Lab 11

BYOW Introduction

Announcements

Project 2B is due Monday, 10/30 at 11:59 pm.

Project 3 has been released!

- Phase 1 (3A) is due Monday, 11/13 at 11:59 pm.

Review Session this Friday (11/3) from 11-1 pm in Soda 2nd floor.

- They'll be covering SPT (shortest path) and MST (minimum spanning trees) algorithms along with Tries.

Build Your Own World (Intro)

Introduction

Build Your Own World or BYOW is Project 3, where our goal is to build a game that generates random, explorable worlds.

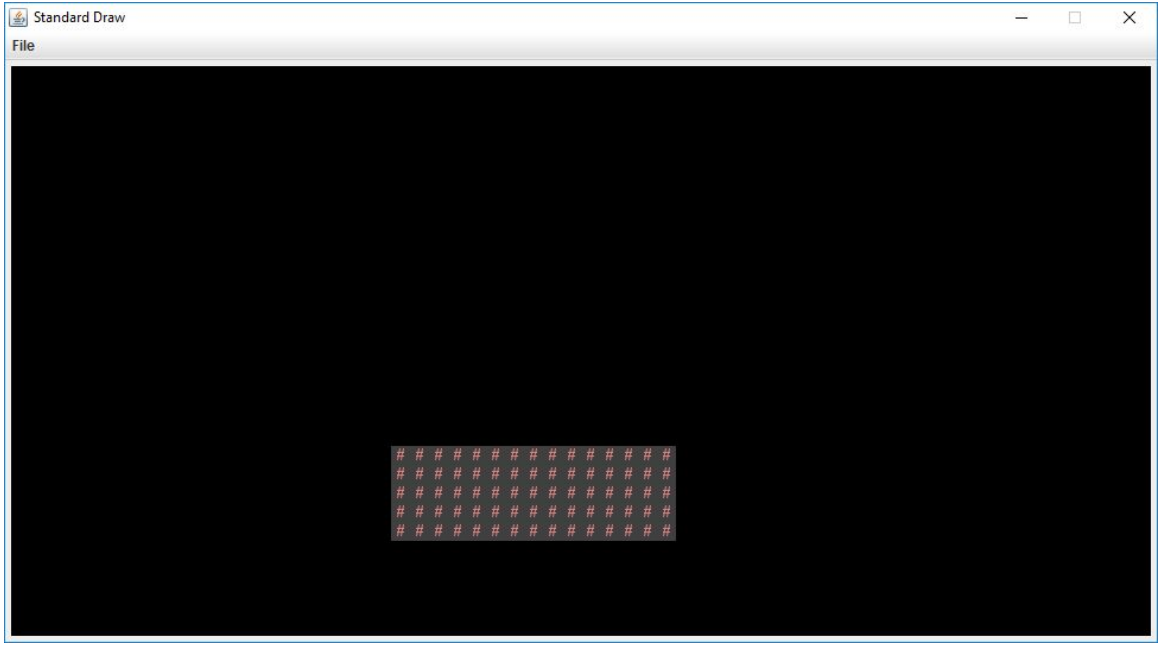
Throughout this project, there are several criterias that you must meet (as detailed in the spec). Lab 11 serves as an introduction to some of the tools you'll want to familiarize yourself with as well as other useful concepts to help you get started on the project.

World Generation

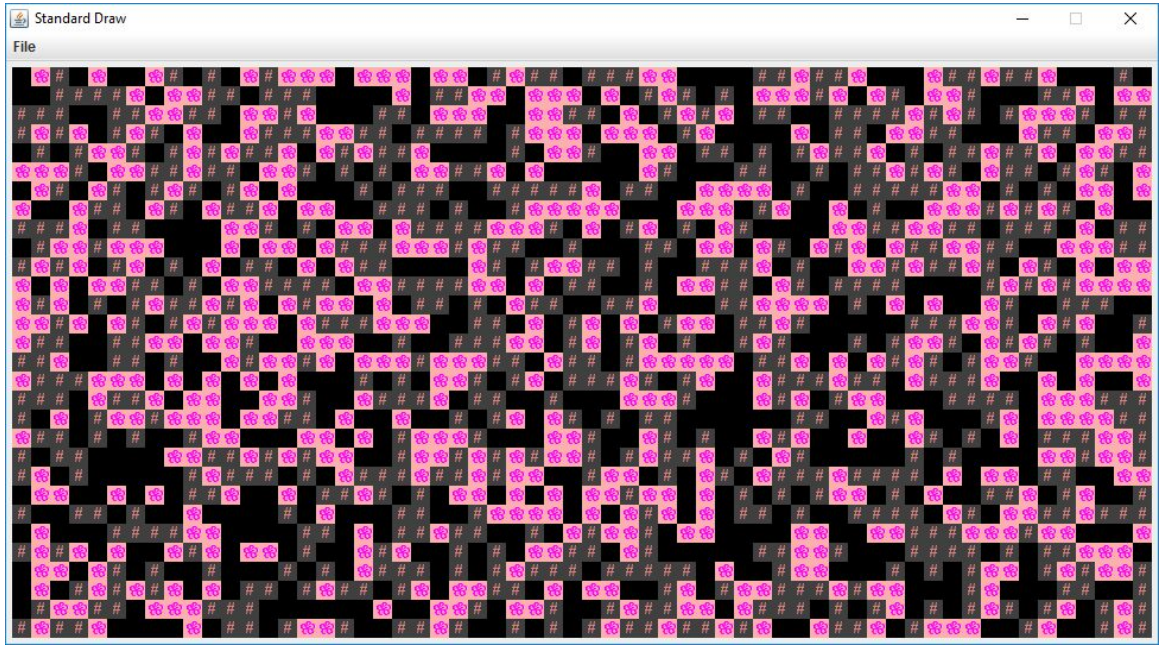
World Generation

What does it look like to build a world?

Something Like This (BoringWorldDemo)



Or This (RandomWorldDemo)



World Generation

What does it look like to build a world?

- For this project and lab, we will be using tiles. We will provide the following for you in lab and Project 3:
  - A tile rendering engine (**`TERenderer`**)
  - A representation of a 2-D tile array (**`TETile`**)
  - Some pre-generated tiles (**`Tileset`**)
- All of this can be found in the package **`tileengine`**

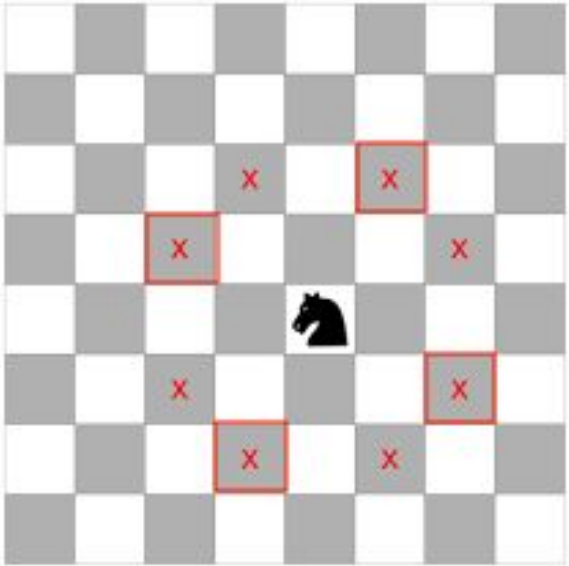
If you haven't already, go ahead and open up Lab 11 to take a look at these files/classes.

KnightWorld

KnightWorld

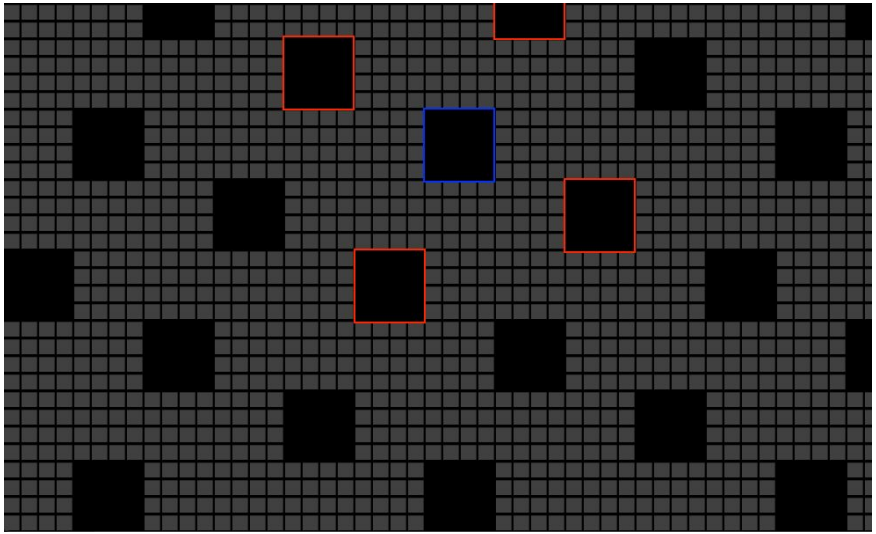
For this lab, our goal is to implement **`KnightWorld`**.

KnightWorld



The squares marked by X are possible locations where the knight can move. **However, we will only be concerned with the ones annotated with the red square.**

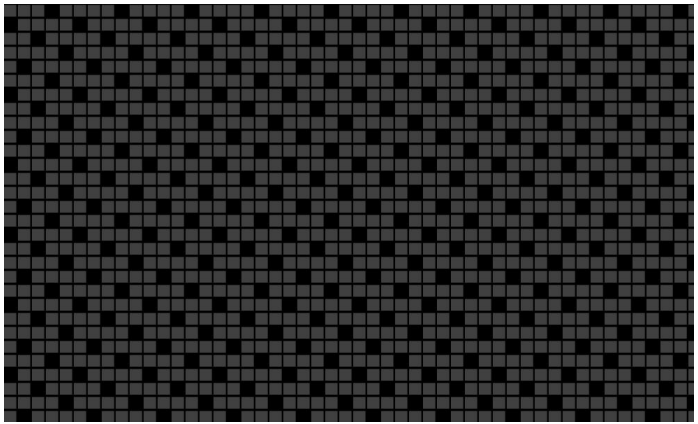
KnightWorld

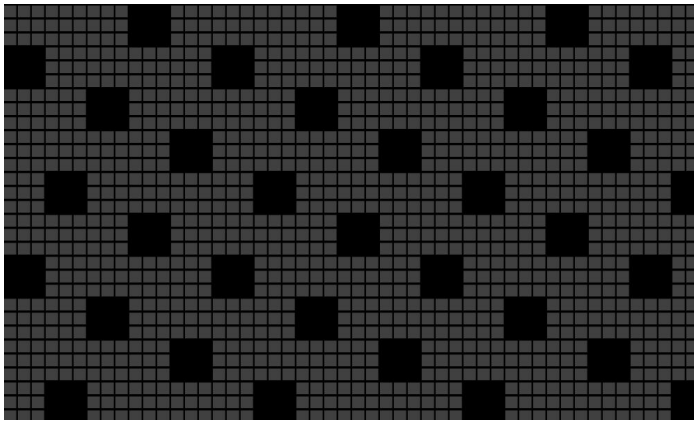


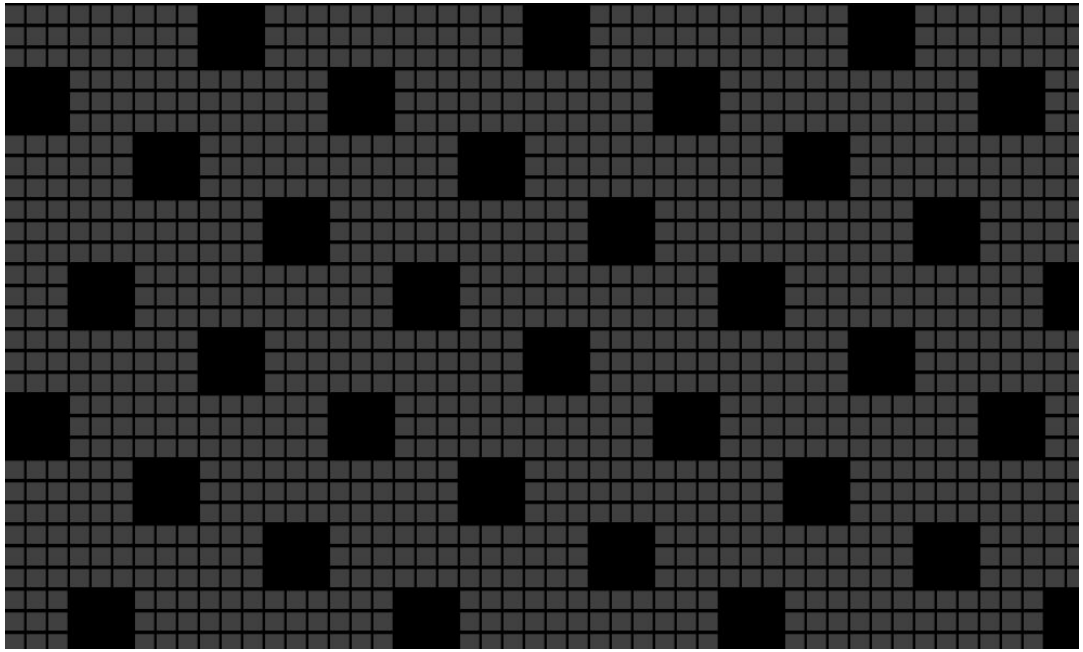
Specifically, we want to generate a pattern like the above (i.e. only concerned with every other knight move).

Knight World

Your "KnightWorld generator" should have the ability to draw differently-sized holes. The previous example is of size-4 holes, but you should be able to create worlds of size-1, 2, 3, etc. It should also work for varying world sizes.







Let's try to break this down. If we look at the world above (hole size of 3), what are some observations we can make? **How spaced apart is each "hole" relative to the "hole size" in each row? Do any of the rows repeat themselves?**

KnightWorld Tips

Some of these details are covered in the spec as well:

KnightWorld Tips

Some of these details are covered in the spec as well:

- Do not try to do this in a single method. It will be **significantly** much more difficult - break the problem down into smaller steps:

KnightWorld Tips

Some of these details are covered in the spec as well:

- Do not try to do this in a single method. It will be **significantly** much more difficult - break the problem down into smaller steps:
    - Everything in the world is a tile. Instead of alternating between two tile types, consider how you might draw out the “background” (this is dependent on your approach) first, then create the pattern with another tile.

KnightWorld Tips

Some of these details are covered in the spec as well:

- Do not try to do this in a single method. It will be **significantly** much more difficult - break the problem down into smaller steps:
  - Everything in the world is a tile. Instead of alternating between two tile types, consider how you might draw out the “background” (this is dependent on your approach) first, then create the pattern with another tile.
  - When creating the pattern, start by drawing out a square of a certain size. Then move onto drawing a row - how many tiles are placed between two holes? Then generalize this to the overall pattern - as mentioned, are there any repeating patterns between the rows?

Pseudorandomness and Persistence

Pseudorandomness

Pseudorandomness

- Earlier, we saw what **`RandomWorldDemo`** looks like. This can be generated through the creation of the object of type **`Random`** (pseudorandom number generator).

Pseudorandomness

Pseudorandomness

- Earlier, we saw what `RandomWorldDemo` looks like. This can be generated through the creation of the object of type `Random` (pseudorandom number generator).

Why is “pseudo”-random?

- The `Random` object can take in a seed (an integer) to generate these “random” values. If the same seed is provided to the constructor, the same sequence of values will be outputted. We recommend going over the example in the spec to see what this looks like.

Saving and Loading

There is a portion of the lab that is ungraded, but that we highly recommend going over.

- It's a short exercise on file reading and writing, but it translates over to a part of Project 3 where you will need to be able to save and load your game state.
- More details are given in the spec and in the skeleton code.

Lab Overview

An Overview

Lab 11 is due Friday, 11/03 at 11:59 pm.

Deliverables:

- Complete **`KnightWorld.java`** and get checked off → once you do so, a magic word will be given to you for submission on Gradescope (this can be done synchronously or asynchronous through Ed)
    - As an additional note, you don't have to use the same tiles in the examples presented, as long as you recreate the same pattern.

For help, use the Lab queue: [INSERT YOUR LAB QUEUE HERE]

Lab Notes

Some key takeaway from this for Project 3:

- The first part of Project 3 (Phase I) focuses on world generation, but each world you generate must appear distinct from one another (requirements mentioned in the spec). How can you use pseudorandomness to help you with this?
- What are some potential ways for you to save and load in a world state (this is related to the ungraded portion)? What information do you think should be saved?
- **Helper methods are your best friends**. **Do not try to do this in a single method.** It will make it much more difficult for you to debug and run through your logic if you do this.