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)

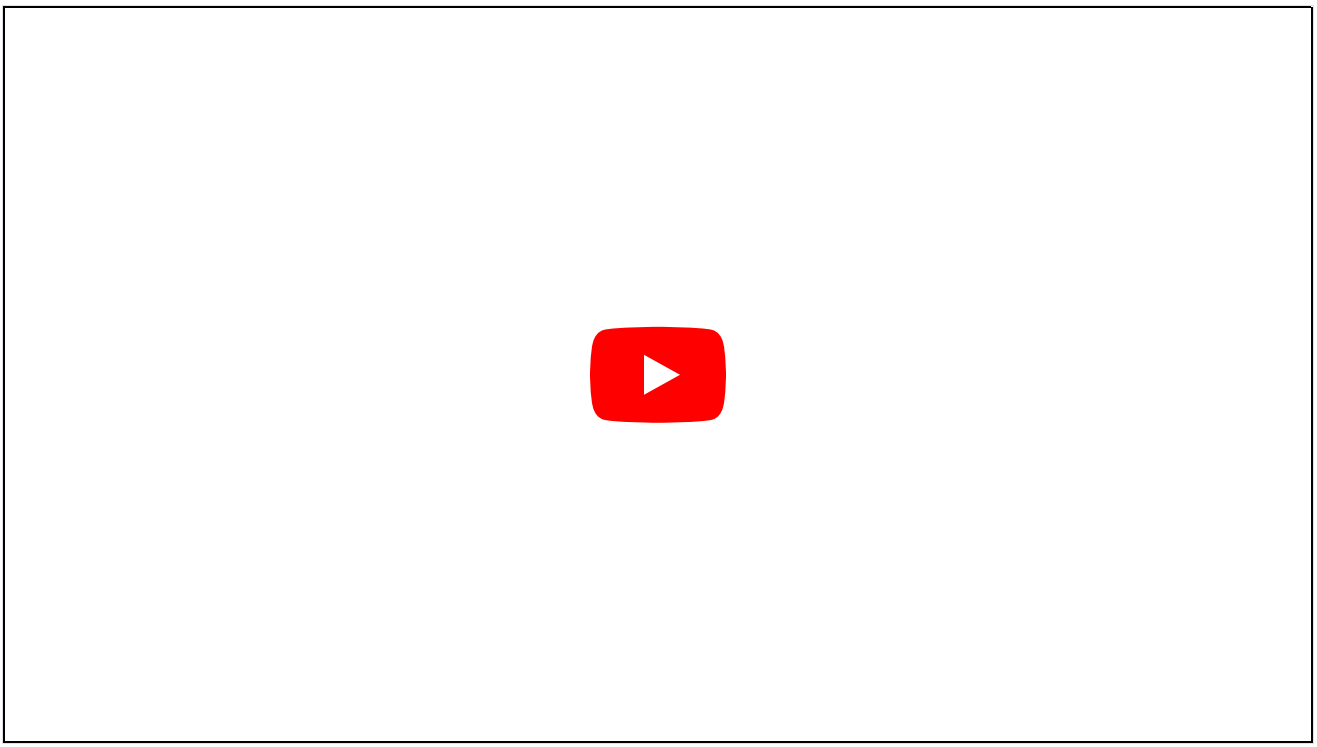
Zelda

Objectives

- Read and understand all of the Legend of Zelda source code from Lecture 5.
- Implement hearts that sometimes drop from enemies at random, which will heal the player for a full heart when picked up (consumed).
- Add pots to the game world (from the tile sheet) at random that the player can pick up, at which point their animation will change to reflect them carrying the pot (shown in the character sprite sheets). The player should not be able to swing their sword when in this state.
- When carrying a pot, the player should be able to throw the pot. When thrown, the pot will travel in a straight line based on where the player is looking. When it collides with a wall, travels more than four tiles, or collides with an enemy, it should disappear. When it collides with an enemy, it should do 1 point of damage to that enemy as well.

Demo

by Edward Kang



Getting Started

Download the distro code for your game from cdn.cs50.net/games/2018/x/projects/5/zelda.zip (https://cdn.cs50.net/games/2018/x/projects/5/zelda.zip) and unzip `zelda.zip`, which should yield a directory called `zelda`.

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 `zelda` (recall that the `cd` command can change your current directory), and run

```
cd zelda
```

A “Pot”ent Weapon

Welcome to your sixth assignment! We've explored the workings of a top-down adventure game in the style of Legend of Zelda and have a fair foundation for anything resembling it, be it a dungeon crawler or a vast 2D game featuring an overworld or the like. Let's add a few pieces to this sample in order to pay homage to some of the classic Zelda titles and to give our character

a shot at actually surviving his trek through the dungeon!

Specification

- *Implement hearts that sometimes drop from vanquished enemies at random, which will heal the player for a full heart when picked up (consumed).* Much of this we've already done in Super Mario Bros., so feel free to reuse some of the code in there! Recall that all `Entities` have a `health` field, including the `Player`. The `Player`'s health is measured numerically but represented via hearts; note that he can have half-hearts, which means that each individual heart should be worth 2 points of damage. Therefore, when we want to heal the `Player` for a full heart, be sure to increment health by 2, but be careful it doesn't go above the visual cap of 6, lest we appear to have a bug! Defining a `GameObject` that has an `onConsume` callback is probably of interest here, which you can refer back to Super Mario Bros. to get a sense of, though feel free to implement however best you see fit!
- *Add pots to the game world (from the tile sheet) at random that the player can pick up, at which point their animation will change to reflect them carrying the pot (shown in the character sprite sheets). The player should not be able to swing their sword when in this state.* In most of the Zelda titles, the hero is able to lift pots over his head, which he can then walk around with and throw at walls or enemies as he chooses. Implement this same functionality; you'll need to incorporate pot `GameObject`s, which should be collidable such that the `Player` can't walk through them. When he presses a key in front of them, perhaps `enter` or `return`, he should lift the pot above his head with an animation and then transition into a state where he walks around with the pot above his head. This will entail not only adding some new states for the `Player` but also ensuring a link exists (pun intended) between a pot and the character such that the pot always tracks the player's position so it can be rendered above his head. Be sure the `Player` cannot swing his sword while in this state, as his hands are full!
- *When carrying a pot, the player should be able to throw the pot. When thrown, the pot will travel in a straight line based on where the player is looking. When it collides with a wall, travels more than four tiles, or collides with an enemy, it should disappear. When it collides with an enemy, it should do 1 point of damage to that enemy as well.* Carrying the pot is one thing; the next step would be to be able to use the pot as a weapon! Allow the `Player` to throw the pot, effectively turning it into a projectile, and ensure it travels in a straight line depending on where the `Player` is facing when they throw it. When it collides with a wall, an enemy, or if it travels farther than four tiles in that direction, the pot should shatter, disappearing (although an actual shatter animation is optional). If it collides with an enemy, ensure the pot does 1 point of damage. There are many ways you can achieve this; think about how you can extend `GameObject` to fit this use case, perhaps adding a

`projectile` field and therefore a `dx` or `dy` to the `GameObject` to allow it to have traveling functionality. Perhaps include a `:fire` method as part of `GameObject` that will trigger this behavior as by passing in said `dx` and `dy` instead. The choice is yours, but `GameObject` is flexible enough to make it work!

How to Submit

When you submit your project, the contents of your `games50/projects/2018/x/zelda` 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/zelda/src/Player.lua` (where `USERNAME` is your own GitHub username as provided in the form, below) we should be brought to the `Player.lua` file for Legend of 50. 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/zelda` or, if you've installed `submit50` (https://cs50.readthedocs.io/submit50/), execute

   ```
   submit50 games50/projects/2018/x/zelda
   ```

   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:
        - Heart spawns from a vanquished enemy
        - Heart does not spawn from a vanquished enemy (thereby showing this drop is random)
        - Pot throw
4. Submit this form (https://forms.cs50.io/aaa37db8-d2eb-4faf-b1b2-c9e274bffb11).

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