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)

Dreadhalls

Objectives

- Read and understand all of the Dreadhalls source code from Lecture 9.
- Spawn holes in the floor of the maze that the player can fall through (but not too many; just three or four per maze is probably sufficient, depending on maze size).
- When the player falls through any holes, transition to a “Game Over” screen similar to the Title Screen, implemented as a separate scene. When the player presses “Enter” in the “Game Over” scene, they should be brought back to the title.
- Add a Text label to the Play scene that keeps track of which maze they’re in, incrementing each time they progress to the next maze. This can be implemented as a static variable, but it should be reset to 0 if they get a Game Over.

Demo

by Edward Kang

Getting Started

Download the distro code for your game from cdn.cs50.net/games/2018/x/projects/9/dreadhalls.zip (https://cdn.cs50.net/games/2018/x/projects/9/dreadhalls.zip) and unzip `dreadhalls.zip`, which should yield a directory called `dreadhalls`.

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

```
cd dreadhalls
```

Having some trouble with Unity? The staff has found that **Version 2018.4.28f1 (https://unity3d.com/unity/qa/lts-releases)** has worked well for them on a variety of different operating systems.

Falls in the Halls

Welcome to your tenth assignment! Though Unity may seem daunting at first, you're probably finding your way around the software more easily this time around. This week's assignment is fairly simple, but it will require you to get involved with scenes, part of the dungeon generation, and more; however, this and next week's assignment will be rather light compared to prior assignments so that you have more time to focus on your final project.

Specification

- *Spawn holes in the floor of the maze that the player can fall through (but not too many; just three or four per maze is probably sufficient, depending on maze size).* This should be very easy and only a few lines of code. The `LevelGenerator` script will be the place to look here; we aren't keeping track of floors or ceilings in the actual maze data being generated, so best to take a look at where the blocks are being insantiated (using the comments to help find!).
- *When the player falls through any holes, transition to a "Game Over" screen similar to the Title Screen, implemented as a separate scene. When the player presses "Enter" in the "Game Over" scene, they should be brought back to the title.* Recall which part of a Unity GameObject maintains control over its position, rotation, and scale? This will be the key to testing for a game over; identify which axis in Unity is up and down in our game world, and then simply check whether our character controller has gone below some given amount (lower than the ceiling block, presumably). Another fairly easy piece to put together, though you should probably create a `MonoBehaviour` for this one (something like `DespawnOnHeight`)! The "Game Over" scene that you should transition to can effectively be a copy of the Title scene, just with different wording for the `Text` labels. Do note that transitioning from the Play to the Game Over and then to the Title will result in the Play scene's music overlapping with the Title scene's music, since the Play scene's music is set to never destroy on load; therefore, how can we go about destroying the audio source object (named `WhisperSource`) at the right time to avoid the overlap?
- *Add a Text label to the Play scene that keeps track of which maze they're in, incrementing each time they progress to the next maze. This can be implemented as a static variable, but it should be reset to 0 if they get a Game Over.* This one should be fairly easy and can be accomplished using static variables; recall that they don't reset on scene reload. Where might be a good place to store it?

How to Submit

When you submit your project, the contents of your `games50/projects/2018/x/dreadhalls` 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/tree/games50/projects/2018/x/dreadhalls/Assets/Scripts` (where `USERNAME` is your own GitHub username as provided in the form, below) we should be brought to the `Scripts` subdirectory for Dreadhalls. 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/dreadhalls` or, if you've installed `submit50` (https://cs50.readthedocs.io/submit50/), execute

```
submit50 games50/projects/2018/x/dreadhalls
```

   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:
     - Level counter increase
     - Fall through hole/game over

4. Submit this form (https://forms.cs50.io/7dd45258-f989-4ea5-824f-983e929e554f).

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