# Project Milestone 2

### CS 4500 Software Development

**Due:** ~~Monday, February 15th, 9pm~~Tuesday, February 16th, 9pm

**Submission:**

1. In the `Snarl` directory in your repository, include the following:
   - for the Design Task, place `state.md` in `Planning/`
   - for the Programming Task, place `level.<ext>` in a new `Game/` directory within `Snarl`
2. Once you are ready to submit, create a release on Github and tag it `p2`.
3. Manually create a ZIP file, only containing `state.md` and `level.<ext>`, together with any *source* files referenced by it and authored by you. Submit this ZIP file via Handins.

---

Keep the Snarl overview and plan in mind when working on these tasks.

## Design Task

An implementation of Snarl demands a data representation for game states. A Snarl state should contain information necessary to check validity of moves and progress the game. The full state will be private to the Game Manager, while other components (players, AIs) might be provided a restricted view of the state.

Describe a data representation for Snarl game states. Think back to data definitions in *Fundies I* and use a mix of English and the data definition constructs from your chosen language. Add a description of an interface with operations that other components may need to perform on the game state, or to interact with it. This might look like a wishlist with function signatures and purpose statements.

The memo must not exceed two pages. Less is more.

**Scope:** The purpose of this task is to think about what information is relevant for the game manager to discharge its responsibilities of running a dungeon, managing actors and progressing the game; and how this information should be represented. We are looking for a careful analysis of the information available to you (including any clarifications), not a perfect spec set in stone.

## Programming Task

Design and implement a Snarl level representation. A level is comprised of *rooms* and *hallways*.

A *room* has an upper-left Cartesian position, boundary dimensions (or size), a layout of non-wall tiles, and one or more ~~exits~~doors. Objects, like the key and the level exit, may be inside of a room.

A ~~*exit*~~*room door* is valid if it is at the boundary dimensions of the room.

A *hallway* has 2 rooms to connect (via their doors) and a (possibly empty) list of waypoints. The waypoints allow for corners in the hallway. ~~A hallway is valid if a line comprised of the waypoints connects the two rooms.~~ A hallway is valid only if:

a) it connects two rooms at its endpoints; and
b) each segment (as delimited by subsequent points[1]) is either horizontal or vertical (i.e., perpendicular with the x or y axis).

A *level* is comprised of a series of rooms connected by hallways. A level is valid if no two rooms overlap, no two hallways overlap, and no hallways overlap with any rooms.

Include examples of level data in the code, again like in *Fundies I*.

> *This representation will help us generate interesting levels in a later milestone.*

Design a function for rendering levels graphically. This function may generate ASCII art, graphical images (if your PL/IDE support this), etc. You may use outside libraries to support with GUI creation, but not for the actual rendering of your data representations.

**Scope:** The purpose of this task it to analyze an informal description of the level and design a corresponding representation. We will be looking for good code design, readability, and if we can find the functionality we asked for in the code.

## Changes

| | |
|---|---|
| **02/10/21** | • (Design Task) Completed a sentence<br>• (Programming Task) *Levels* have an *exit*, *rooms* have *doors*<br>• (Programming Task) More specific validity conditions for hallways (no angled hallways) |

[1] Here, a *point* means either an endpoint or a waypoint.