip`, which should yield a directory called `flappy`.

Then, in a terminal window (located in `/Applications/Utilities` on Mac or by typing `cmd` in the Windows task bar), move to the directory where you extracted `flappy` (recall that the `cd` command can change your current directory), and run

```
cd flappy
```

Flapping Your Wings

Your second assignment won't be quite as easy as last week's, but don't worry! The pieces, taken one at a time, are still quite bite-sized and manageable and will mainly be a recap of what we've covered thoroughly in lecture leading up to this point :) For a refresher on LÖVE2D, as well as some helpful links for getting started, do just visit the following:

https://love2d.org/ (https://love2d.org/)

https://love2d.org/wiki/Getting_Started (https://love2d.org/wiki/Getting_Started)

Be sure to watch Lecture 1 and read through the code so you have a firm understanding of how it works before diving in! In particular, take note of where the logic is for spawning pipes and the parameters that drive both the gap between pipes and the interval at which pipes spawn, as those will be two primary components of this update! You'll be making some notable changes to the ScoreState, so be sure to read through that as well and get a sense for how images are stored, since you'll be incorporating your own! Lastly, think about what you need in order to incorporate a pause feature (a simple version of which we saw in lecture!). And if we want to pause the music, we'll probably need a method to do this that belongs to the audio object LÖVE gives us when we call `love.audio.newSource`; try browsing the documentation on the LÖVE2D wiki to find out what it is!

Specification

- Randomize the gap between pipes (vertical space), such that they're no longer hardcoded to 90 pixels.
- Randomize the interval at which pairs of pipes spawn, such that they're no longer always 2 seconds apart.
- When a player enters the ScoreState, award them a "medal" via an image displayed along with the score; this can be any image or any type of medal you choose (e.g., ribbons, actual medals, trophies, etc.), so long as each is different and based on the points they scored that life. Choose 3 different ones, as well as the minimum score needed for each one (though make it fair and not too hard to test :)).
- Implement a pause feature, such that the user can simply press "P" (or some other key) and pause the state of the game. This pause effect will be slightly fancier than the pause feature we showed in class, though not ultimately that much different. When they pause the game, a simple sound effect should play (I recommend testing out bfxr for this, as seen in Lecture 0!). At the same time this sound effect plays, the music should pause, and once the user presses P again, the gameplay and the music should resume just as they were! To cap it off, display a pause icon in the middle of the screen, nice and large, so as to make it clear the game is paused.

How to Submit

When you submit your project, the contents of your `games50/projects/2018/x/flappy` branch must match the file structure of the unzipped distribution code exactly as originally received. That is to say, your files should not be nested inside of any other directories of your own creation or otherwise deviate from the file structure we gave you. Your branch should also not contain any code from any other projects, only this one. Failure to adhere to this file structure will likely result in your submission being rejected.

By way of a simple example, for this project that means that if the grading staff visits `https://github.com/me50/USERNAME/blob/games50/projects/2018/x/flappy/PipePair.lua` (where `USERNAME` is your own GitHub username as provided in the form, below) we should be brought to the `PipePair.lua` file for Fifty Bird. If that's not how your code is organized when you check (e.g., you get a 404 error or don't see your edits), reorganize your repository as needed to match this paradigm.

1. If you haven't done so already, visit this link (https://submit.cs50.io/invites/46e6f2ea29954ce9bb1bdc478a440055), log in with your GitHub account, and click **Authorize cs50**. Then, check the box indicating that you'd like to grant course staff access to your submissions, and click **Join course**.

The change to `/projects/2018` below is intentional, as CS50 courses have changed to a scheme that reflects when the project was initially released. So the 2018 here is correct, even though it's no longer 2018!

2. Using Git (https://git-scm.com/downloads), push your work to `https://github.com/me50/USERNAME.git`, where `USERNAME` is your GitHub username, on a branch called `games50/projects/2018/x/flappy` or, if you've installed `submit50` (https://cs50.readthedocs.io/submit50/), execute

   ```
   submit50 games50/projects/2018/x/flappy
   ```

   instead.
3. Record a screencast (https://www.howtogeek.com/205742/how-to-record-your-windows-mac-linux-android-or-ios-screen/), not to exceed 5 minutes in length (and not uploaded more than one month prior to your submission of this project) in which you demonstrate your app's functionality. Upload that video to YouTube (https://www.youtube.com/upload) (as unlisted or public, but not private) or somewhere else.
   - To aid in the staff's review, in your video's description on YouTube, you should timestamp where each of the following occurs *in your gameplay demonstration*. This is **not optional**; failure to do this will result in your submission being rejected:
     - Clear difference in horizontal gap between pipes
     - Clear difference in vertical gap between pipes
     - Pausing the game

4. Submit this form (https://forms.cs50.io/5dac2f62-7526-4787-bc46-f30869aa834e).

You can then go to https://cs50.me/cs50g (https://cs50.me/cs50g) to view your current progress!