Project

Project proposal: *Lambdabirynth*

A short description of the program

> I opted for a 2D game. More specifically, I will make a roguelike in the terminal -- much like the original Rogue.

Backstory (*ish*)

As you wander through the desert (why is unbeknownst to me), you come across an ancient-looking entrance of some sort. It is shaped like a Λ.
You enter, and find yourself at the top of a staircase into the darkness.
Descending, you enter a dark room.
As you light your torch and peer around, you see doors that are shaped like λ's, and descending stairways in the shape of Λ's.

> **Excuses...**
>
> (I had to find an excuse to use the name, OK?)

"I must get to the lowest level", you think, as cursed monsters are awaken from their slumber.
Little do you know there is no such end, the *Lambdabirynth* is a **lazily-evaluated endless labyrinth**!

Minimum viable product (MVP)

> I will explain the planned game features in the shape of an MVP, and use this as a basis for planning and deliverables.

Explanation	Acceptance criteria
Player can choose name and class	Front page with name input and class selection (*wizard*, *explorer,* etc.)

Explanation	Acceptance criteria
Procedural random map generation	Map consists of levels, each level of rooms. Rooms are connected with tunnels, and levels with stairs.
Player movement, attacks, and using items	Player can move with `WASD`, attack monsters with `R`, and use inventory items by clicking on them.
Randomly placed chests with items	Player can find and open chests. Items can be transferred from chest to player inventory.
Randomly generated selection of monsters	At least two monster types that try to kill the player. Implement FSM, movement, attacks, and death.
Player inventory + equipped items	A separate pane with the player's inventory. The player has armour and weapon slots for equipped items that modify player stats.
Player death	It's all in the name! HP reaches 0, and the player dies. No saving; you die, you restart.

Additional comments

- Movement is turn-based, i.e., monsters move only after player has completed some action.
- The map construction will be heavily inspired from the original *Rogue*.
- The original *Rogue* used symbols for doors, monsters, etc. We can use *emojis*!
  Oh, and add a flag to disable emojis for terminals that don't support them.

Expanding on MVP if time

- Classes need not *do* anything in MVP. Implement some attributes/abilities.
- All rooms on a level will be visible by default. Add *fog-of-war* to improve exportability.
  - We could drag this even further with adding lighting, where the player can hold a torch, and light torches on the wall.
- Add looting to dead monsters.

An overview of the program design and libraries you will use

I will use *Brick* for the TUI aspect of the game.
I've seen several instances of games running in Brick, like Tetris and Solitaire.
I mentioned a flag to disable emojis, this seems like a good fit for *optparse-*

applicative.
As far as I can see, there is no need for additional libraries given my current scope.

When it comes to program design, I don't know what the best program/module structure will be.
I've found this article, which I suppose is a good starting point.
Also, to understand more about how I should use Brick (which I've never touched before), I found this article from the author of the *Solitaire* project.

An explanation of which functional programming techniques will be used

- The `State` monad is a good starting point to model an FSM for monster behaviour.
- The player (and monsters) will have stats like health, attack power, etc. I think the `Reader` monad is a good starting point.
- Monad transformers! Can't get anywhere without them, can we? I don't know yet what the monad stacks will look like, but we'll cross that bridge when we get there :D

There is a lot of uncharted territory for me, here, so there will be a lot of learning along the way.
However, the Internet is full of resources, it didn't take me long to find this list, which I will probably use a lot.

A discussion of what you expect to complete before the deadline.

See MVP.