Abstract
My goal for the semester was to create a game belonging to a favorite genre of mine, roguelikes. Requirements include not using available roguelike libraries, and being viable for web deployment.

Introduction
RogueScape is a game written in Java, belonging to the "roguelike" genre. While the goal of developing my own roguelike without using libraries has remained the same, the tools of implementation have changed significantly. What was once painstaking development in JavaScript and HTML5 has been much more enjoyable after switching to Java. I challenged myself to learn something new, and I have. I've learned to use the right tool for the right job. RogueScape is far from complete, but it's functional and I'm proud to have it as a pet project in the future.

Problem Scope
RogueScape does not particularly address any problems, besides needing a project for my capstone.

Requirements
I am my own client for RogueScape. The primary requirement for the game is that it be classifiable as a roguelike when all was said and done. According to the Berlin Interpretation, I believe RogueScape satisfies this requirement. Shallow and mindlessly simple as it currently is, it fits the interpretation.

Other requirements I've set for myself include not using available roguelike libraries, and being viable for web deployment. Other than an open source library called AsciiPanel, provided by fellow GitHub user **trystan**, all work has been done by and for myself.

Planning
Throughout the development of RogueScape, I've tried to model my engine to allow for the development of key features found in other roguelike games. This includes planning for graphical tile sets, items, inventories, combat, actions, and other events. The path was laid out for me; all I had to do is find out how I'd like to pave it.

The Berlin Interpretation
As previously stated, the requirements for a roguelike are practically prescribed by the genre itself. The Berlin Interpretation was created at the International Roguelike Development Conference 2008 to define "Roguelike."

Features from this interpretation that I've incorporated into the requirements for RogueScape include:

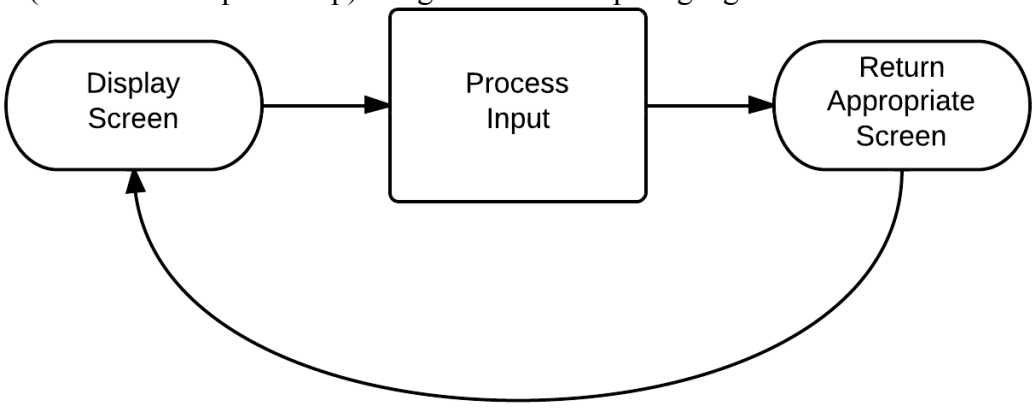
- **Random Environment Generation** – Every floor of the dungeon is procedurally generated.
- **Permadeath** – While death hasn't quite made it into the game, it would be permanent should it happen.
- **Turn-based** – Every action in game constitutes a turn, and an infinite amount of time is allowed between turns.
- **Grid-based** – The dungeons are created on a grid around which entities navigated.
- **Resource management** – Living entities in the game have health that they must maintain, else perish.
- **Hack'n'slash** – The bulk of the game is slicing through dragons by running into them headfirst. Killing monsters is a core feature.
- **Exploration and discovery** – Unfortunately my algorithm for lighting only places the player been to or can see is dysfunctional, but was an intended feature.
- **ASCII display** – Accomplished thanks to the AsciiPanel library. My previous solution was reasonable but clunky.
- **Dungeons** – The game contains dungeons which are procedurally generated.

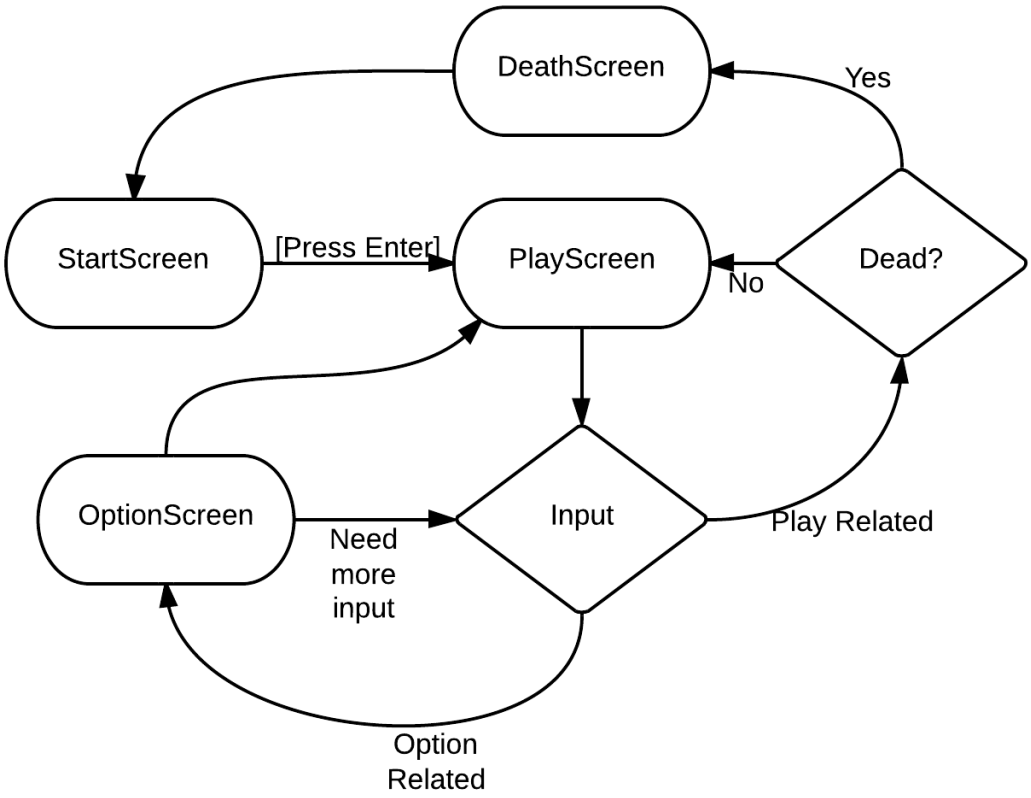
Design

Nearly every class was derived inherently or created specifically to support one of two key interfaces: Entity and Screen.

Entity objects represent everything that can be seen or used in the game: a tile in the dungeon, a player, or an item on the ground. These things all have representations, names, traits, and even the ability to affect each other.

Meanwhile, Screen objects handle the interfacing of the player's actions with the logic game itself. Each screen has a purpose, a start screen, a player screen, an options screen, and even a death screen. A screen takes input in, evaluates the change in game state, and returns an appropriate screen. It is very analogous to the REPL (read-evaluate-print loop) design found in Lisp languages.





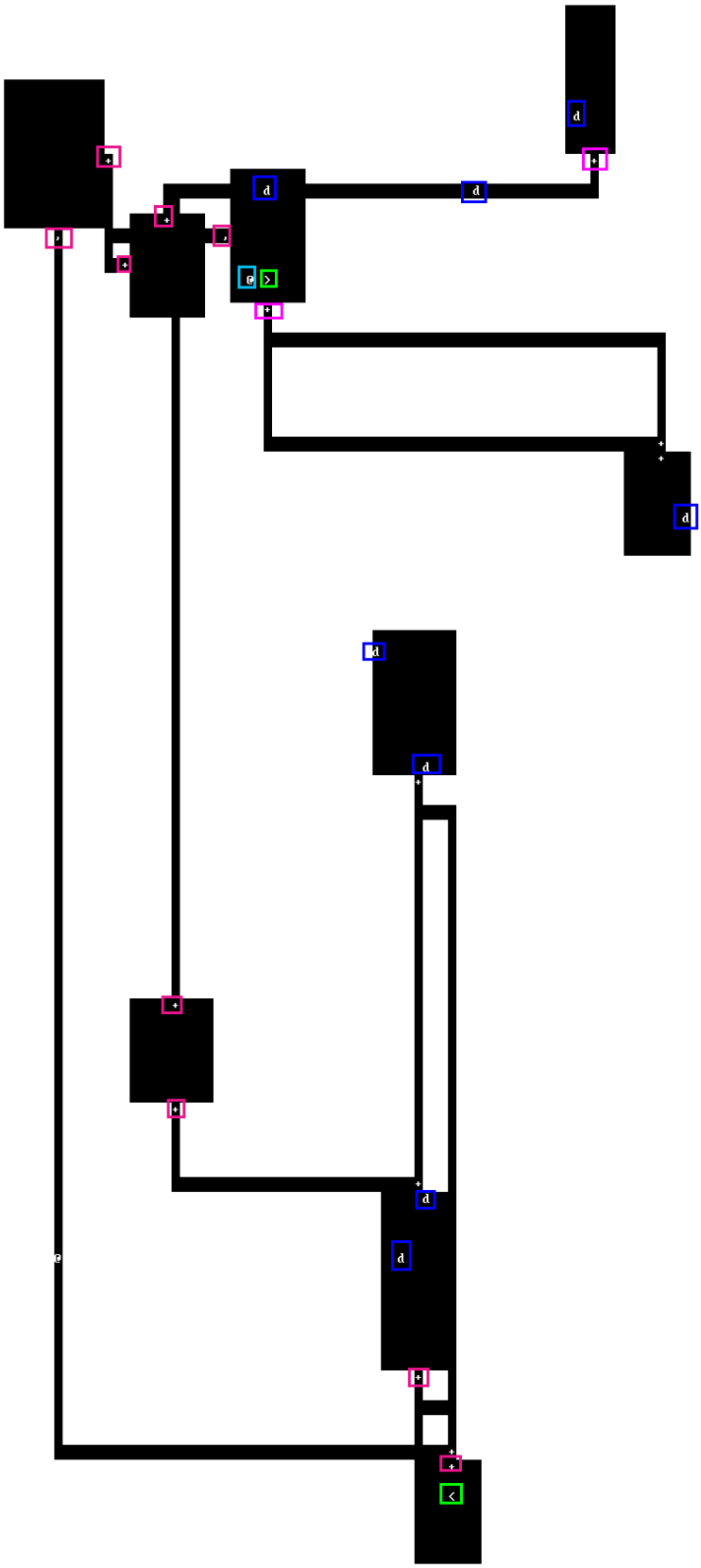
The hardest part of design, the only aspect in which I'm disappointed in my progress, is theme and content. The game is lacking of a storyline, characters, items, etc. respective to a particular theme and age.

Dungeon Generation

Dungeon generation turned out to be a tricky process. My algorithm boiled down to a few simple steps:

- Fill in the grid with stone
- Carve out an appropriate numbers of rectangle rooms of appropriate size
- Connect rooms with corridors so that none are unreachable
- Place doors at the ends of each corridor, and stairs up and down somewhere reachable.
- Populate the dungeon with monsters.

An example:



Development Process
I've tried to model my development process after the agile method. I believe I could have accomplished more had I stuck to the this agile method better by making more use of techniques such as task analysis and estimation, and burn up charts. However, due to a vast base feature set and a limited time frame, the order in which features were developed has been more or less prescribed by the nature of the project.

My development cycle consisted mostly of developing a core feature, testing its integration, debugging, then moving on to the next required feature. The first important features included dungeon generation, living entities such as players and creatures, and handling user input.

Another hurdle in the development process was trying to use unfamiliar technologies. The initial goal of using HTML5 and JavaScript was admirable, but difficult. After changing gears more than halfway through the semester, I was able to accomplish more than I had to date by using Java.

My work schedule consisted of working on the project anytime I found while trying to keep up in my other classes.

Analysis, Results, Discussion
RogueScape works. It's enough to be considered a roguelike. It's deployable to the web via applets, and has been written without the use of roguelike libraries. While I have not accomplished as much as I hoped to, I have learned a lot.

The game is available at https://github.com/dmsalsman/RogueScape and is open source under the permissive MIT license.

My only regret in this project is not having a smaller workload.

References
Berlin Interpretation
http://www.roguebasin.com/index.php?title=Berlin_Interpretation

AsciiPanel
https://github.com/trystan/AsciiPanel