HOW TO SOLO?

REFLECTIONS ON GAME CONCEPT VALIDATION

PROCESSES

EETU LEPPÄLÄ

WRITTEN AS PART OF THE COLLABORATIVE AND INDUSTRIAL
DESIGN DEGREE IN AALTO UNIVERSTIY

ACADEMIC ADVISOR: SEVERI UUSITALO

18.01.2023

TABLE OF CONTENTS

1 INTRODUCTION 4

1.1 PROJECT CONSTRAINTS **4**
1.2 GAME PROTOTYPES **5**

2 CONCIERGE 6

2.1 DESIGN PROCESS **6**
2.2 REFLECTIONS **11**

3 CONCIERGE CARDS 12

3.1 DESIGN PROCESS **12**
3.2 REFLECTIONS **15**

4 SPACE SALOON 16

4.1 DESIGN PROCESS **16**
INITIAL DESIGN 16
TECH DEMO 20
ALTERNATIVE PLAY 22
4.2 REFLECTIONS **25**

5 HEGEMONY 28

5.1 DESIGN PROCESS **28**
5.2 REFLECTIONS **28**

6 MOVING MANOR 30

6.1 DESIGN PROCESS **30**
PUZZLE GAME 30
6.2 DECK-BUILDER **33**
6.3 REFLECTIONS **40**

7 MOVING MANOR – NEXT STEPS 42

8 LEARNINGS 44

9 RESOURCES AND LINKS 46

1 INTRODUCTION

In this report, I will reflect on the game design techniques I utilised when trying to find an optimal long-term game project to work on next. This project is a continuation of my graduate thesis project, where I explored the importance of meaningful play in learning games. I wanted to continue this work, now focusing less on the game being explicitly thought of as a learning game, and more on it being a complete video game, that may produce some learnings when played.

I will first outline the goal state and constraints for the entire endeavour, as well as give a brief overview of each prototype. Then I will delve into each game concept and prototype and the successes and failures made along the way, focusing on the choices I made that affected the ultimate decision of whether to continue with the iteration. In the end, I will lay out collected learnings from these prototypes.

1.1 PROJECT CONSTRAINTS

Seeing that I was working on this alone, I needed to find a game project that I could finish optimally by myself. While I still haven't excluded the possibility of hiring outside help when needed, my current situation is such that I wanted to preserve the option of working alone. This meant I needed a simple game in terms of the complexity of project architecture.

I also set myself an artificial constraint that I would be able to finish the selected game project within a year. This also constrained the space of the potential games further.

Seeing that I had just finished working on my thesis project before starting on this project, I still aimed at creating gameplay that was not only fun but would also foster some key capabilities we as humans need to thrive on this planet together. While this was secondary to the aim of just completing a solo game project, it still guided me.

Also, as I explored in my thesis, the prerequisite for any game that aims to teach something is that it needs to be interesting enough for people to play it. This is done by enabling players to extract meaning from the game. In my opinion, by far the most effective tool to create this type of meaning is by creating complex yet distilled play spaces. I wanted to find a game concept that in its foundation exhibited the potential to create emergent gameplay from the interesting combination of game elements while maintaining a simple exterior. Ultimately, finding a game concept like this would cut down on development time in relation to the potential playtime within the game.

1.2 GAME PROTOTYPES

I worked on the game prototypes over the period of approximately 6 months. The project has logged over 300 hours at the time of writing this document.

Here is a summary of all the game prototypes I worked on:

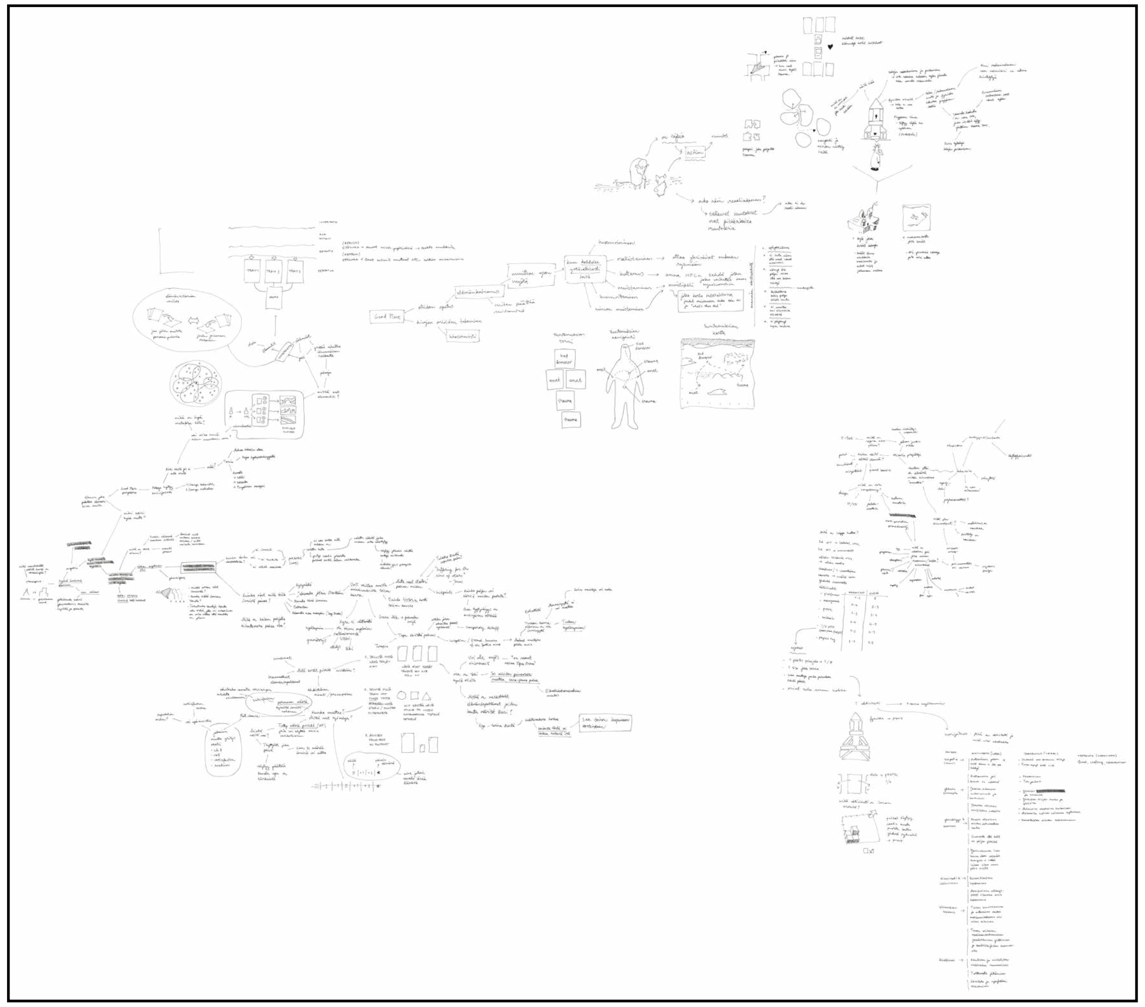
- Concierge – Game concept exploring how to train players in empathy in a hotel management setting.
- Concierge 2 – A more refined concept, where the player has to balance running a hotel operation while ensuring customer satisfaction.
- Concierge Cards – A technical prototype of Concierge 2, exploring if a particular card-based mechanic would induce interesting gameplay in this context.
- Space Saloon – A tile-placing game prototype. A natural continuation to Concierge, where you still need to satisfy customers but it's all tied to building the optimal hotel structure for each batch of customers.
- Space Saloon Tech demo – Programmed demo implementing basic room movement, rotation and checking for accessibility between rooms. Becomes important later.
- Hegemony – A single-player mobile game version of the game Crisis, which I developed for my graduate thesis.
- Moving Manor Puzzle – A dungeon-crawler puzzle game build on the constraints of the Space Saloon tech demo.
- Moving Manor deck-builder – A dungeon-crawling rogue-lite deck builder with puzzle elements.

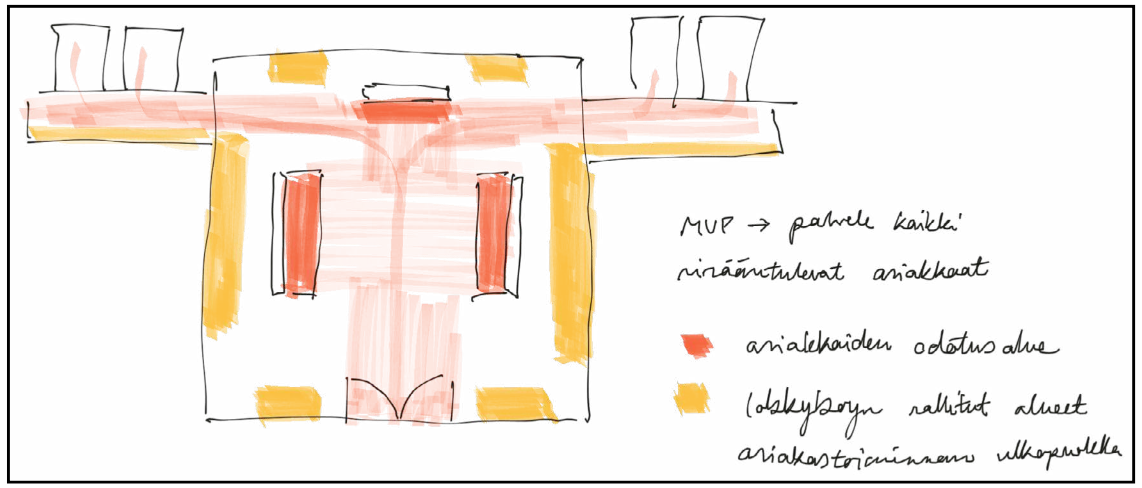
2 CONCIERGE

A hotel management game where you need to go above and beyond to fulfil the requests of your customers to increase your ratings, while simultaneously keeping your hotel operations running without a hitch.

2.1 DESIGN PROCESS

The idea behind Concierge began when I first watched The Good Place. In it, the protagonist finds themselves in a pickle: after their death, they have been accidentally placed in the Good Place, a religion-agnostic heaven-equivalent, even though during their time in the Land of the Living they were a horrible person. An ethics professor is let in on this secret, and they advise the protagonist to start doing good deeds even though their intentions were not pure. This way the behaviour comes first and the intention to be good will follow.


Mindmapping, trying to find good arguments the game might make, in conjunction with game mechanics, that could create desired behaviours.



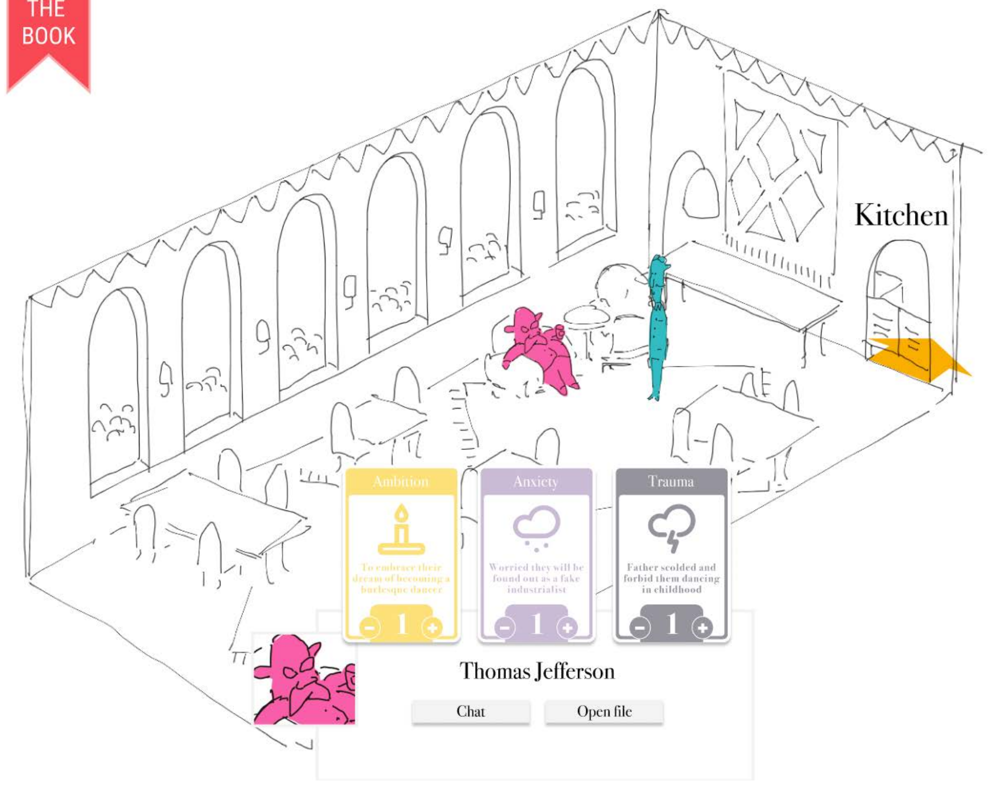
The first idea of how the game could work. Areas could be designated for different activities, and workers could follow those guidelines.

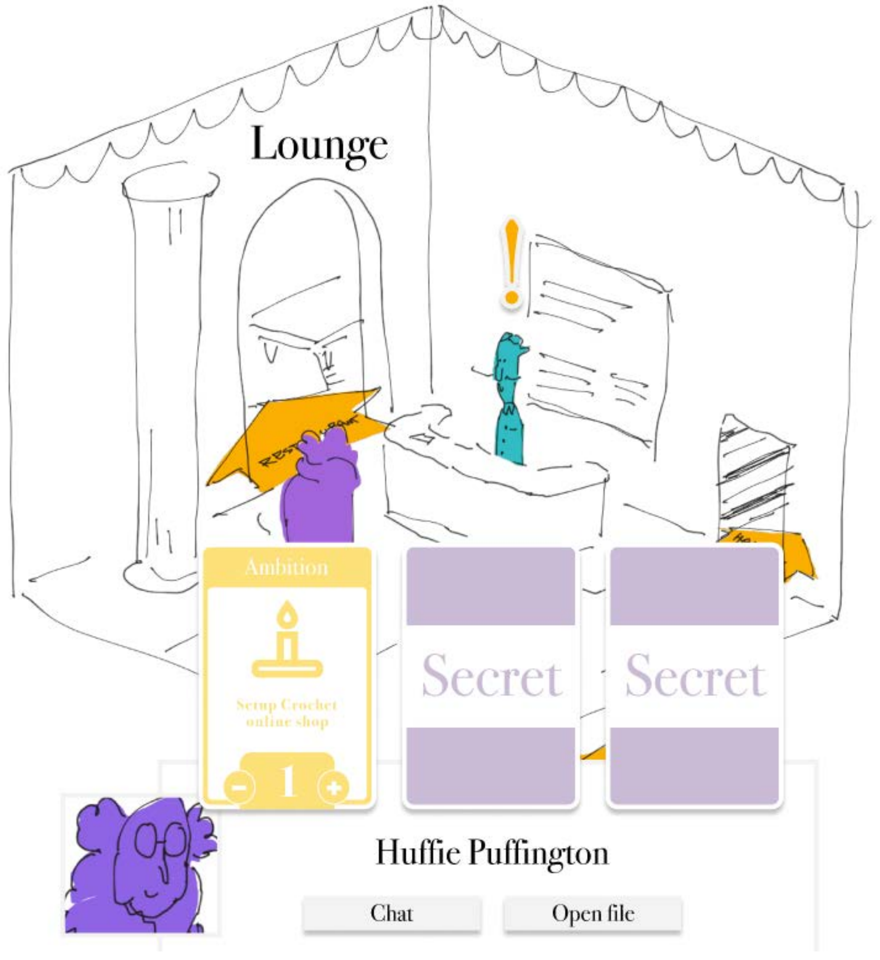
In a similar way, I wanted to explore the idea of simulating an environment where you need to treat others with respect and get others to feel good. Would this encourage moral behaviour in the real world? To unpack this, I started researching what empathy and acting for others' benefit is or could be as a game experience.

In order to force players to act empathetically, I wanted to find a real-world counterpart, where it could be naturally enforced. The two candidates at the time were the hospitality industry and nursing in hospitals. From these two, I felt like the hospitality industry would be more light-hearted, whereas in hospitals you really have no choice but to take care of the wounded and sick.

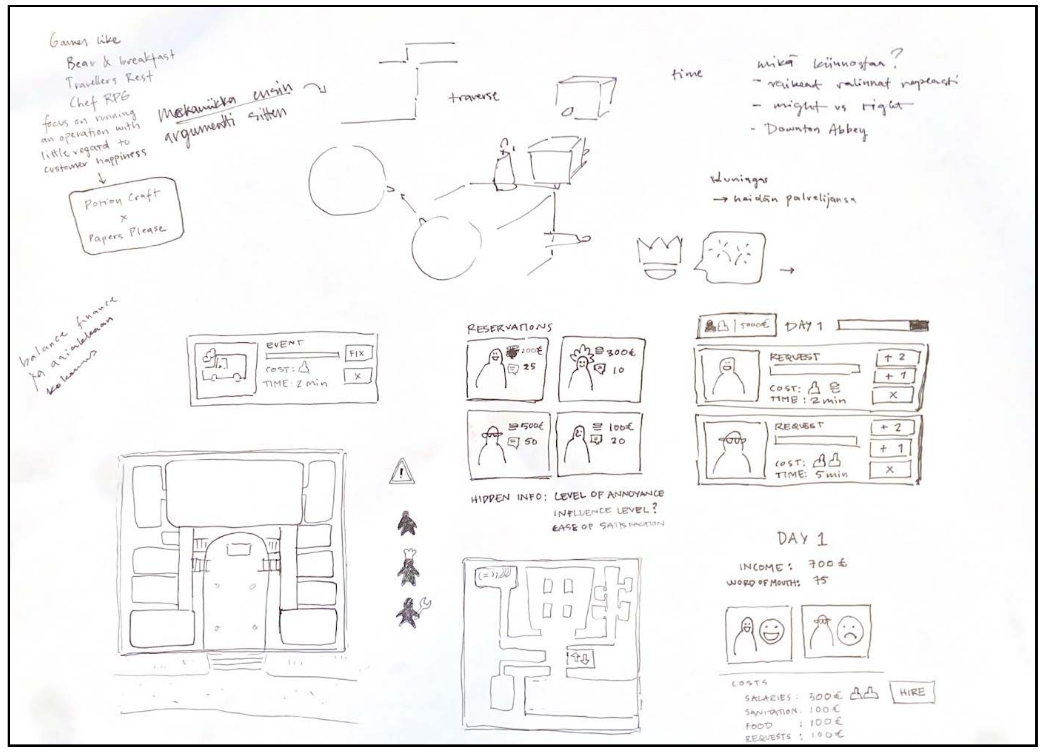
After these quick ideations and once I had settled on the concept of setting the game in the world of fancy hotels, I went to Figma and created an interactive prototype just to feel out what the game could be like. The idea of the game was that you would need to get to know your customers so well that you could actually arrange for them things they haven't even asked for. But I never actually got an actual playable demo of this game. I just couldn't figure out an interesting way to turn this type of gameplay into discrete mechanics, especially the kinds of mechanics I would be comfortable coding. Mainly I just concepted more and imagined how good the game would eventually be. Finally, it just got stuffed into the drawer for a later day.

A couple of months passed and I decided to give it another shot. This eventually morphed into Concierge 2, where the interesting part was that the player had to find out and fulfil what their customers want while also managing the day-to-day hotel operations. This meant the player would have to choose every time to be nice and go the extra mile for their customers, and not just take care of the routine of running a hotel. This would be the only way to get your ratings to rise, and for you to progress in the game.

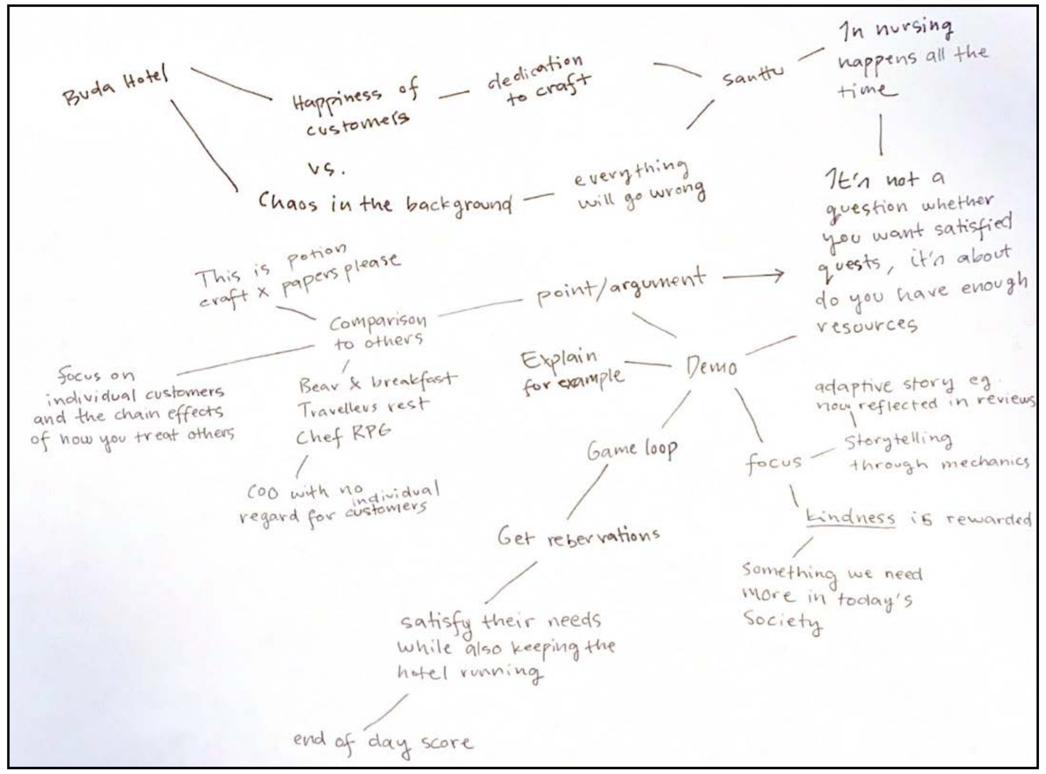

Kitchen
Ambition
To embrace their dream of becoming a burlesque dancer
Anxiety
Worried they will be found out as a fake industrialist
Trauma
Father scolded and forbid them dancing in childhood
Thomas Jefferson
Chat
Open file



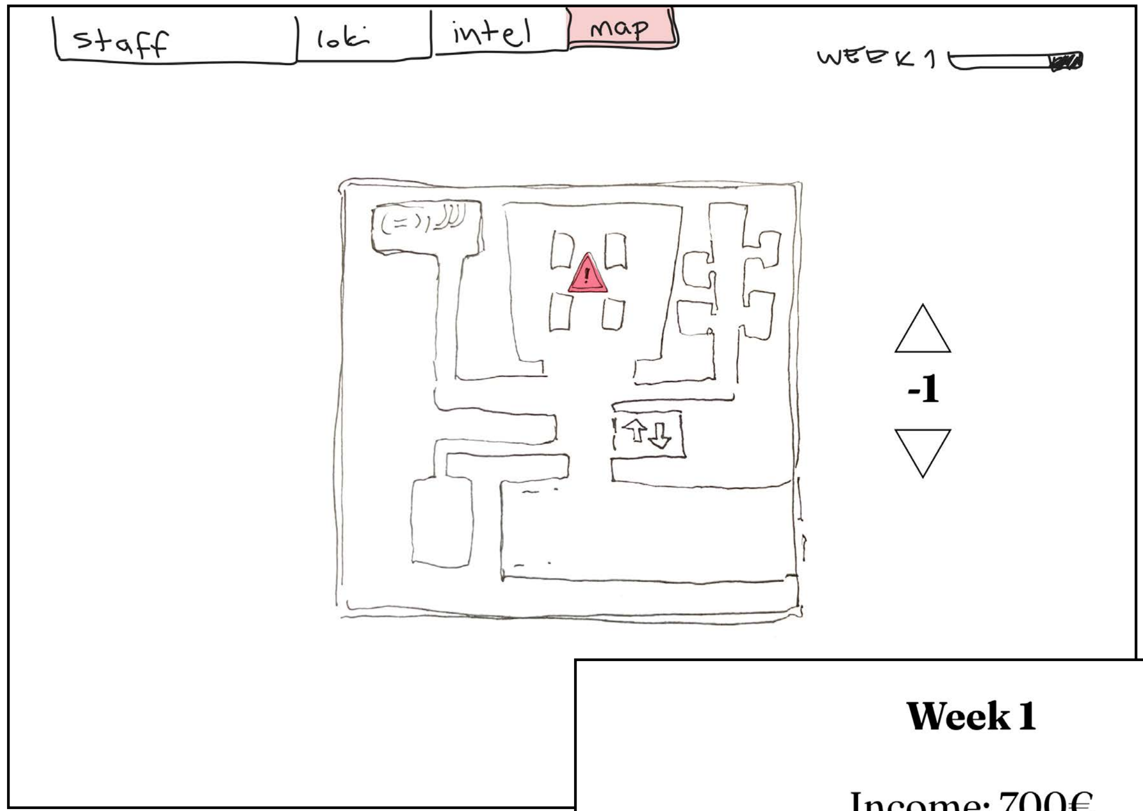
Screenshots of the first Concierge prototype. In it, you go and talk to your patrons, and by learning about them, you can make their rooms a little bit more comfortable or serve them in a more particular manner, giving you the chance to learn even more about them. I tried a system where each customer had three secrets they would reveal about themselves if the quality of the service was on par. I just couldn't figure out how this would tie into actual game mechanics the player interacted with in the game in a lean way.

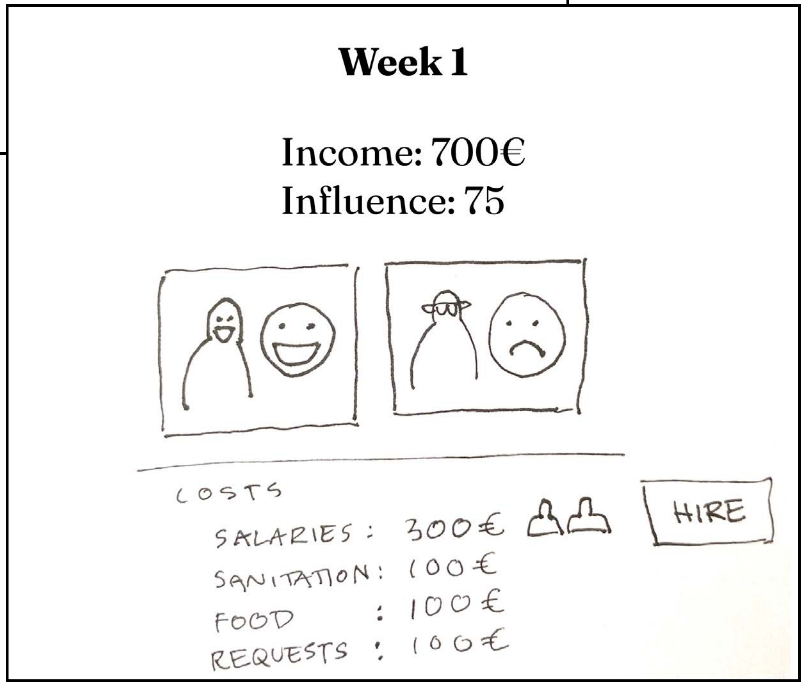


Sketches trying to identify actual gameplay features the player could perform in the game.


Buda Hotel
Happiness of customers
dedication to craft
Santtu
In nursing happens all the time
vs.
Chaos in the background
everything will go wrong
It's not a question whether you want satisfied guests, it's about do you have enough resources
This is potion craft x papers please
point/argument
Comparison to others
Explain for example
Demo
Focus on individual customers and the chain effects of how you treat others
Bear & breakfast
Travellers rest
Chef RPG
COO with no individual regard for customers
Game loop
focus
adaptive story eg how reflected in reviews
Storytelling through mechanics
Kindness is rewarded
Get reservations
Something we need more in today's Society
satisfy their needs while also keeping the hotel running
end of day score

Mindmapping trying to identify what is different in this game compared to its closest rivals. Here I identified that whereas most hospitality tycoon games focus on either running the establishment, or some particular craft, such as with Potioncraft or Chef RPG, I wanted to focus on the individual customers, making them feel like individuals, instead of the precondition for making money.




Week 1

Income: 700€
Influence: 75

COSTS
SALARIES : 300€
SANITATION : 100€
FOOD : 100€
REQUESTS : 100€

Screenshot of the second iteration of Concierge. The game would be played on a map view, with various symbols identifying problems or things to solve in the hotel, whether it was related to customers or the actual infrastructure of the hotel. The player would also be provided with a staff view, telling them who works for them, what their special skills are and how much they are paid. The player could also access a log view, that listed all the possible things that had happened or are happening currently. And finally, they have a view identifying each customer and any particularities about them. This intel view would grow in size the more the player got to know their customers.

At the end of the week, the player would get a summary view, telling them how well they did with the customers living in their hotel that week. Based on the player's performance, they would get a number of influence points, that would affect their star rating and the types of customers that visited their establishment. At the end of the week, they would also need to pay any costs related to running their hotel. These costs and benefits would then inform them on how many customers they needed to take in the week after and what to focus on.

2.2 REFLECTIONS

I had a nagging feeling that while the final game could be interesting, for me as a solo developer there were too many kinds of interactions for me to program instead of only a few mechanics that would combine in interesting ways. This was a clear red flag. This would mean that the development timeline would probably expand uncontrollably. So finally I had to press the breaks and call quits on this iteration of the game. None of the constraints I had set for myself fitted this game concept. Also as the core gameplay was based purely on computationally heavy interactions, instead of something I could prototype on paper, I couldn't get to validating the game unless I developed a hefty bit of code. Seeing that there was no evidence as of yet that the game would work, I opted to rather find a concept I could validate without coding before committing to it. Also, while I could imagine getting a working build of a hotel management game going, I still couldn't figure out mechanics that could simulate the player finding out what their customers wanted without it feeling like an invasion of privacy or otherwise conflicting with social conventions in the real world.

There were too many things wrong with the basic concept of the game and it felt like too much work had to be put into every decision or even trying to imagine what kinds of mechanics it would have. Previously working with games, at this stage, when the right idea has been found, the ideas to broaden or deepen the gameplay come rather naturally. Here that wasn't the case.

There were a few things I did find interesting in this concept. As each customer was procedurally generated from a set of characteristics, each customer would require a different kind of treatment. I wanted to keep that somehow in the later iterations. Also, the fact that each customer is an individual instead of treating them like a mass was interesting to me and I wanted to continue pulling on that thread.

3 CONCIERGE CARDS

The Concierge game turned from a hotel tycoon game into a card game.

3.1 DESIGN PROCESS

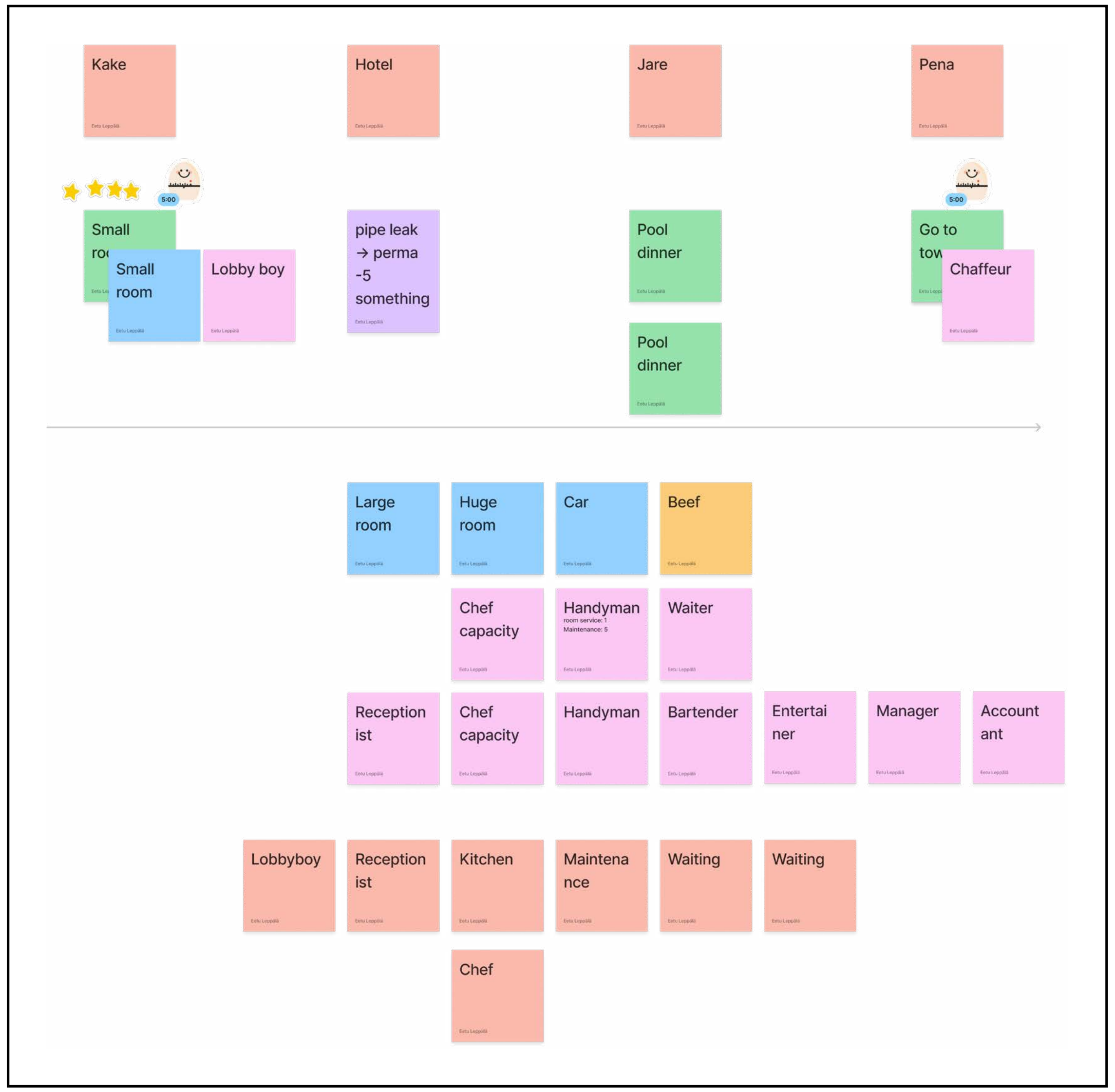
I wasn't ready to give up on Concierge right away. While there was overwhelming evidence that I couldn't get the earlier version to work, at least on my own, I could look into the logic of the game and find out what was the interesting bit of behaviour I wanted to see in the game and discard everything else.

To me it felt like the process of allocating resources to various things depending on the situation was interesting. These were game situations such as: "Do I have the lobby boy carry the bags of the customer to their room, increasing the final score I get from the customer; or do I have them mop the dirty floor, increasing the overall customer satisfaction?" Adding to that, if some of the decisions were in a different category, such as choosing to allocate resources to putting out a literal fire vs. allocating resources to having a pleasant conversation with a customer, it created an interesting tension on what is happening on the surface of the hotel vs what is happening behind the scenes. Customers assume that everything operates smoothly, but that is just because there is a constant battle against the elements hidden from view.

So I decided to try if the game would work purely as a card game. The player would get requests from time to time, appearing on the top of the screen. They would need to allocate the available staff or other resources to those tasks until they were complete by laying resource cards on top of request cards. The resources would be tied to that task until it was completed, leaving a deficit in the player's resource pool for that period. This would create the central tension of the game.

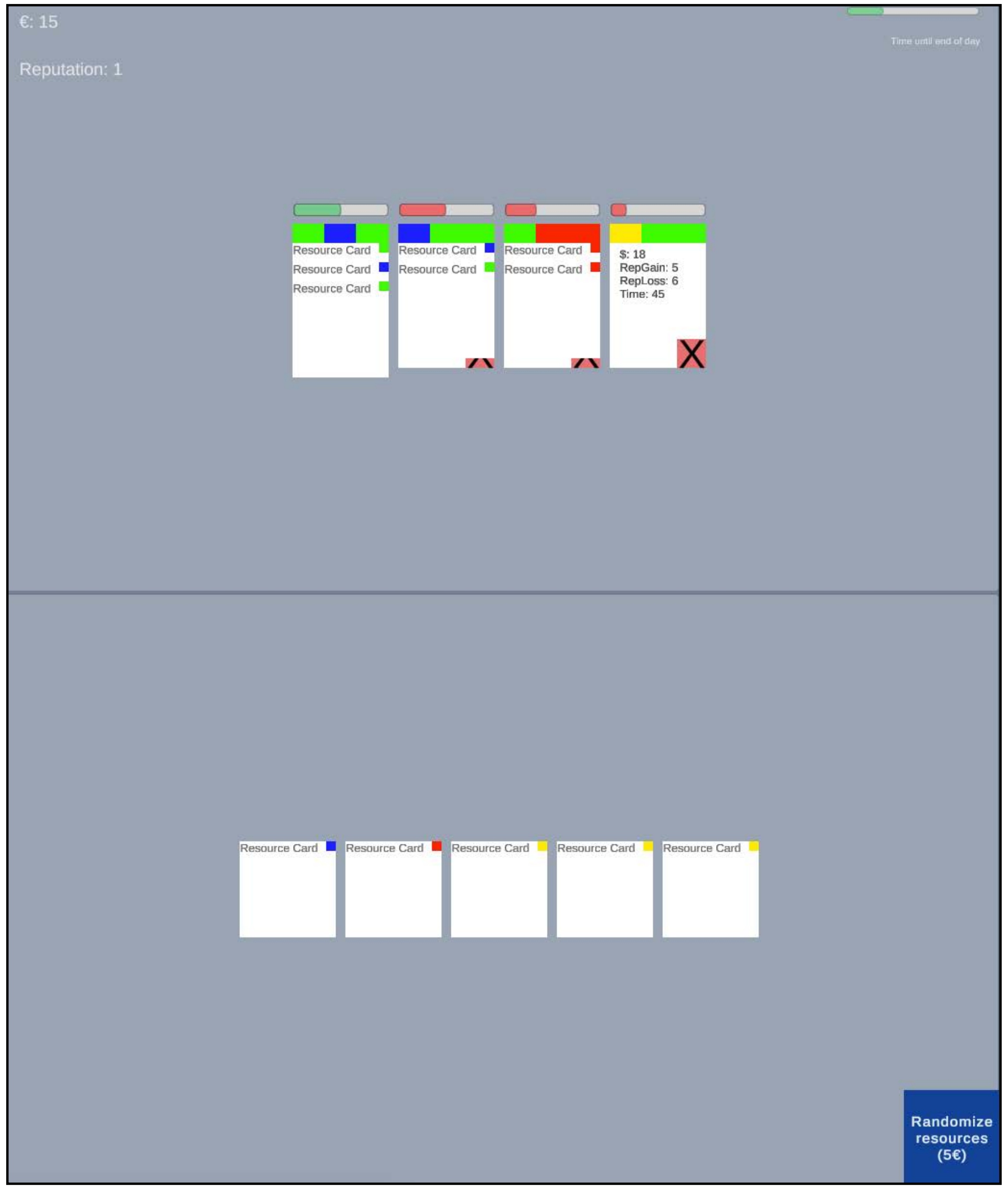
I tried it a few times on Figma, but because I wanted the game to be real-time, with actual timers ticking away above each request, there wasn't any sensible way of simulating this with only a paper proto-type. So I jumped into Unity.

In Unity, I got the basic card pick-up and dropping mechanism to work in no time. Once the requirements of the request were fulfilled, a green timer would start, replacing the red timer indicating when the request would go old. Once the green timer was done, the resources tied to that request would be released, and you would get the amount



Figma prototype simulating fulfilling requests. I had playtesters play this while I acted as the computer, putting new requests to the top of the screen, or tracking whether some request was completed or not. It became quite chaotic quite fast. It also didn't provide me with any indication of whether the gameplay was fun or not.

of money and reputation indicated on the card. This was a preliminary way of differentiating between important and unimportant tasks for the player, creating some decision-making in the otherwise quite linear experience. Once the progress bar, on the top right corner of the screen, indicating the length of one week (the time for which every customer would spend at your establishment) would fill up, the game loop would end, and you would get any bonuses for that week, and you would also get a chance to hire or fire personnel or make other changes to your capacity, such as upgrade your rooms.



Unity prototype implementing the basic logic of fulfilling requests. The link to the GitHub repository is available at the end of this document.

3.2 REFLECTIONS

In the playtesting the gameplay turned out to be too deterministic. There wasn’t actually any decisions to do. Even though I added the “Randomize resources” button in the lower right corner, player would just spam that until they got the right resources they would need, with total disregard to their bank account balance, just because fulfilling the requests and the timers were sort of oppressive and demanded attention. This also went against the metaphor of the game, where each resource card represented some physical being or thing in the real world, so actually you couldn’t just randomise them. I guess while I was working on the prototype, I just wanted to make it work, and didn’t really pay attention to what I was trying to accomplish.

But here, instead of quitting on the project, I think I could have continued a bit further. Because now looking at this after a while, I see potential in it if it was changed to match more the original Figma concept image. I also see that this game could work even as an actual board or solitaire card game, by taking the timer out, and having things happen in turn-order.

When deciding to kill this prototype, I probably got stuck into analysing this version of the game, just because I had developed code for it, and so I couldn’t see myself changing it. For some reason I couldn’t see that changing things would affect anything, so it would be best to kill it early.

All in all, if I had found an interesting mechanic and less deterministic gameplay, this prototype would have met all my requirements for a long-term game project. The card placement code was easy to develop, all other rules and elements in the game would have probably been constraints on where I can place a card and where not. Had I killed the timers, I could have continued development faster with only paper prototyping. I might come back to this later.

4 SPACE SALOON

Space Saloon is another hotel management game, where you cater to your procedurally generated batches of customers by actually changing the layout of your space saloon to match each of your customers' requests in a harmonious balance.

4.1 DESIGN PROCESS

While I was over Concierge, I still believed there was some way of accomplishing this catering to the needs of customers in a way that would meet my constraints. Also, personally, I just wanted to create a game where there would be agents moving on the board, instead of a pure card game. I think I never admitted this to myself until now.

INITIAL DESIGN

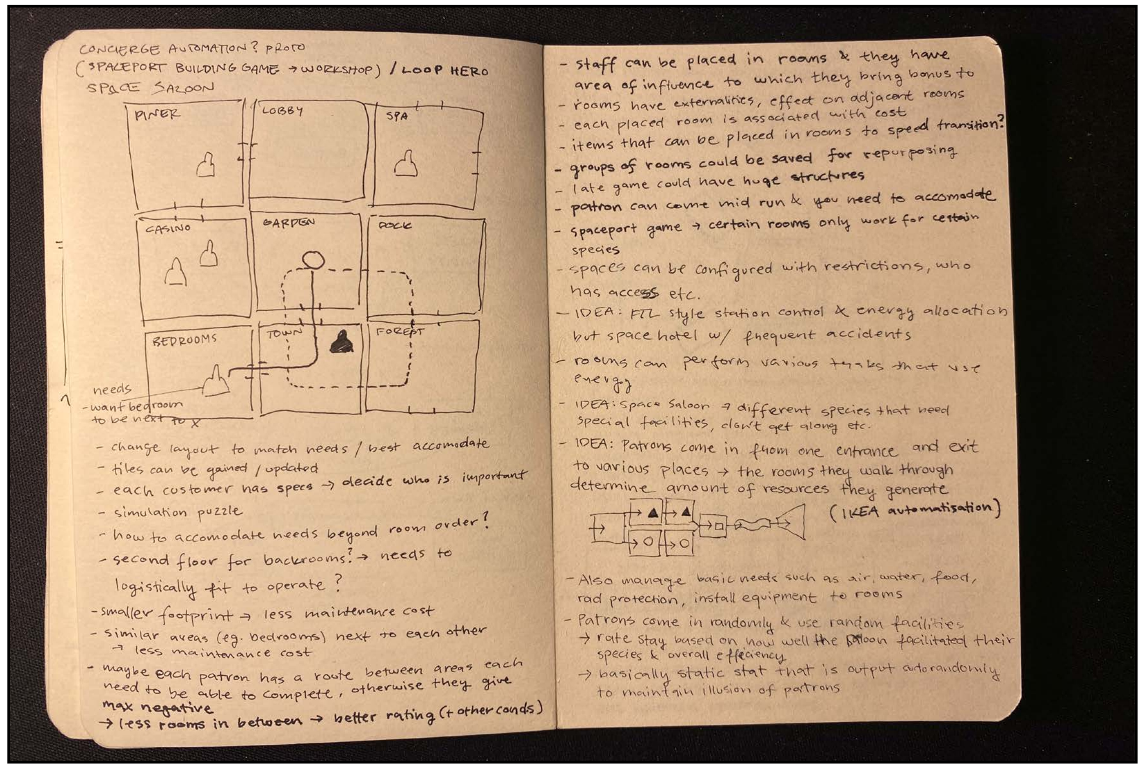
Nevertheless, I got into designing the experience quite quickly. The initial idea for the game came from thinking of the Concierge less from the point of view of the customers and more from the point of view of the hotel as a structure. What if you could change the structure of the entire building to match the needs of the customers. As it didn't make thematially sense to have a hotel that reconfigured itself, I decided to go into a sci-fi direction. (As if that was more plausible) The game then became about putting tiles in the right order to satisfy the customers.

As I had learned from my previous prototypes, I wanted to first create a game I could simulate quite well with paper prototyping before committing any code. The grid structure at least partly allowed that. After nailing down the first iteration of the rules, I went into Figma to create the first playable prototype so I could show it to someone else as soon as possible and get feedback on the very basic interactions.

The rules were as follows:

1. You have a set number of room and bridge pieces you can place on the board.

2. You will get a new batch of customers every turn, while the old customers leave. You need to build a spaceship, that will cater to their specific needs in a way that is not in conflict with the requests of others.


CONCIERGE AUTOMATION? PROTO
(SPACEPORT BUILDING GAME → WORKSHOP) / LOOP HERO
SPACE SALOON

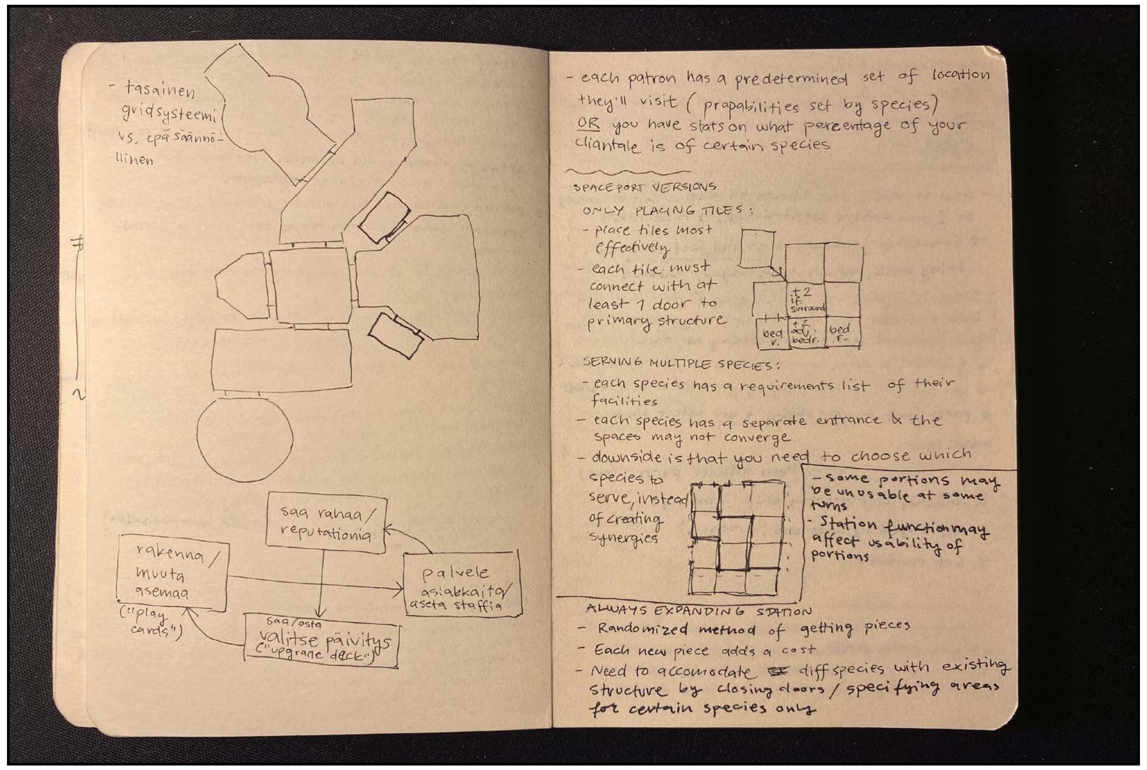
DINER
LOBBY
SPA
CASINO
GARDEN
DOCK
BEDROOMS
TOWN
FOREST

needs
- want bedroom to be next to X

- change layout to match needs / best accomodate
- tiles can be gained / updated
- each customer has specs → decide who is important
- simulation puzzle
- how to accomodate needs beyond room order?
- second floor for backrooms? → needs to logistically fit to operate?
- smaller footprint → less maintenance cost
- similar areas (eg. bedrooms) next to each other → less maintenance cost
- maybe each patron has a route between areas each need to be able to complete, otherwise they give max negative
→ less rooms in between → better rating (+ other conds)

- staff can be placed in rooms & they have area of influence to which they bring bonus to
- rooms have externalities, effect on adjacent rooms
- each placed room is associated with cost
- items that can be placed in rooms to speed transition?
- groups of rooms could be saved for repurposing
- late game could have huge structures
- patron can come mid run & you need to accomodate
- spaceport game → certain rooms only work for certain species
- spaces can be configured with restrictions, who has access etc.
- IDEA: FTL style station control & energy allocation but space hotel w/ frequent accidents
- rooms can perform various tasks that use energy
- IDEA: Space Saloon → different species that need special facilities, don't get along etc.
- IDEA: Patrons come in from one entrance and exit to various places → the rooms they walk through determine amount of resources they generate
(IKEA automatisation)
- Also manage basic needs such as air, water, food, rad protection, install equipment to rooms
- Patrons come in randomly & use random facilities
→ rate stay based on how well the saloon facilitated their species & overall effeciency
→ basically static stat that is output semirandomly to maintain illusion of patrons

First notes on transforming Concierge into a spaceport building game. The first idea was to think of customers as items on a conveyor belt, you could design the route they take and they properties they gain along the way.


- tasainen gridsysteemi vs. epäsäännöllinen

saa rahaa / reputationia
rakenna / muuta asemaa ("play cards")
palvele asiakkaita / aseta staffia
saa/osta valitse päivitys ("upgrade deck")
+2 if surround
bed r.
+2 adj. bedr.
bed r.

- each patron has a predetermined set of location they'll visit (propabilities set by species)
OR you have stats on what percentage of your cliantale is of certain species

SPACEPORT VERSIONS
ONLY PLACING TILES:
- place tiles most effectively
- each tile must connect with at least 1 door to primary structure

SERVING MULTIPLE SPECIES:
- each species has a requirements list of their facilities
- each species has a separate entrance & the spaces may not converge
- downside is that you need to choose which species to serve, instead of creating synergies
- some portions may be unusable at some turns
- Station function may affect usability of portions

ALWAYS EXPANDING STATION
- Randomized method of getting pieces
- Each new piece adds a cost
- Need to accomodate diff species with existing structure by closing doors / specifying areas for certain species only

Concidering whether interesting decisions came from a grid-like structure that would allow all sorts of adjacency effects, or from irregular shapes, such as in Castles of Mad King Ludwig, where the difficulty comes from having right types of spaces for the rooms available.

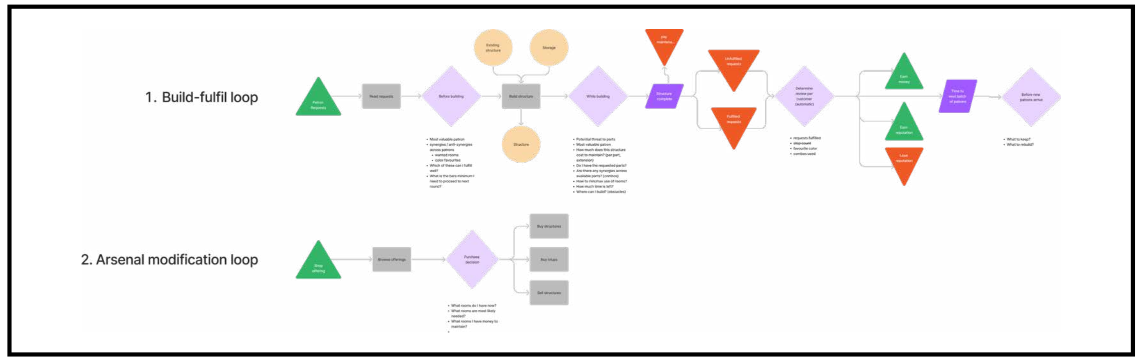
3. A spaceship design is valid when each of the customers can reach their bedroom from the lobby tile, which is in the center.

4. Each tile costs money to use, and the farther out you build from the center, the more expensive it becomes. This encourages, to build compact, multi-purpose stations.

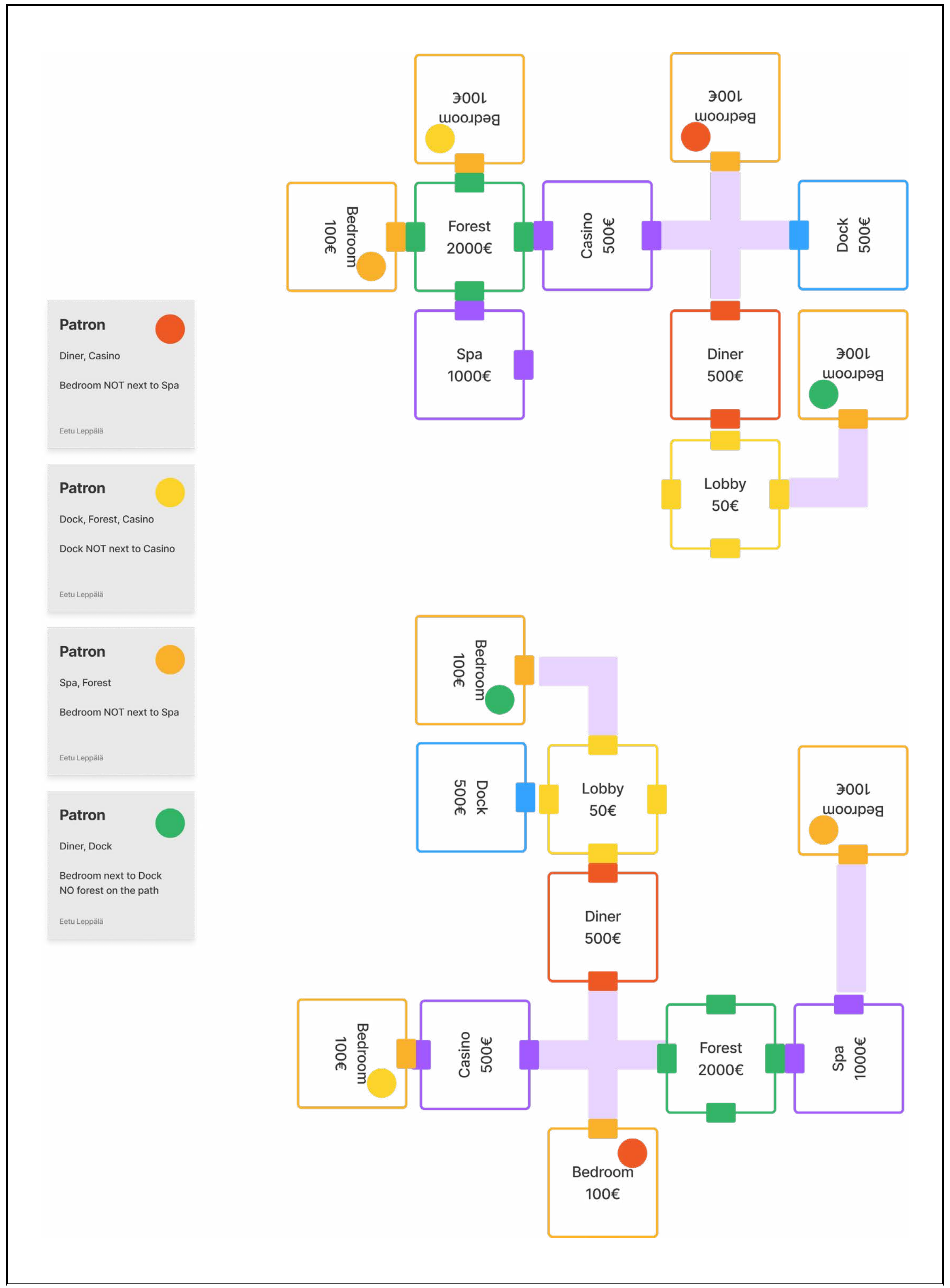
5. You would get points based on how well you were able to accomplish each customer's requests while at the same time minimising the amount of tiles the customer had to traverse in order to get from the lobby to their bedroom.

6. The requests could be something like: "I want my bedroom to be next to the spa"; or "I don't want the restaurant to be next to the docking stations."

While these constraints evoked interesting gameplay, the process of designing a new spaceship became quite stale after only a few tries, because there was usually only one optimum place where to put the tile and it was easy to spot. I added some additional requests that were more like bonuses that needn't be completed and that you could overlook if you were low on cash. This seemed good at least on paper, but once again, the number of variables was becoming too high to handle with a paper prototype. In hindsight, I probably just wanted to get to prgramming instead of trying to figure out how I could continue this development on paper without any code. Looking back, I could have utilised a simple Python script that would produce a deck of tiles that had all the various properties for me. Then it would be only a little manual work to get each hand into the paper prototype.



Me trying to graph out the gameplay loops and possible choices along the way. This was later in the development, where I had a bit more refined idea on how the rooms would exist as a deck of cards in the player's hand.

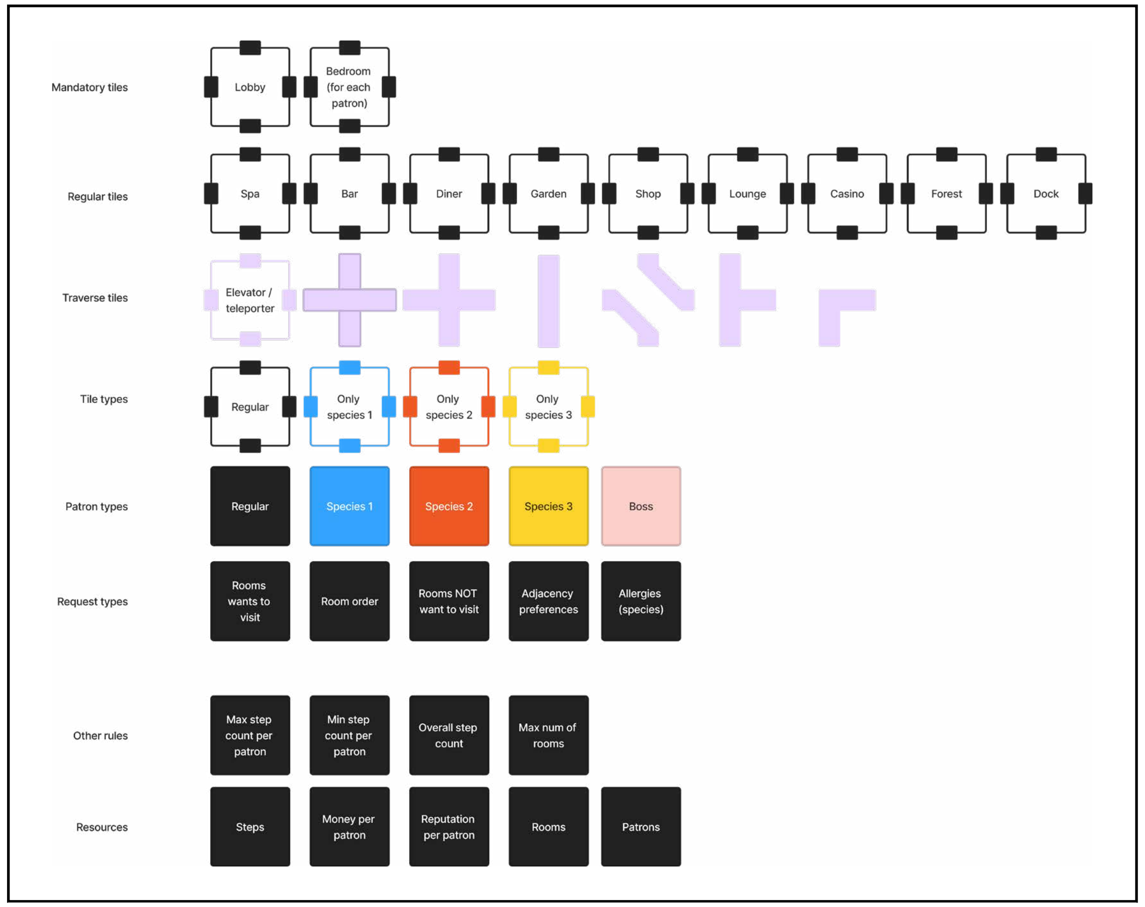

Patron ●
Diner, Casino
Bedroom NOT next to Spa
Eetu Leppälä

Patron ●
Dock, Forest, Casino
Dock NOT next to Casino
Eetu Leppälä

Patron ●
Spa, Forest
Bedroom NOT next to Spa
Eetu Leppälä

Patron ●
Diner, Dock
Bedroom next to Dock
NO forest on the path
Eetu Leppälä

Playtesting the game in Figma. On the left there are the requests of four different patrons or customers, and the graphs are two players' attempts at fulfilling each request. Both succeeded.



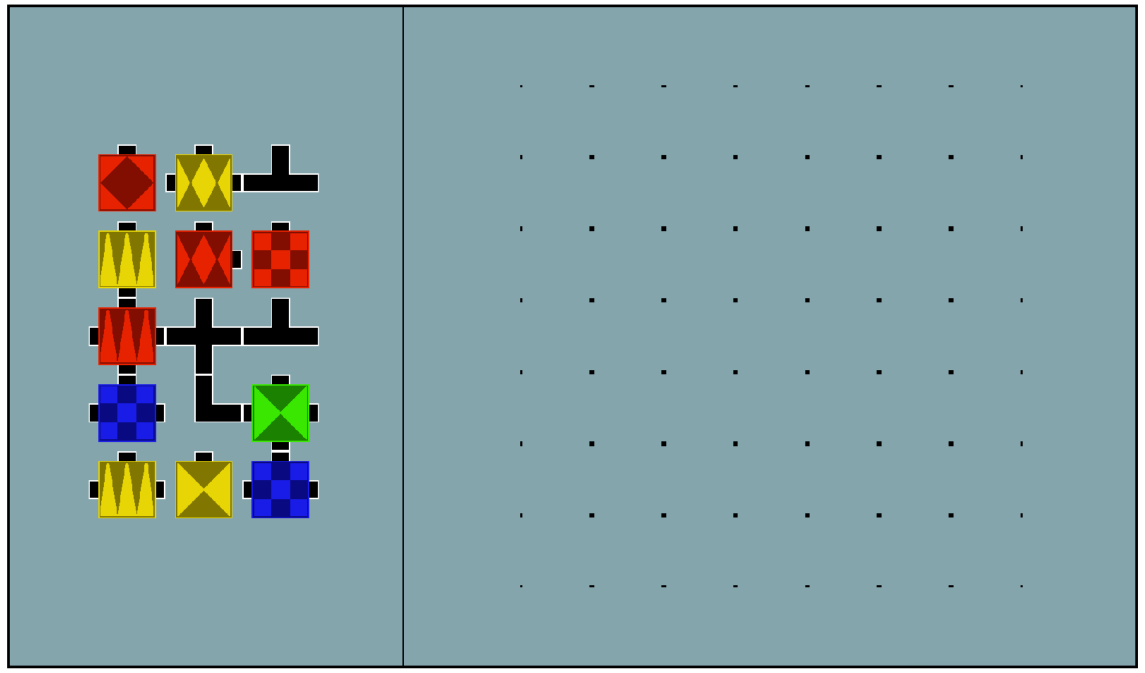
Ideating gameplay content. Here it became clear that to create all the combinations of these elements would be impossible without some form of computation. In hindsight I could have used some form of hybrid prototyping as with later prototypes, adding very little programming where it was needed. (See Moving Manor)

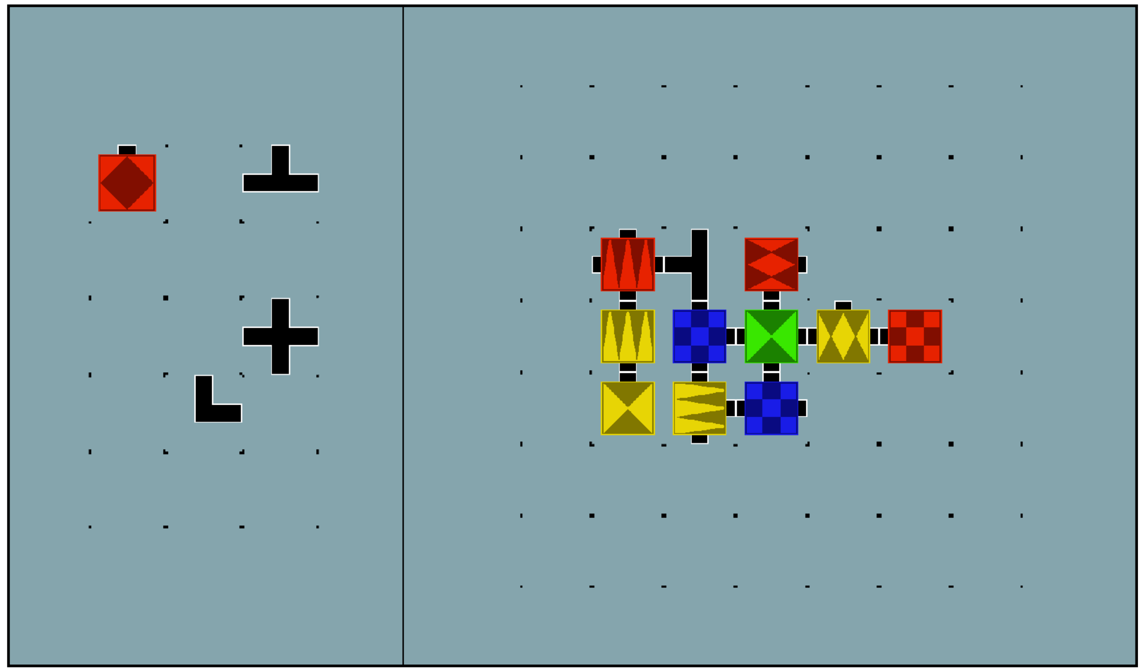
TECH DEMO

For the tech demo, I first wanted to build the very basic interactions to see if only moving tiles around and rotating them would be fun. Also, as I'm sort of a beginner when it comes to programming, I wanted to push myself and went in to develop robust code where there was a separation of the visuals, the controllers and the data.

The problem with this is that while it is a standard way of producing code in C#, the work model doesn't really extend to Unity, where there are game objects and an editor. Also, because I'm quite a novice when it comes to this stuff, I would easily get lost in my own code, and it would take a while every time to know what was my next course of action.

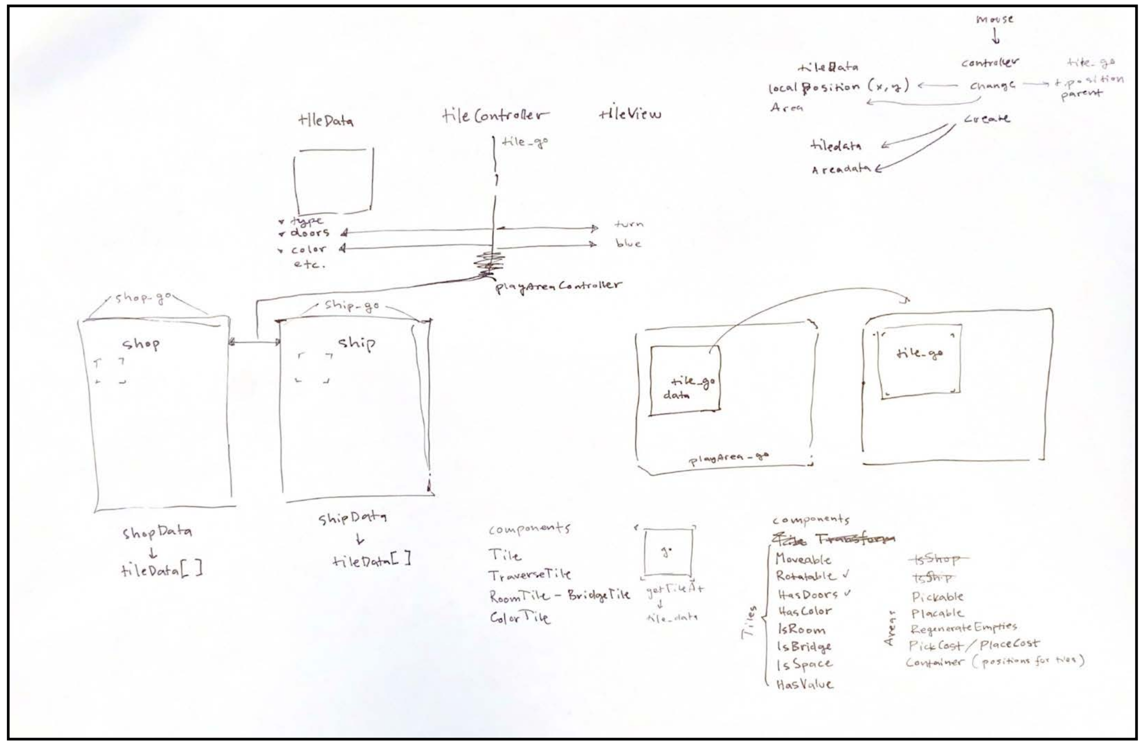
This led to quite a bit of refactoring, where I wanted to convert from one type of code structure to another. Eventually, I read about code being split into smaller single-purpose classes, which makes changing





Screenshots from the working demo. The demo itself worked like a dream, once I got it to work. I could take tiles from the right side of the screen, rotate them and place them into either empty squares, or on top of other tiles, swapping their places. Each tile was able to read what tile was next to them and whether there was access between the two via connected doors.

behaviour in Unity extremely easy, by just loading or offloading scripts onto game objects. So I needed to transform into that style of working in order to guarantee the scalability of the project in the future. I was in the middle of this refactoring when I totally lost fate in the whole endeavour. So I went back to the drawing board.



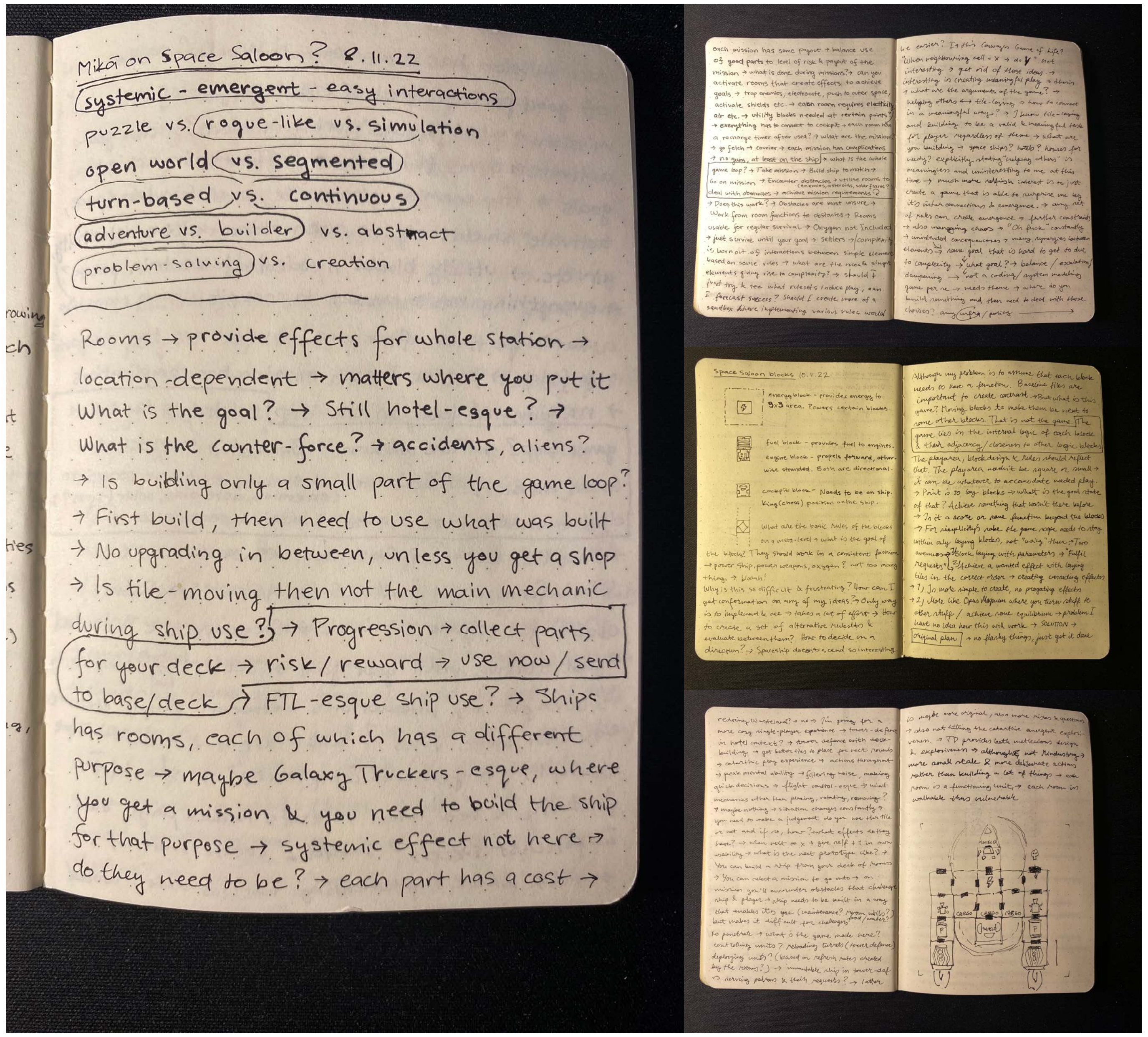
Here I started to analyse how I would have to restructure the code to make it more modular if and when changes in the game design happen. In the lower right corner, I've listed some attributes, each of which would be its own class that could be loaded onto an object depending on the current game state.

ALTERNATIVE PLAY

Once I had finished and refactored the space saloon code multiple times I felt I was sort of trapped. I had a working technical system for moving and attaching tiles on a grid, and on a technical level, it worked quite well. I just didn't have a game. I felt like the prototypes I had worked on in Figma weren't as exciting as they were at first. All-in-all it felt like the whole idea didn't make any sense, not from a thematic point of view nor from an interesting gameplay point of view.

I didn't want to let go of the code I had worked so hard to build, so I decided to stick with it a bit and try to find a game that would fit that constraint.

The first iterations of these explorations were naturally trying to twist the spaceship idea into different forms. First I scrapped the whole idea of caring and being empathetic to your customers. It didn't have anything to do with this game, and I felt it was a bit too much of a challenge for me. Instead, if framed correctly, I could have a



Mikā on Space Saloon? 8.11.22

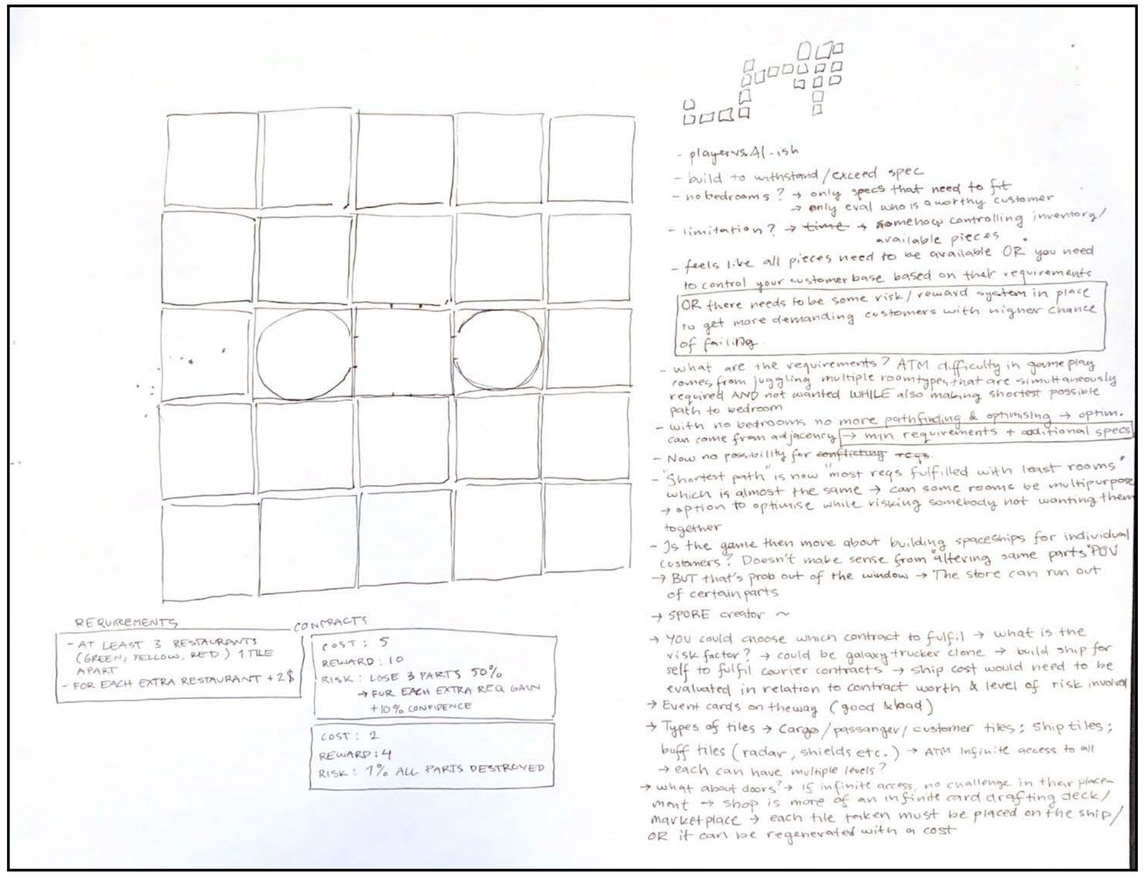
systemic - emergent - easy interactions
puzzle vs. rogue-like vs. simulation
open world vs. segmented
turn-based vs. continuous
adventure vs. builder vs. abstract
problem-solving vs. creation

Rooms → provide effects for whole station → location-dependent → matters where you put it
What is the goal? → Still hotel-esque? →
What is the counter-force? → accidents, aliens?
→ Is building only a small part of the game loop?
→ First build, then need to use what was built
→ No upgrading in between, unless you get a shop
→ Is tile-moving then not the main mechanic during ship use? → Progression → collect parts for your deck → risk / reward → use now / send to base/deck → FTL-esque ship use? → Ships has rooms, each of which has a different purpose → maybe Galaxy Truckers-esque, where you get a mission & you need to build the ship for that purpose → systemic effect not here → do they need to be? → each part has a cost →

Here I tried to write out my thoughts on what I wanted the game to be about, what it was now, and how I could bridge the gap. This technique has proven useful here but also in any other problem-solving context. When I have to spell out my thoughts, logical errors will reveal themselves when analysing the text. I try to keep no filters and instead write stream-of-consciousness writing. Important for me is to just keep moving lest I get stuck on some technicality in the text itself. After this exercise, I will highlight any interesting bits, and carry them over to a separate similar session or on post-its.

game that would teach about utilising complex interactions to the player's favour with all the propagating effects that were now technically possible.

The first proper idea was to copy a bit of what Galaxy Trucker does: Build spaceships to facilitate a courier service of various items



- playervs.AI-ish
- build to withstand / exceed spec
- no bedrooms ? → only specs that need to fit
  → only eval who is a worthy customer
- limitation ? → ~~time~~ → somehow controlling inventory / available pieces
- feels like all pieces need to be available OR you need to control your customer base based on their requirements

OR there needs to be some risk / reward system in place to get more demanding customers with higher chance of failing

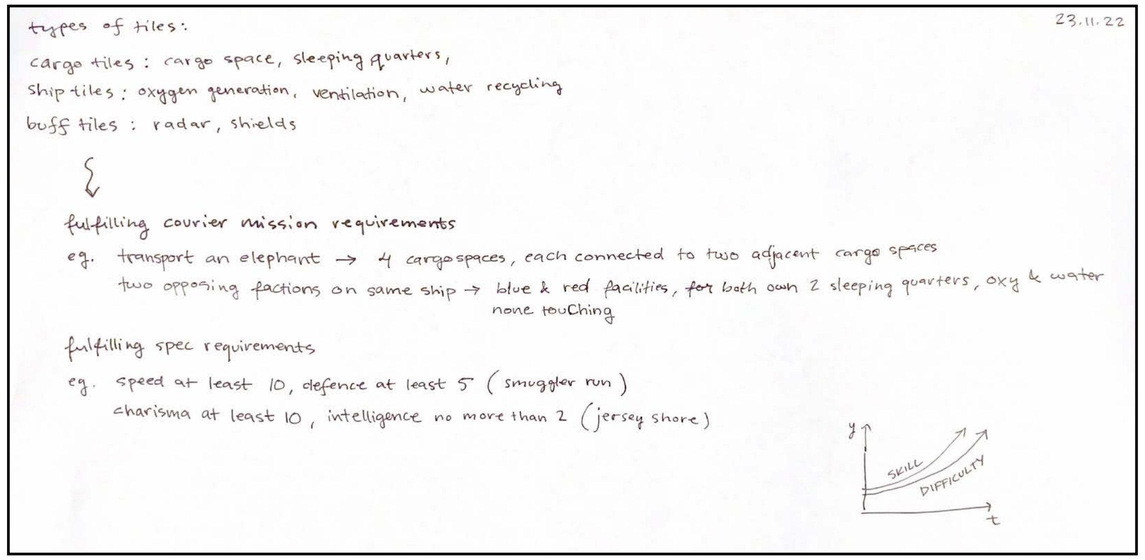
- what are the requirements ? ATM difficulty in gameplay comes from juggling multiple roomtypes that are simultaneously required AND not wanted WHILE also making shortest possible path to bedroom
- with no bedrooms no more pathfinding & optimising → optim. can come from adjacency → min requirements + additional specs
- Now no possibility for ~~conflicting reqs~~
- "Shortest path" is now "most reqs fulfilled with least rooms" which is almost the same → can some rooms be multipurpose → option to optimise while risking somebody not wanting them together
- Is the game then more about building spaceships for individual customers? Doesn't make sense from "altering same parts" POV
→ BUT that's prob out of the window → The store can run out of certain parts
→ SPORE creator ~
→ YOU could choose which contract to fulfil → what is the risk factor? → could be galaxy trucker clone → build ship for self to fulfil courier contracts → ship cost would need to be evaluated in relation to contract worth & level of risk involved
→ Event cards on the way (good & bad)
→ Types of tiles → Cargo / passanger / customer tiles ; ship tiles ; buff tiles (radar, shields etc.) → ATM infinite access to all
→ each can have multiple levels?
→ what about doors? → If infinite access, no challenge in their placement → shop is more of an infinite card drafting deck / marketplace → each tile taken must be placed on the ship / OR it can be regenerated with a cost

REQUIREMENTS
- AT LEAST 3 RESTAURANTS (GREEN, YELLOW, RED) 1 TILE APART
- FOR EACH EXTRA RESTAURANT + 2$

CONTRACTS

COST : 5
REWARD : 10
RISK : LOSE 3 PARTS 50%
→ FOR EACH EXTRA REQ GAIN +10% CONFIDENCE

COST : 2
REWARD : 4
RISK : 1% ALL PARTS DESTROYED



23.11.22

types of tiles:
cargo tiles : cargo space, sleeping quarters,
ship tiles : oxygen generation, ventilation, water recycling
buff tiles : radar, shields

fulfilling courier mission requirements
eg. transport an elephant → 4 cargospaces, each connected to two adjacent cargo spaces
two opposing factions on same ship → blue & red facilities, for both own 2 sleeping quarters, oxy & water none touching

fulfilling spec requirements
eg. speed at least 10, defence at least 5 (smuggler run)
charisma at least 10, intelligence no more than 2 (jersey shore)

Here I tried to formalise my thoughts on the stream-of-consciousness writing exercise into rules and game elements. Yet I still couldn't figure out any specific mechanics that would induce the wanted gameplay.

that require specific conditions eg. space for big items, or special containers for others. This wasn't too far from the original Space Saloon concept but the main focus this time would be on the balance between survivability and making a profit with the operation. Who cared if the cargo was emotionally scarred for life after the transport?

4.2 REFLECTIONS

Looking back at Space Saloon, I feel like there is a game there, but only if all theme is stripped away. I remember struggling to try to justify all the different mechanics and their existence in the game world. This wouldn't be such an issue I think with games that have more traditionally game-like themes, such as dungeon crawling with monsters and wizards, but in the context of the hospitality industry, there is a certain expectation of things functioning somewhat like in the real world. Even though this too can be dressed in a fantasy setting, I felt like with every decision I made regarding the gameplay, I had to make up some new ridiculous explanation of why things work this way in this world. For example, the fact that you can get new pieces of spaceships you can use in your hotel construction was justified by setting the Space Saloon into an asteroid belt, where brave adventurers would accidentally run into asteroids, destroying their vehicles, and those parts would then float to the Space Saloon for later use. Again, there was too much resistance.

But also in general the way customers in the player's saloon were graded was sort of ridiculous. I would really have to forget how the real world worked, in order to accept this is how any system worked. Who expects hotels to reconfigure to match your needs? But despite all of these issues, I just wanted to push forward. A big reason for this, I think, was that I was frustrated with the process and wanted to just commit to something. This led me to waste a lot of time trying to build the Space Saloon system in code. It also locked several consecutive iterations and prototypes into trying to utilise this code.

I had somehow fooled myself into thinking that the game cannot be prototyped in paper prototype format. But as I've learned by now, this is an immediate red flag and premature coding should be avoided like the plague. At least in my case.

In terms of the technical demo, the blunders I made in coding are part of the cycle of trying to learn software development, I guess. What could have been beneficial was to ask a real developer beforehand these sorts of fundamental questions. The problem with that is that the questions cannot be asked before there have been problems arising from mistakes. I couldn't have known to ask that question before I realised the structure of the code will block any future progress. In other words, more experience is needed.

In the last iteration, I think things sort of unravelled. I was trying to push for an idea when in reality I should have just let the game project rest for a while. Looking back it felt forced, similar to Concierge. Even though I could see the spaceship and cargo hauling game being played in my mind's eye, I couldn't figure out the foundational truths about the game everything else would be built on. It's almost like trying to describe a dream.

I've found later that when approaching a game idea from a very logical and linear point of view, trying to describe the game in its narrowest manifestation, things should still make sense for the game to be built on solid ground. If the game requires a ton of content and rules to even approach the idea I had, the probability of it working is close to zero. With the starship idea, I think this was the case. I couldn't describe what the core components were or what the narrowest possible experience would be like or whether it would be fun.

5 HEGEMONY

Hegemony is a single-player mobile game experience based on a voting board game I developed for my graduate thesis project. It was an attempt at a palette cleanser and a complete change in direction from the earlier game concepts.

5.1 DESIGN PROCESS

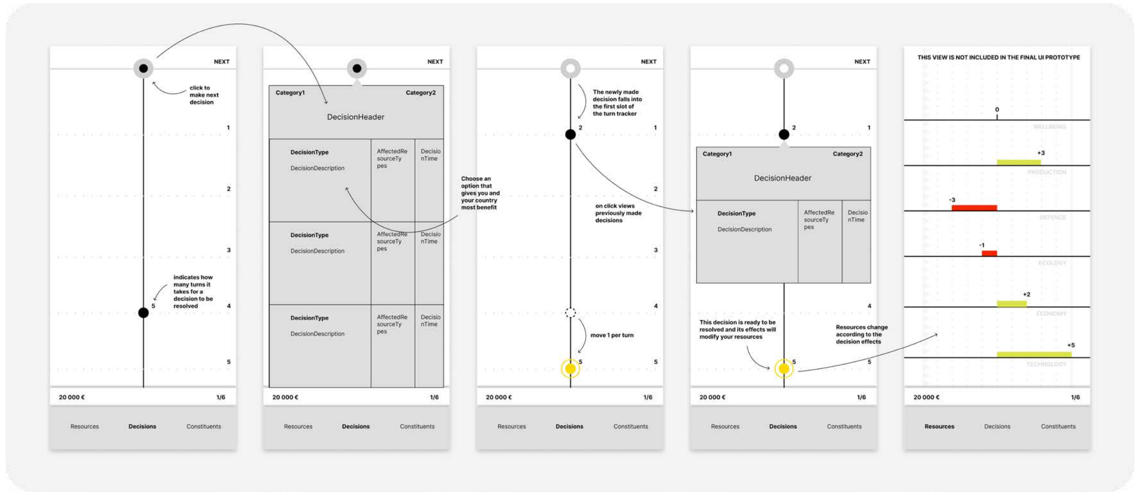
I came back to the games I had developed in my graduate thesis project basically just to get inspired. While they were similarly painful in their design process, they were now more or less finished and play-able games. I wondered if I could transform one of them into a single-player experience. This led to Hegemony, a game where you rule over a nation, making decisions while keeping your constituents happy, and resources in balance.

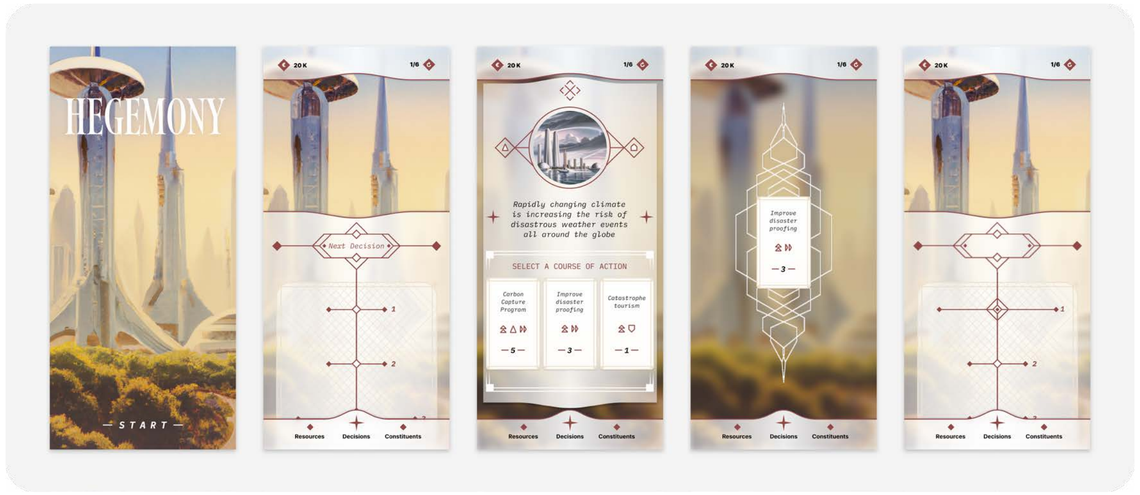
This wasn't necessarily a serious attempt at making a long-term game project, but rather looking at an existing design, and spending a little bit of time in other phases of game development. Often the most laborious part of any design project is the beginning. As all my attempts thus far had been stopped short, I wanted to taste the rela-tive ease at which decisions can be made in the later stages and in other design tasks such as UI.

So I designed a few screens for the imaginary mobile game, as well as some transition animations.

5.2 REFLECTIONS

I guess it was good I did this project because it gave my brain a chance to rest a bit and experience a fuller picture of game devel-opment. Often times the brain ties itself into knots over time if no progress is made. It can create pockets of self-doubt and shake self-confidence. This bit of foolishness gave me chance to enjoy whole-heartedly something I know relatively well. I guess this also gave me time to process all that had been going on with earlier proto-types and mentally let go of some ideas that were nagging and look at the whole endeavour from a fresh, more objective perspective.





HEGEMONY
START
Next Decision
Rapidly changing climate is increasing the risk of disastrous weather events all around the globe
SELECT A COURSE OF ACTION
Carbon Capture Program
Improve disaster proofing
Catastrophe tourism
Resources
Decisions
Constituents

Some of the screens developed for Hegemony. I first created a basic UX flow and then started to refine them, of course working through the general look of the game.

6 MOVING MANOR

Moving Manor is a rogue-lite deckbuilding puzzler, where the player controls an adventurer who is trying to find their way out of the moving maze-like corridors of an archmage's floating abode hidden within the Astral Sea. The player progresses through the game by successfully navigating their way through rooms filled with monsters and treasure by utilising a deck of cards through which they can manipulate the very environment around them. Various denizens of the majestic halls, mostly former adventurers, who have now given up hopes of escaping, are willing to improve your chances of escaping by upgrading your cards in exchange for treasure.

6.1 DESIGN PROCESS

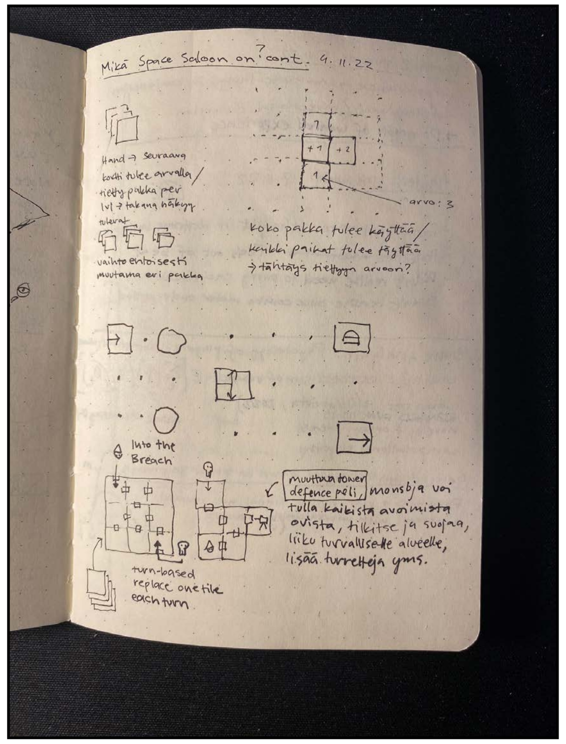
Looking back at my notes, I had been scribbling little doodles of a dungeon crawler gameplay many times in the corners of my notes, but I had always pushed that away. Once I actually gave it a chance and thought about what kind of gameplay would be possible, I immediately started getting excited. It was a clear sign of something clicking in place, something that had not happened to this extent earlier during this project.

PUZZLE GAME

My initial reaction was to start with as small as possible. I was constantly trying to design in such directions where I didn't have to involve programming. I wanted something a little janky, reminicient of old RPGs where movement is turn-based, giving me as the designer more control.

The first idea came literally from mashing the two previous concepts together: dungeon crawling on a grid, where tiles can be moved and rotated. The first iteration reminded me actually quite a bit of The Moving Labyrinth. The goal of the player is to go through a squence of rooms, all the while avoiding monsters and collecting loot by rotating a single tile per turn or tile of movement. Each turn the enemies would also try to reach you and kill you. The design process was very controlled and grew from this small seedling.

The game initially took the form of a puzzle game. I would build these rooms with a solving strategy in mind. But after only a few games playing it, it turned out there isn't too many possiblities on a 4x4 grid. So grid size got increased.



Mikä Space Saloon on? cont. 4.11.22

Hand → seuraava kortti tulee arvalla / tietty pakka per lvl → takana näkyy tulevat

vaihtoehtoisesti muutama eri pakka

+1 +2 1 ?

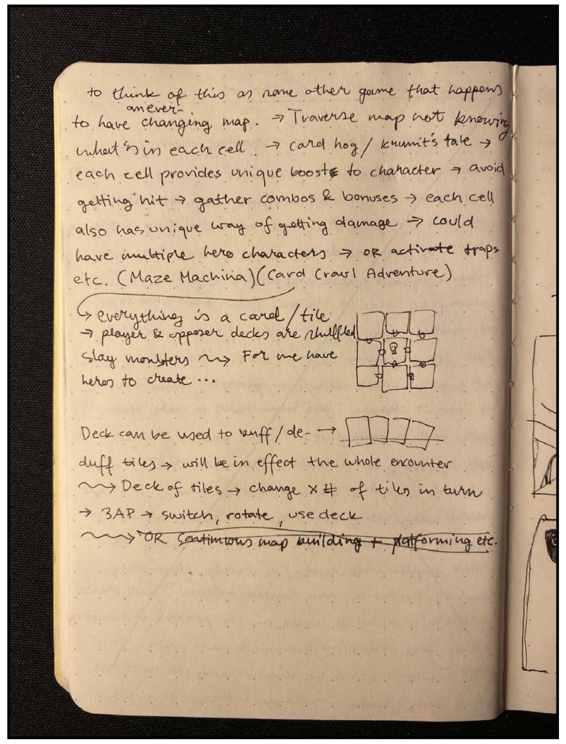
arvo: 3

koko pakka tulee käyttää / kaikki paikat tulee täyttää → tähtäys tiettyyn arvoon?

Into the Breach

muutama tower defence peli, monsuja voi tulla kaikista avoimista ovista, tilkitse ja suojaa, liiku turvalliselle alueelle, lisää turretteja yms.

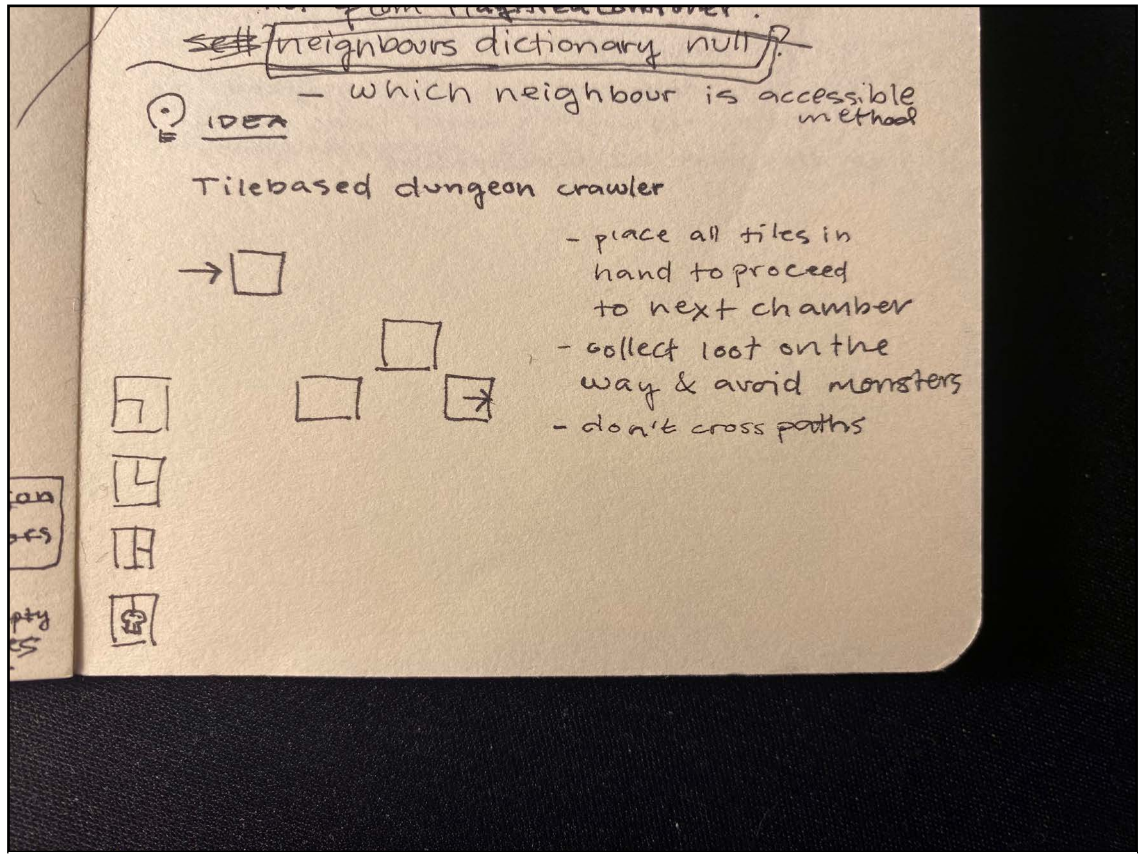
turn-based replace one tile each turn



to think of this as rome other game that happens to have an ever-changing map. → Traverse map not knowing what's in each cell → card hog / kuunit's tale → each cell provides unique boosts to character → avoid getting hit → gather combos & bonuses → each cell also has unique way of getting damage → could have multiple hero characters → OR activate traps etc. (Maze Machina)(Card Crawl Adventure)

→ everything is a card / tile
→ player & opposer decks are shuffled
slay monsters → For me have heros to create ...

Deck can be used to buff / de- → duff tiles → will be in effect the whole encounter
→ Deck of tiles → change x # of tiles in turn
→ 3AP → switch, rotate, use deck
→ OR ~~Continuous map building + platforming etc.~~



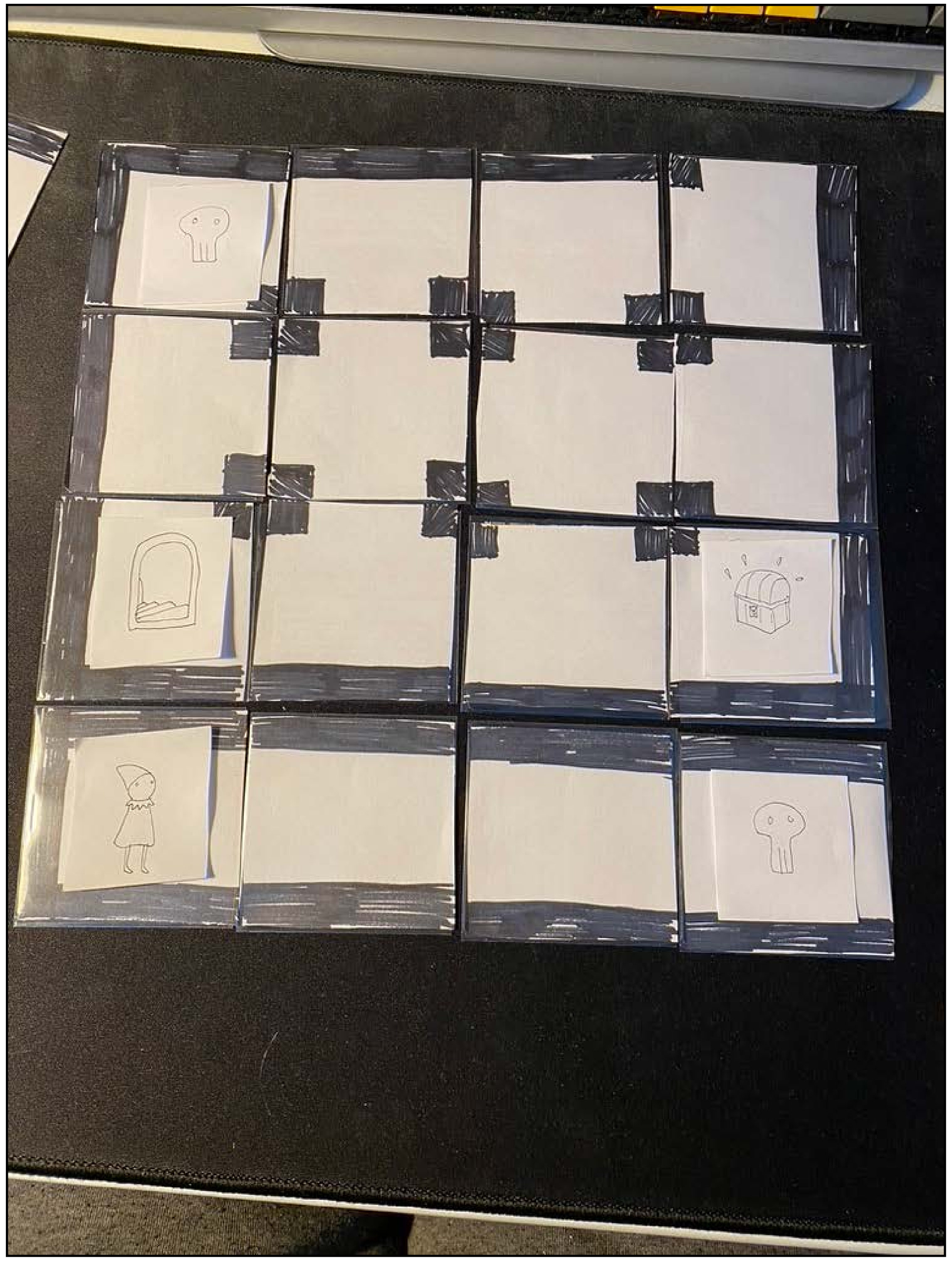
~~self~~ neighbours dictionary null?
– which neighbour is accessible method

IDEA

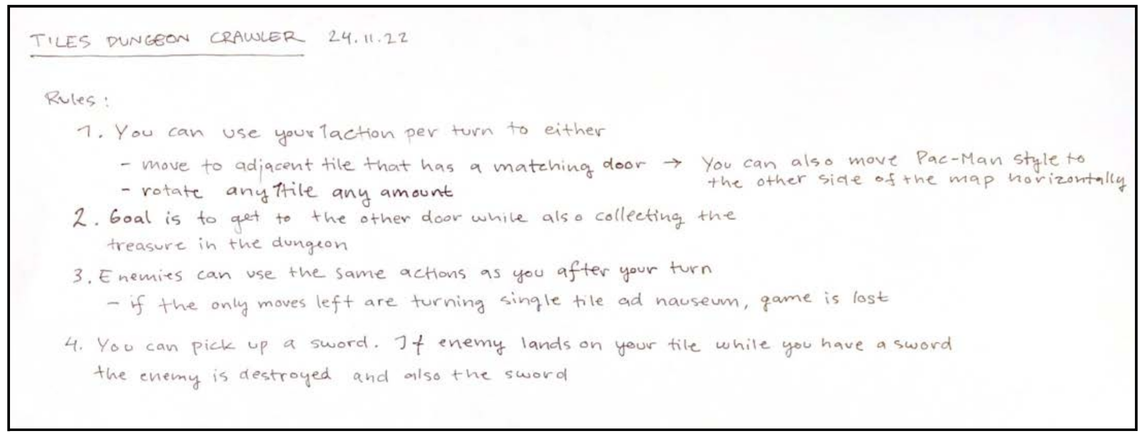
Tilebased dungeon crawler

– place all tiles in hand to proceed to next chamber
– collect loot on the way & avoid monsters
– don't cross paths

Notes I had scribbled in-between other prototype notes. There was clearly something that inspired me in this concept, because I kept getting back to it repeatedly.



One of the first puzzles for Moving Manor. While this more puzzler version of the game required meticulous placement of the tiles on my part, I would have liked to create an algorithm that would place the tiles for me, following particular distribution rules, thus increasing the potential of the game to new heights. As it turned out, the puzzle-ness wasn't what I wanted, so I pivoted to a rogue-lite deckbuilder quite quickly.



TILES DUNGEON CRAWLER 24.11.22

Rules:

1. You can use your 1action per turn to either
   - move to adjacent tile that has a matching door → You can also move Pac-Man style to the other side of the map horizontally
   - rotate any 1tile any amount
2. Goal is to get to the other door while also collecting the treasure in the dungeon
3. Enemies can use the same actions as you after your turn
   - if the only moves left are turning single tile ad nauseum, game is lost
4. You can pick up a sword. If enemy lands on your tile while you have a sword the enemy is destroyed and also the sword

First rule set I used to playtest the idea on a bunch of people. It laid the foundation for the rest of the game. This game already felt fun, and easy to grasp even for novice players, which was a good sign.

While the game was fun and dead simple, it was still missing the strategic depth I was hoping for, some element that would increase the depth of the game while keeping it technically still very elegant. So very quickly I decided to add deck building into the game.

6.2 DECK-BUILDER

This is where I first noticed markings of something I could commit to. I had already had a blast with playing the game, but adding a deck-building aspect to the game made it sort of organic, and something I as a designer could uncover, instead of me trying to shoehorn things in.

Here are the current rules for the game:

GOAL

- Player must successfully navigate through the various ever-changing floors and find the final exit out of the mage's castle.

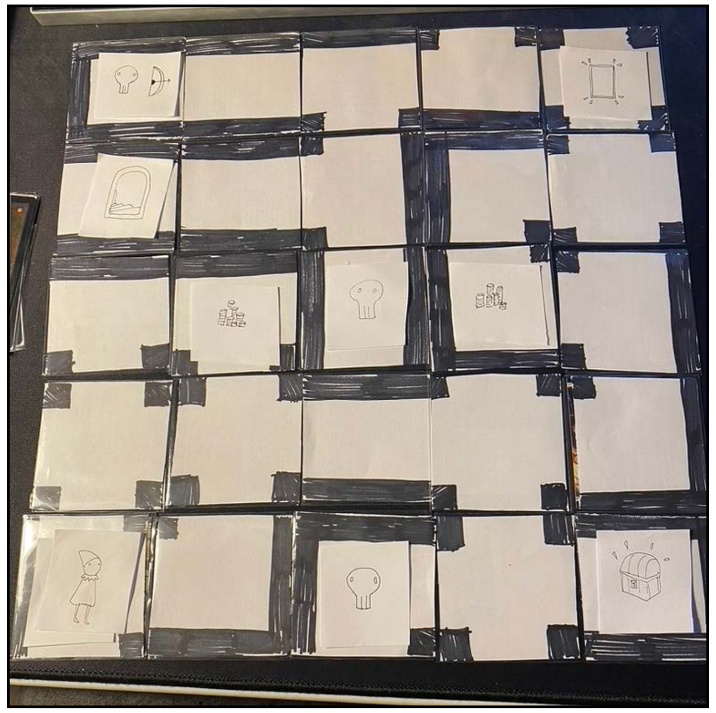
ROUNDS AND TURNS

- Each floor consists of procedurally placed tiles of varying type on a 7 by 7 grid.
- The floor has successfully been completed when the player enters the exit stairs tile. On their way to the exit, player may attempt to collect treasure placed on the floor while avoiding enemies by playing cards from their hand.

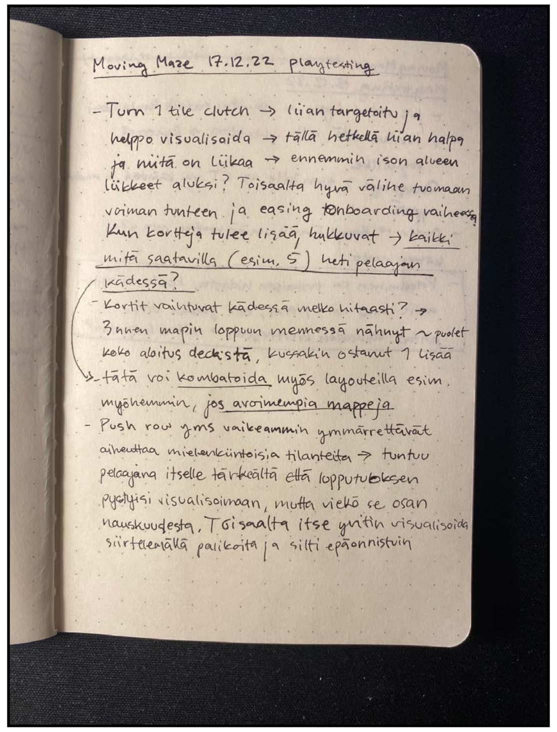
- The game is played in turns. Each turn as the player has taken their turn, the enemies and other objects in the floor take their turn.

  1. The player has the opportunity to play cards from their hand or spend Action Points (AP) to draw cards, this can be skipped.
  2. The player moves.
  3. Enemies attack if possible.
  4. Enemies move.
  5. REPEAT from 1.

- Both the player and the enemies will see the entire map and each opponent unless stated otherwise.
- The game will end if the player runs out of health points (HP) or if they have no viable path to the exit. (compare to a stalemate in chess) This can also happen if the player accidentally traps themselves inside the maze by moving the tiles in such a way as to block their access to the exit tile.

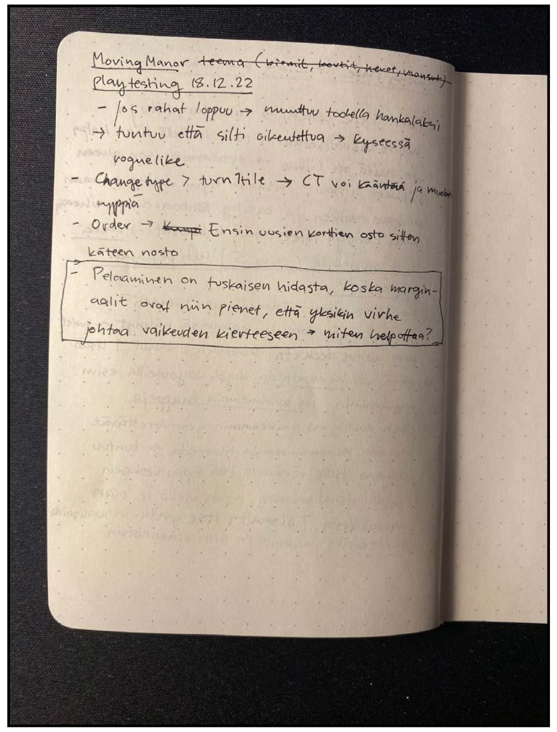


The first iterations of the game were actually done on paper this time. I felt like I was on a slippery slope if I involved computers at all in the beginning phases, as in the earlier prototypes it had led to me quickly going into Unity to prototype, which had turned out to be wasted time.



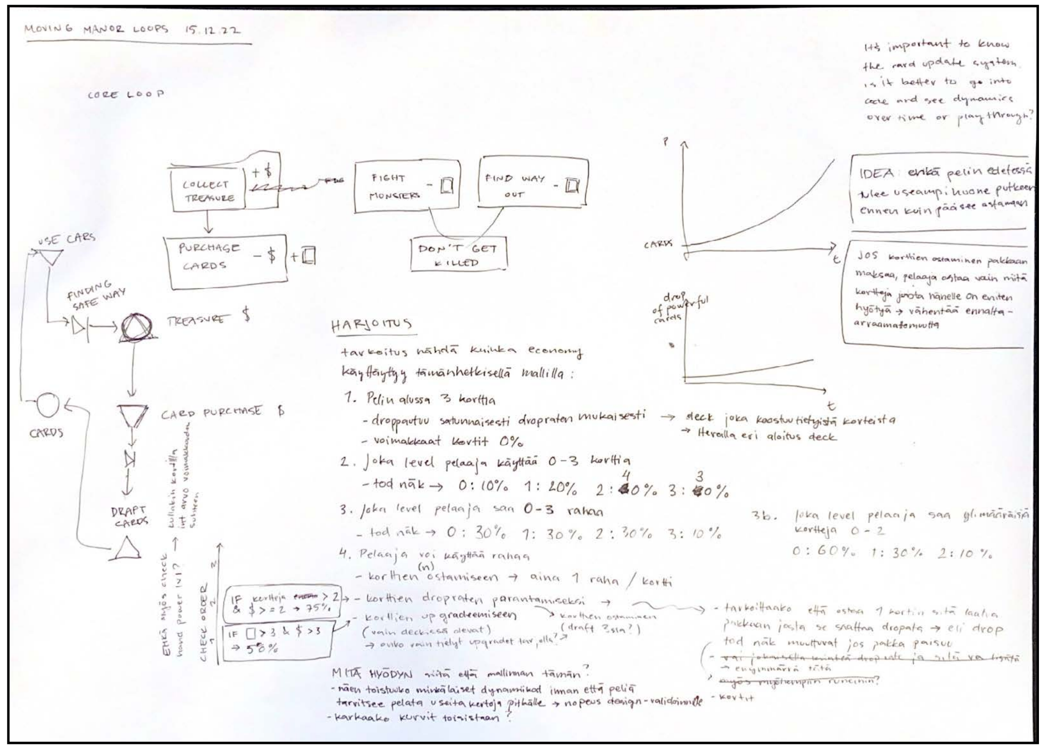
Moving Maze 17.12.22 playtesting

- Turn 1 tile clutch → liian targetoitu ja helppo visualisoida → tällä hetkellä liian halpa ja niitä on liikaa → ennemmin ison alueen liikkeet aluksi? Toisaalta hyvä väline tuomaan voiman tunteen ja easing ~~t~~onboarding vaiheessa
Kun kortteja tulee lisää, hukkuvat → kaikki mitä saatavilla (esim, 5) heti pelaajan kädessä?
- Kortit vaihtuvat kädessä melko hitaasti? → 3nnen mapin loppuun mennessä nähnyt ~ puolet koko aloitus deckistä, kussakin ostanut 1 lisää
- tätä voi kombatoida myös layouteilla esim. myöhemmin, jos avoimempia mappeja
- Push row yms vaikeammin ymmärrettävät aiheuttaa mielenkiintoisia tilanteita → tuntuu pelaajana itselle tärkeältä että lopputuloksen pystyisi visualisoimaan, mutta viekö se osan nauskuudesta. Toisaalta itse yritin visualisoida siirtelemällä palikoita ja silti epäonnistuin



Moving Manor ~~teema (biomit, kortit, herot, monsut)~~
playtesting 18.12.22
- Jos rahat loppuu → muuttuu todella hankalaksi → tuntuu että silti oikeutettua → kyseessä roguelike
- Change type > turn 1 tile → CT voi ~~kääntää~~ ja muokata tyyppiä
- Order → ~~kauppa~~ Ensin uusien korttien osto sitten käteen nosto
- Pelaaminen on tuskaisen hidasta, koska marginaalit ovat niin pienet, että yksikin virhe johtaa vaikeuden kierteeseen → miten helpottaa?

Initial notes on the first versions of the game. As always, I was struggling to answer the key questions about the game I was making: Do I have cards in this game? How powerful they are? Are there Action Points you need to expend in order to use the cards? How many cards do you have in hand? Thankfully I was able to answer these questions with a simple playtesting session, as the game was so easy to set up and test on paper.



It's important to know the card update system
is it better to go into code and see dynamics over time or play through?

IDEA: ennä pelin edetessä tulee useampi huone putkeen ennen kuin pääsee ostamaan

JOS korttien ostaminen pakkaan maksaa, pelaaja ostaa vain niitä kortteja joista hänelle on eniten hyötyä → vähentää ennalta-arvaamattomuutta

HARJOITUS

tarkoitus nähdä kuinka economy käyttäytyy tämänhetkisellä mallilla:

1. Pelin alussa 3 korttia
   - droppautuu satunnaisesti dropraten mukaisesti → deck joka koostuu tietyistä korteista → Heroilla eri aloitus deck
   - voimakkaat kortit 0%
2. Joka level pelaaja käyttää 0–3 korttia
   - tod näk → 0: 10% 1: 20% 2: 40% 3: 30%
3. Joka level pelaaja saa 0–3 rahaa
   - tod näk → 0: 30% 1: 30% 2: 30% 3: 10%

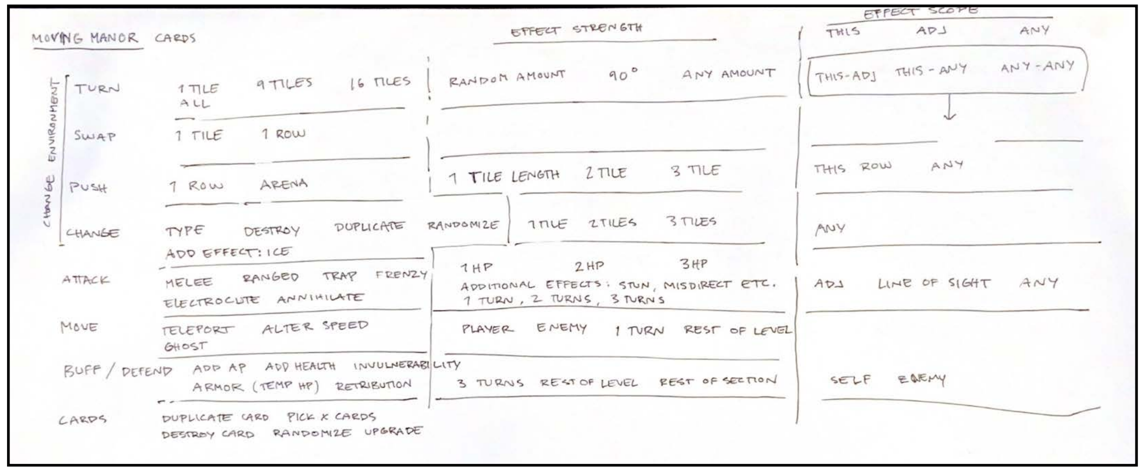
3b. Joka level pelaaja saa glimääräisiä kortteja 0–2
0: 60% 1: 30% 2: 10%

4. Pelaaja voi käyttää rahaa (n)
   - korttien ostamiseen → aina 1 raha / kortti
   - korttien dropraten parantamiseksi → → korttien ostaminen (draft 3sta?)
   - korttien upgradeemiseen (vain deckissä olevat) → onko vain tietyt upgradet tarjolla?

- tarkoittaako että ostaa 1 kortin siitä laatua pakkaan josta se saattaa dropata → eli drop tod näk muuttuvat jos pakka paisuu
- ~~vai jokaisella kortilla drop rate ja siitä voi lisätä~~ → enyimmärrä tätä
~~myös myöhempiin runeihin?~~
- kortit

MITÄ HYÖDYN siitä että mallinnan tämän?
- näen toistuuko minkälaiset dynamiikat ilman että peliä tarvitsee pelata useita kertoja pitkälle → nopeus design-validoinnille
- karkaako kurvit toisistaan?

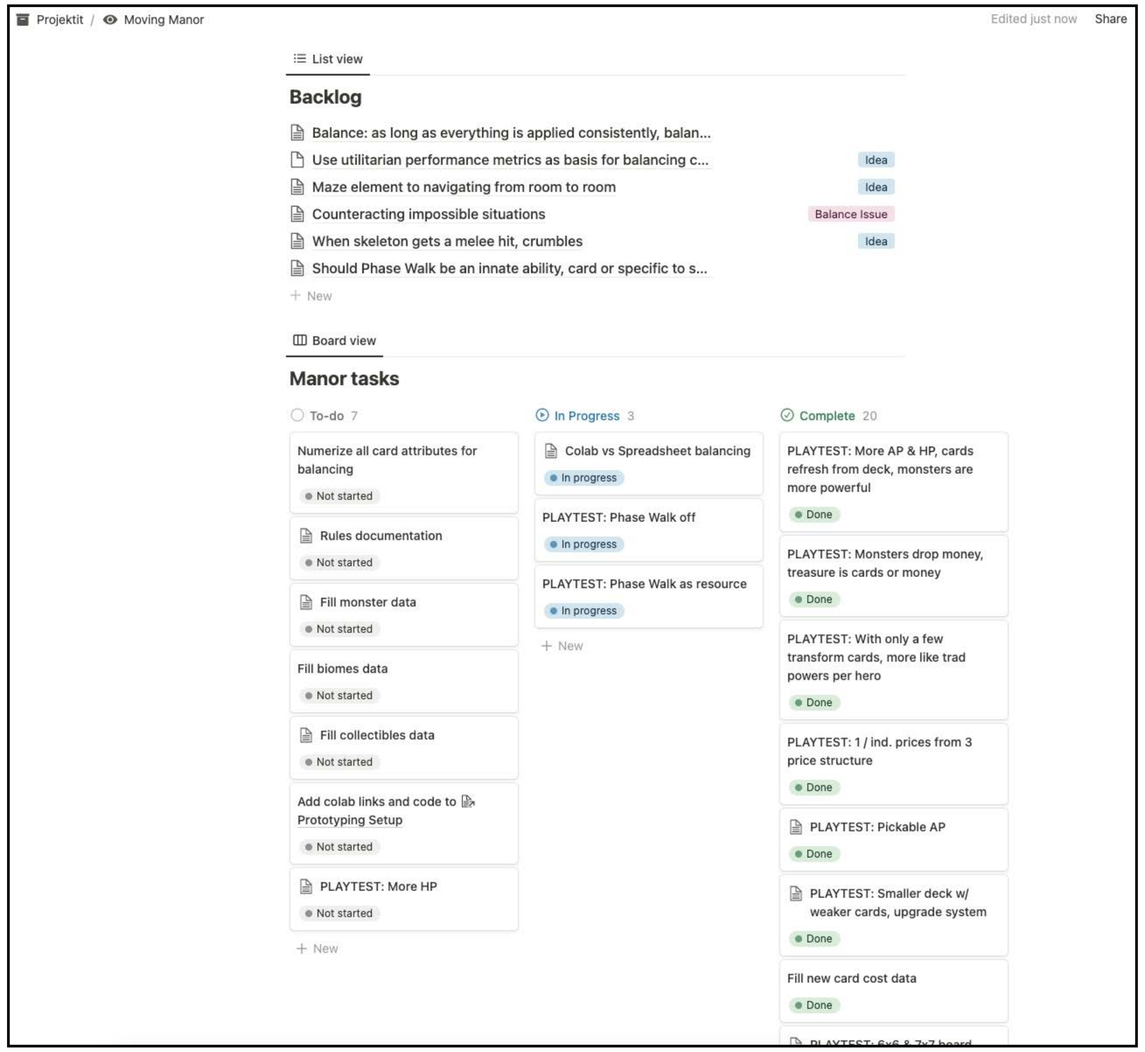
Attempt to formalise some of the more numbers-heavy elements of the game.



MOVING MANOR CARDS

			EFFECT STRENGTH	EFFECT SCOPE: THIS / ADJ / ANY
CHANGE ENVIRONMENT	TURN	1 TILE 9 TILES 16 TILES ALL	RANDOM AMOUNT 90° ANY AMOUNT	THIS-ADJ THIS-ANY ANY-ANY
	SWAP	1 TILE 1 ROW		
	PUSH	1 ROW ARENA	1 TILE LENGTH 2 TILE 3 TILE	THIS ROW ANY
	CHANGE	TYPE DESTROY DUPLICATE RANDOMIZE ADD EFFECT: ICE	1 TILE 2 TILES 3 TILES	ANY
	ATTACK	MELEE RANGED TRAP FRENZY ELECTROCUTE ANNIHILATE	1HP 2HP 3HP ADDITIONAL EFFECTS: STUN, MISDIRECT ETC. 1 TURN, 2 TURNS, 3 TURNS	ADJ LINE OF SIGHT ANY
	MOVE	TELEPORT ALTER SPEED GHOST	PLAYER ENEMY 1 TURN REST OF LEVEL	
	BUFF / DEFEND	ADD AP ADD HEALTH INVULNERABILITY ARMOR (TEMP HP) RETRIBUTION	3 TURNS REST OF LEVEL REST OF SECTION	SELF ENEMY
	CARDS	DUPLICATE CARD PICK X CARDS DESTROY CARD RANDOMIZE UPGRADE		

Ideating card effects by revealing all possible combinations.



Projektit / Moving Manor — Edited just now — Share

List view

Backlog

- Balance: as long as everything is applied consistently, balan...
- Use utilitarian performance metrics as basis for balancing c... — Idea
- Maze element to navigating from room to room — Idea
- Counteracting impossible situations — Balance Issue
- When skeleton gets a melee hit, crumbles — Idea
- Should Phase Walk be an innate ability, card or specific to s...
- + New

Board view

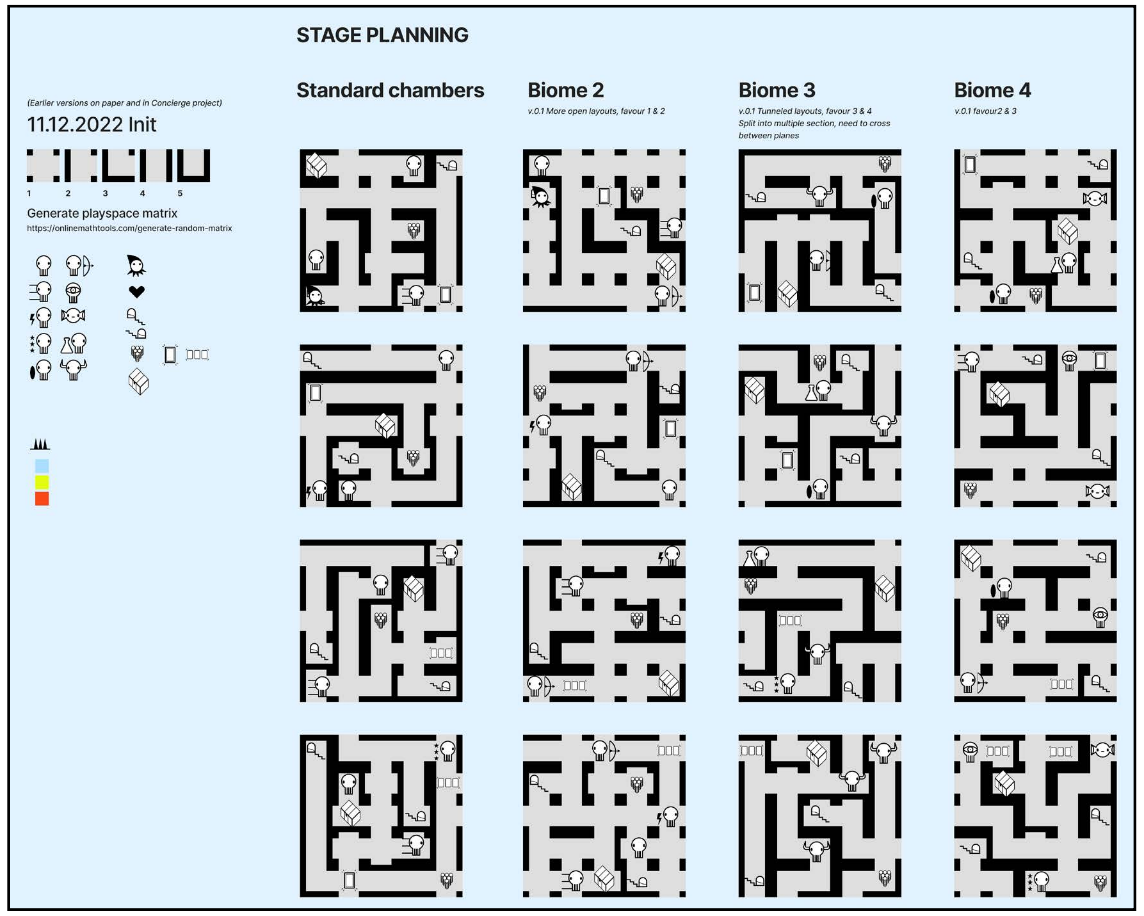
Manor tasks

To-do 7
- Numerize all card attributes for balancing — Not started
- Rules documentation — Not started
- Fill monster data — Not started
- Fill biomes data — Not started
- Fill collectibles data — Not started
- Add colab links and code to Prototyping Setup — Not started
- PLAYTEST: More HP — Not started
- + New

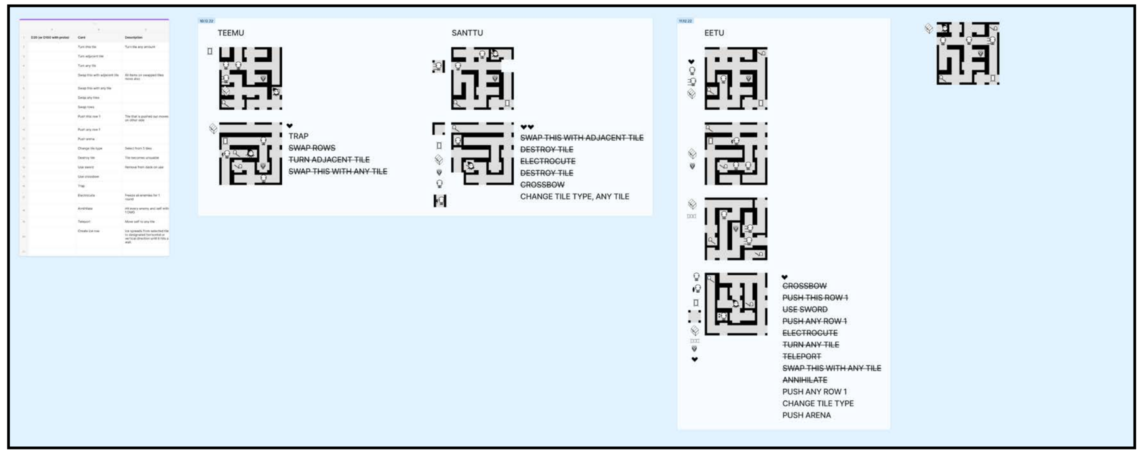
In Progress 3
- Colab vs Spreadsheet balancing — In progress
- PLAYTEST: Phase Walk off — In progress
- PLAYTEST: Phase Walk as resource — In progress
- + New

Complete 20
- PLAYTEST: More AP & HP, cards refresh from deck, monsters are more powerful — Done
- PLAYTEST: Monsters drop money, treasure is cards or money — Done
- PLAYTEST: With only a few transform cards, more like trad powers per hero — Done
- PLAYTEST: 1 / ind. prices from 3 price structure — Done
- PLAYTEST: Pickable AP — Done
- PLAYTEST: Smaller deck w/ weaker cards, upgrade system — Done
- Fill new card cost data — Done
- PLAYTEST: 6x6 & 7x7 board

After realising I had an actual potential long-term game project in my hands, I quickly built a project management setup into Notion, where I tracked all my experiments, ideas, and other tasks.


STAGE PLANNING
(Earlier versions on paper and in Concierge project)
11.12.2022 Init
1
2
3
4
5
Generate playspace matrix
https://onlinemathtools.com/generate-random-matrix
Standard chambers
Biome 2
v.0.1 More open layouts, favour 1 & 2
Biome 3
v.0.1 Tunneled layouts, favour 3 & 4
Split into multiple section, need to cross between planes
Biome 4
v.0.1 favour2 & 3

The Manor would consist of Biomes players would come across. Each biome would have slightly different types of layouts and monsters waiting for them. As opposed to previous game prototypes, here coming up with new content or rules was effortless. They seemed to always fit the theme.



TEEMU

TRAP
~~SWAP ROWS~~
~~TURN ADJACENT TILE~~
~~SWAP THIS WITH ANY TILE~~

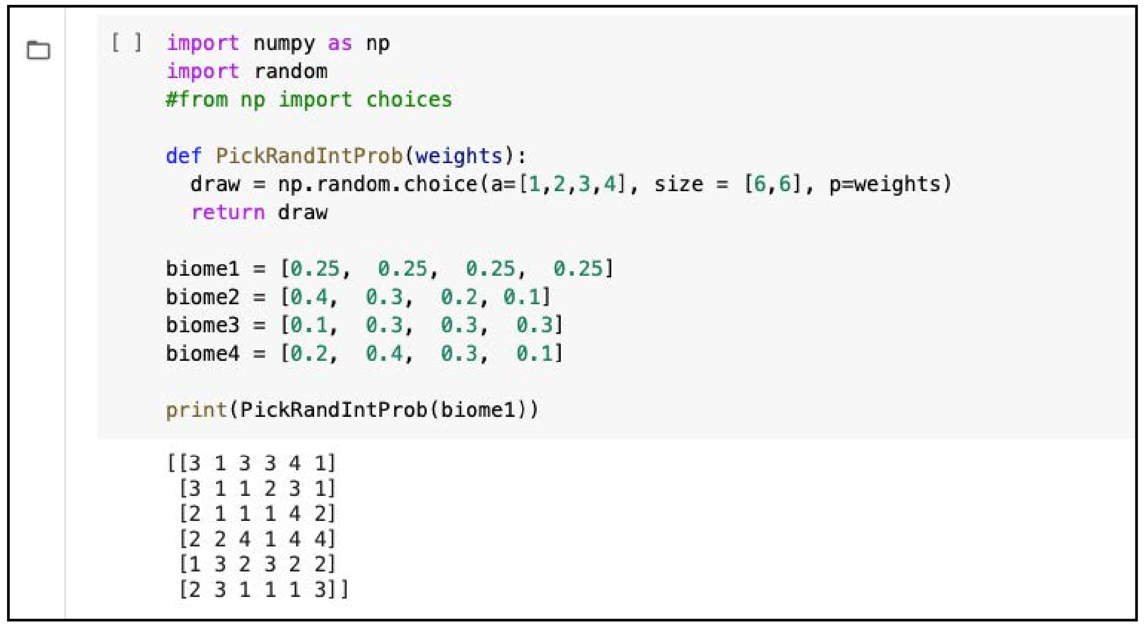
SANTTU

~~SWAP THIS WITH ADJACENT TILE~~
~~DESTROY TILE~~
~~ELECTROCUTE~~
~~DESTROY TILE~~
~~CROSSBOW~~
CHANGE TILE TYPE, ANY TILE

EETU

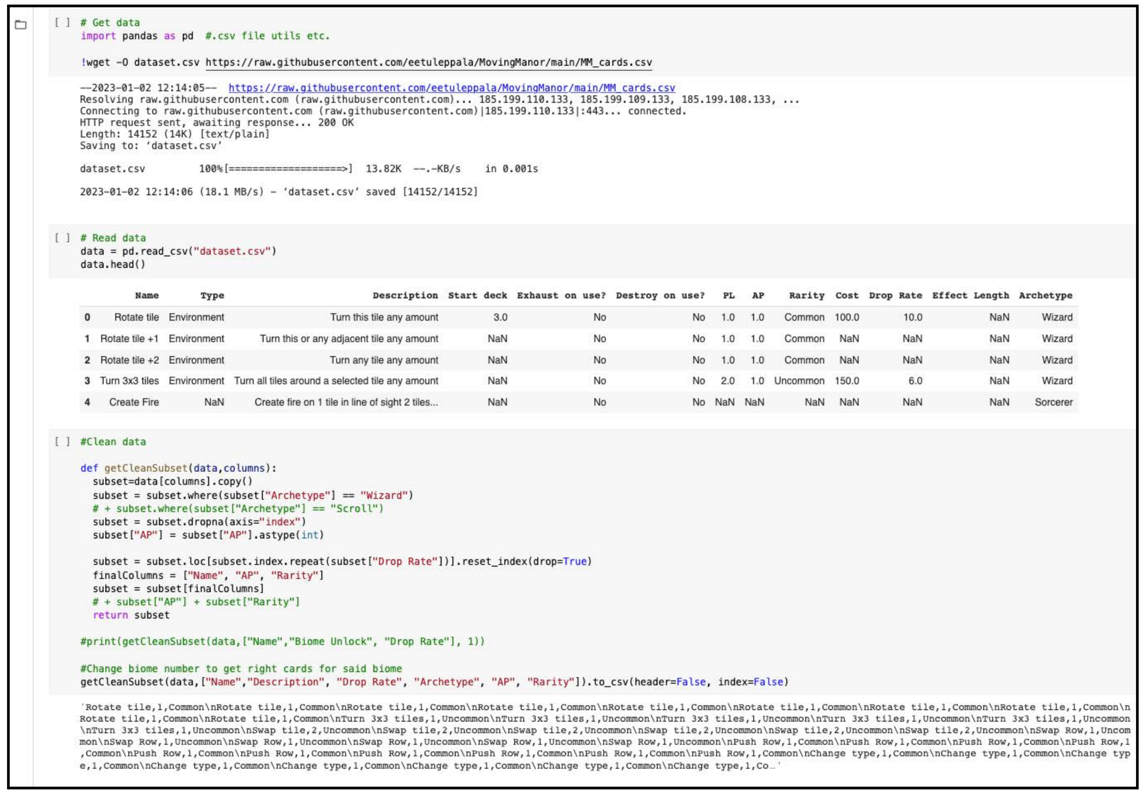
~~CROSSBOW~~
~~PUSH THIS ROW 1~~
~~USE SWORD~~
~~PUSH ANY ROW 1~~
~~ELECTROCUTE~~
~~TURN ANY TILE~~
~~TELEPORT~~
~~SWAP THIS WITH ANY TILE~~
~~ANNIHILATE~~
PUSH ANY ROW 1
CHANGE TILE TYPE
PUSH ARENA

I would host playtesting sessions straight in Figma, because it fitted perfectly the requirements I had for the gameplay, grabbing a certain portion of the map and rotating or moving it. It was actually easier in Figma than in paper prototype form, because physical things tended to fly around once the map was touched. In Figma, I could also track the stats of each player and monitor their card decks.



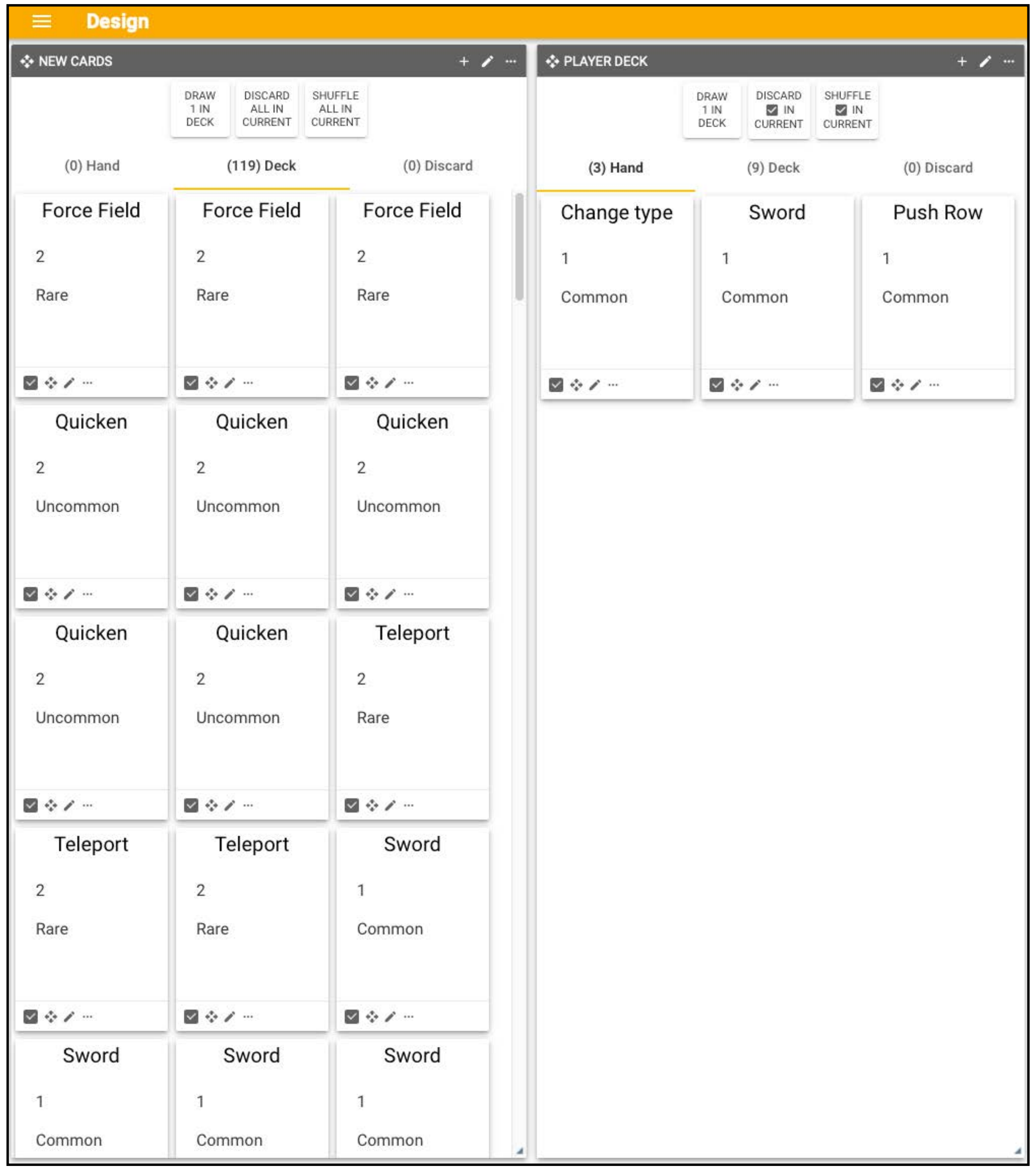
```
[ ] import numpy as np
    import random
    #from np import choices

    def PickRandIntProb(weights):
      draw = np.random.choice(a=[1,2,3,4], size = [6,6], p=weights)
      return draw

    biome1 = [0.25,  0.25,  0.25,  0.25]
    biome2 = [0.4,  0.3,  0.2, 0.1]
    biome3 = [0.1,  0.3,  0.3,  0.3]
    biome4 = [0.2,  0.4,  0.3,  0.1]

    print(PickRandIntProb(biome1))

    [[3 1 3 3 4 1]
     [3 1 1 2 3 1]
     [2 1 1 1 4 2]
     [2 2 4 1 4 4]
     [1 3 2 3 2 2]
     [2 3 1 1 1 3]]
```

I built a system of setting up random floors by scripting a small Python script that created matrices in wanted dimensions and with wanted distributions of numbers. These numbers would correspond to particular types of tiles. I would build the floor by hand based on these matrices. This enabled me to test varying levels of probabilities for each tile type, and how that would affect the gameplay. For example, more open floors enabled ranged enemies to be more dominant.



```
[ ] # Get data
    import pandas as pd  #.csv file utils etc.

    !wget -O dataset.csv https://raw.githubusercontent.com/eetuleppala/MovingManor/main/MM_cards.csv

    --2023-01-02 12:14:05--  https://raw.githubusercontent.com/eetuleppala/MovingManor/main/MM_cards.csv
    Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.109.133, 185.199.108.133, ...
    Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 14152 (14K) [text/plain]
    Saving to: ‘dataset.csv’

    dataset.csv         100%[===================>]  13.82K  --.-KB/s    in 0.001s

    2023-01-02 12:14:06 (18.1 MB/s) - ‘dataset.csv’ saved [14152/14152]


[ ] # Read data
    data = pd.read_csv("dataset.csv")
    data.head()

              Name         Type                                   Description  Start deck  Exhaust on use?  Destroy on use?   PL   AP    Rarity   Cost  Drop Rate  Effect Length  Archetype
    0      Rotate tile  Environment                        Turn this tile any amount       3.0               No               No  1.0  1.0    Common  100.0       10.0            NaN     Wizard
    1   Rotate tile +1  Environment         Turn this or any adjacent tile any amount       NaN               No               No  1.0  1.0    Common    NaN        NaN            NaN     Wizard
    2   Rotate tile +2  Environment                         Turn any tile any amount       NaN               No               No  1.0  1.0    Common    NaN        NaN            NaN     Wizard
    3   Turn 3x3 tiles  Environment  Turn all tiles around a selected tile any amount       NaN               No               No  2.0  1.0  Uncommon  150.0        6.0            NaN     Wizard
    4      Create Fire          NaN        Create fire on 1 tile in line of sight 2 tiles...       NaN               No               No  NaN  NaN       NaN    NaN        NaN            NaN   Sorcerer

[ ] #Clean data

    def getCleanSubset(data,columns):
      subset=data[columns].copy()
      subset = subset.where(subset["Archetype"] == "Wizard")
      # + subset.where(subset["Archetype"] == "Scroll")
      subset = subset.dropna(axis="index")
      subset["AP"] = subset["AP"].astype(int)

      subset = subset.loc[subset.index.repeat(subset["Drop Rate"])].reset_index(drop=True)
      finalColumns = ["Name", "AP", "Rarity"]
      subset = subset[finalColumns]
      # + subset["AP"] + subset["Rarity"]
      return subset

    #print(getCleanSubset(data,["Name","Biome Unlock", "Drop Rate"], 1))

    #Change biome number to get right cards for said biome
    getCleanSubset(data,["Name","Description", "Drop Rate", "Archetype", "AP", "Rarity"]).to_csv(header=False, index=False)

    'Rotate tile,1,Common\nRotate tile,1,Common\nRotate tile,1,Common\nRotate tile,1,Common\nRotate tile,1,Common\nRotate tile,1,Common\nRotate tile,1,Common\nRotate tile,1,Common\n
    Rotate tile,1,Common\nRotate tile,1,Common\nTurn 3x3 tiles,1,Uncommon\nTurn 3x3 tiles,1,Uncommon\nTurn 3x3 tiles,1,Uncommon\nTurn 3x3 tiles,1,Uncommon\nTurn 3x3 tiles,1,Uncommon
    \nTurn 3x3 tiles,1,Uncommon\nSwap tile,2,Uncommon\nSwap tile,2,Uncommon\nSwap tile,2,Uncommon\nSwap tile,2,Uncommon\nSwap tile,2,Uncommon\nSwap tile,2,Uncommon\nSwap Row,1,Uncom
    mon\nSwap Row,1,Uncommon\nSwap Row,1,Uncommon\nSwap Row,1,Uncommon\nSwap Row,1,Uncommon\nSwap Row,1,Uncommon\nPush Row,1,Common\nPush Row,1,Common\nPush Row,1,Common\nPush Row,1
    ,Common\nPush Row,1,Common\nPush Row,1,Common\nPush Row,1,Common\nPush Row,1,Common\nPush Row,1,Common\nPush Row,1,Common\nChange type,1,Common\nChange type,1,Common\nChange typ
    e,1,Common\nChange type,1,Common\nChange type,1,Common\nChange type,1,Common\nChange type,1,Co…'
```

For this game, I also scripted this CSV processor that enabled me to get subsets of my data for an external tool called Iterary I used for managing card decks easily.



Design

NEW CARDS

DRAW 1 IN DECK | DISCARD ALL IN CURRENT | SHUFFLE ALL IN CURRENT

(0) Hand | (119) Deck | (0) Discard

Force Field 2 Rare	Force Field 2 Rare	Force Field 2 Rare
Quicken 2 Uncommon	Quicken 2 Uncommon	Quicken 2 Uncommon
Quicken 2 Uncommon	Quicken 2 Uncommon	Teleport 2 Rare
Teleport 2 Rare	Teleport 2 Rare	Sword 1 Common
Sword 1 Common	Sword 1 Common	Sword 1 Common

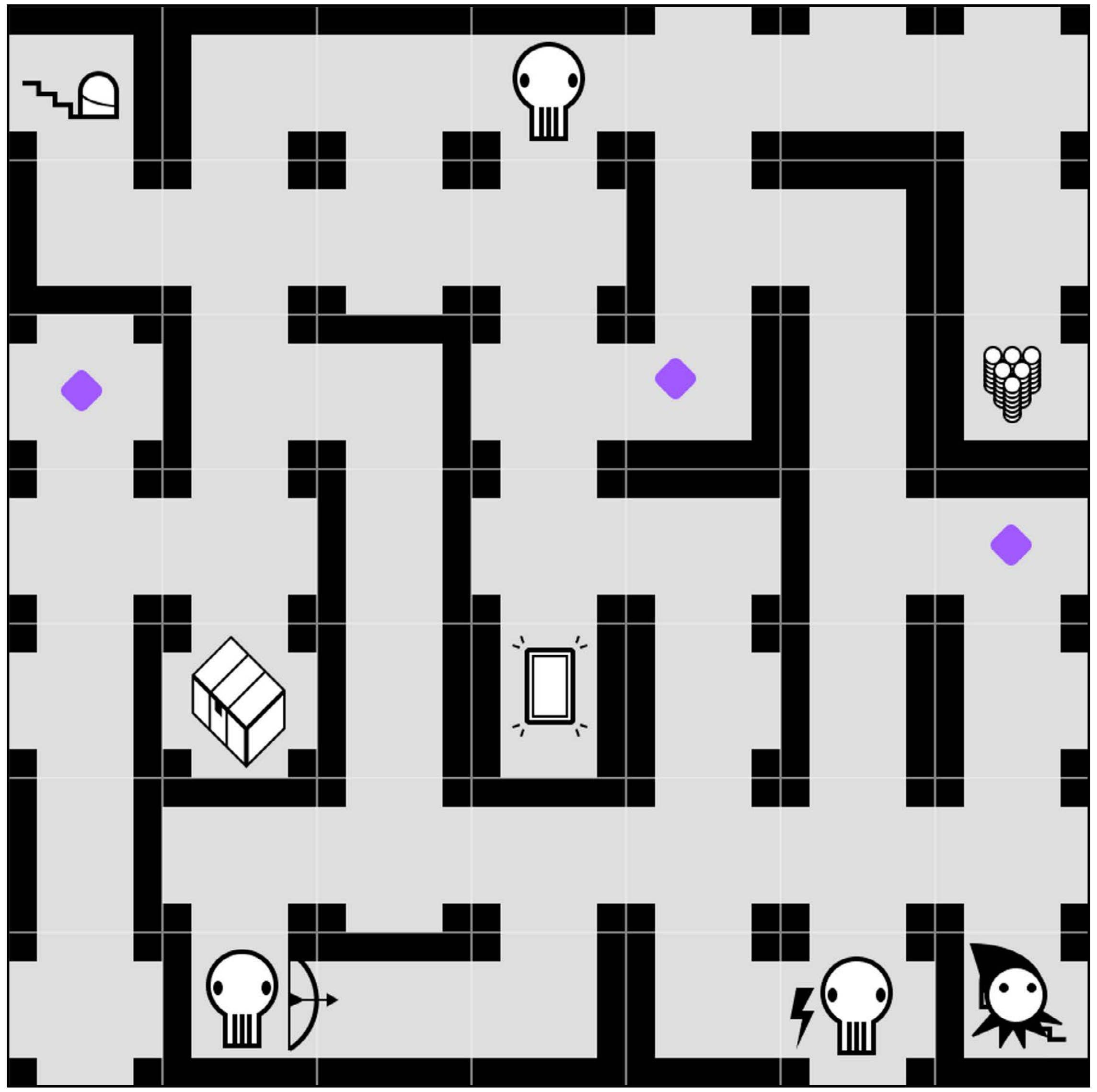
PLAYER DECK

DRAW 1 IN DECK | DISCARD ✓ IN CURRENT | SHUFFLE ✓ IN CURRENT

(3) Hand | (9) Deck | (0) Discard

Change type 1 Common	Sword 1 Common	Push Row 1 Common

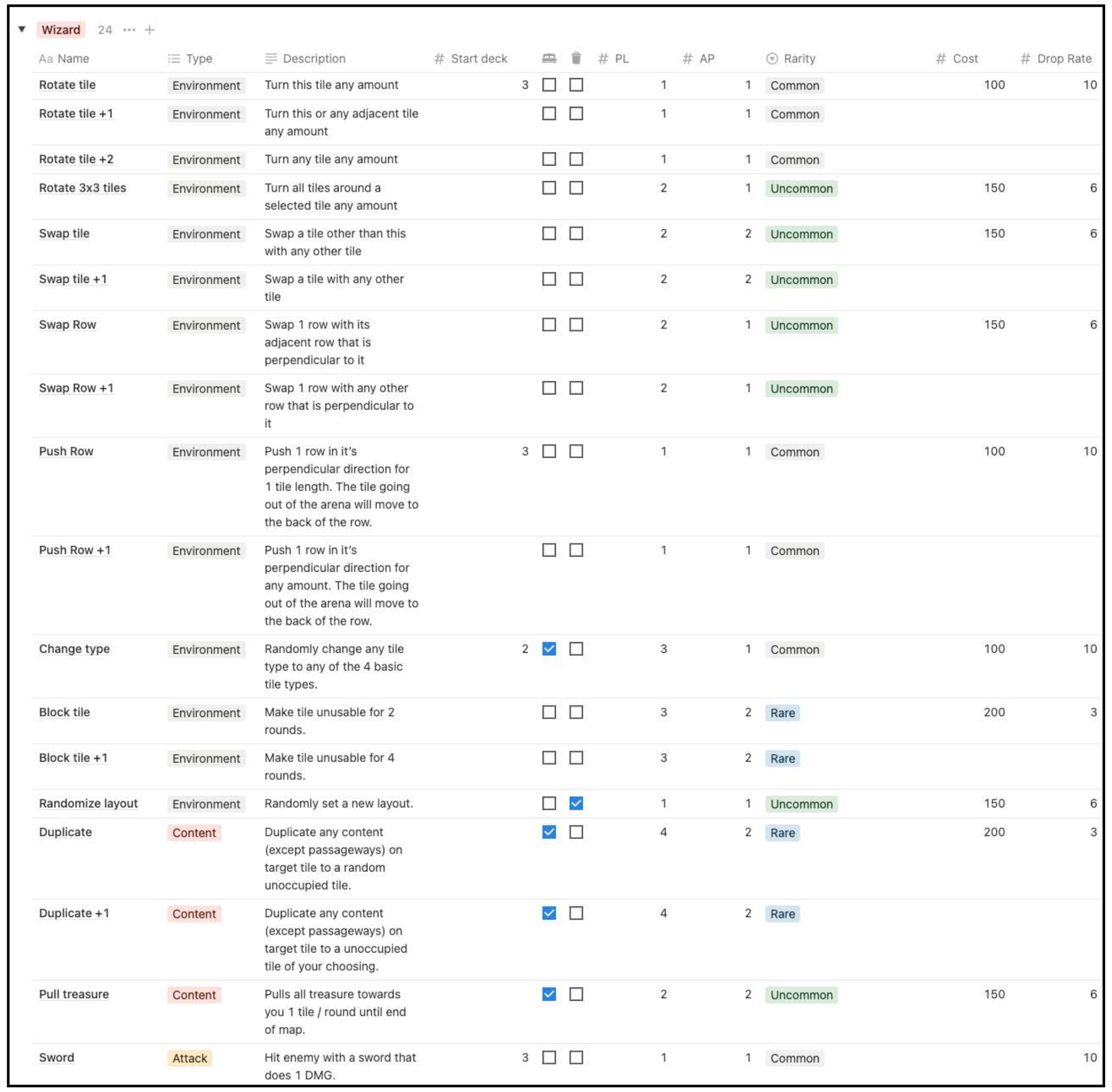
I used Iterary to manage player decks. It allowed me to create decks of cards from a CSV file and allows basic functionality such as drawing and shuffling cards. This was a perfect compromise for my use case, as both making the cards by hand and programming a full-fledged system to handle it seemed cumbersome. While using it is quite heavy, it allowed me to validate a few key questions about the game, namely if the cards scaled correctly in power and if the starting decks were balanced.



Currently, the floors are a bit bigger than in the beginning. I will probably also test 8x8 just because it's the chess standard size and every time I've increased the size of the map, there have been positive developments in terms of gameplay depth. But there's probably a limit.

6.3 REFLECTIONS

I feel like by consciously thinking about the constraints in the beginning stages of this iteration I managed to avoid the common pitfalls I had fallen into earlier. I managed to stay away from Unity long enough to have a valid game concept that had been proven fun on paper. I managed to create a concept that was symbolic and simple enough that I could code it. I also managed to create it based on the earlier coded demo, which I hopefully get to use in some capacity once I get into coding this iteration. I managed to create a system that evokes complexity from the interactions of the components. And most of all I really have a feeling there is a potentially successful game in there.



Wizard 24

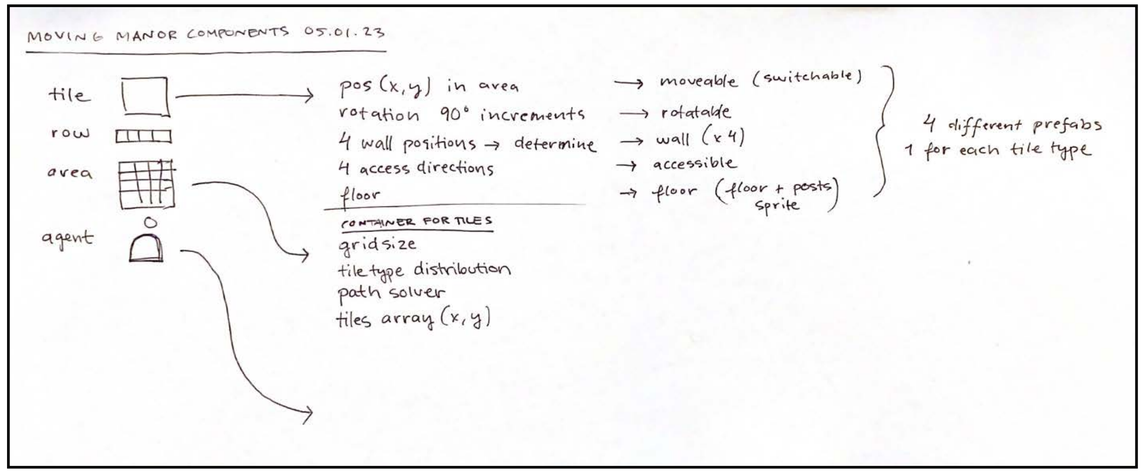
Name	Type	Description	Start deck			PL	AP	Rarity	Cost	Drop Rate
Rotate tile	Environment	Turn this tile any amount	3	☐	☐	1	1	Common	100	10
Rotate tile +1	Environment	Turn this or any adjacent tile any amount		☐	☐	1	1	Common		
Rotate tile +2	Environment	Turn any tile any amount		☐	☐	1	1	Common		
Rotate 3x3 tiles	Environment	Turn all tiles around a selected tile any amount		☐	☐	2	1	Uncommon	150	6
Swap tile	Environment	Swap a tile other than this with any other tile		☐	☐	2	2	Uncommon	150	6
Swap tile +1	Environment	Swap a tile with any other tile		☐	☐	2	2	Uncommon		
Swap Row	Environment	Swap 1 row with its adjacent row that is perpendicular to it		☐	☐	2	1	Uncommon	150	6
Swap Row +1	Environment	Swap 1 row with any other row that is perpendicular to it		☐	☐	2	1	Uncommon		
Push Row	Environment	Push 1 row in it's perpendicular direction for 1 tile length. The tile going out of the arena will move to the back of the row.	3	☐	☐	1	1	Common	100	10
Push Row +1	Environment	Push 1 row in it's perpendicular direction for any amount. The tile going out of the arena will move to the back of the row.		☐	☐	1	1	Common		
Change type	Environment	Randomly change any tile type to any of the 4 basic tile types.	2	☑	☐	3	1	Common	100	10
Block tile	Environment	Make tile unusable for 2 rounds.		☐	☐	3	2	Rare	200	3
Block tile +1	Environment	Make tile unusable for 4 rounds.		☐	☐	3	2	Rare		
Randomize layout	Environment	Randomly set a new layout.		☐	☑	1	1	Uncommon	150	6
Duplicate	Content	Duplicate any content (except passageways) on target tile to a random unoccupied tile.		☑	☐	4	2	Rare	200	3
Duplicate +1	Content	Duplicate any content (except passageways) on target tile to a unoccupied tile of your choosing.		☑	☐	4	2	Rare		
Pull treasure	Content	Pulls all treasure towards you 1 tile / round until end of map.		☑	☐	2	2	Uncommon	150	6
Sword	Attack	Hit enemy with a sword that does 1 DMG.	3	☐	☐	1	1	Common		10

Here is an example of some of the cards in the Wizard's deck. The game will have multiple classes, each with its own playstyle. With the combination of tiles and floors, it's easy to come up with different ways of traversing the floors without them being too foreign in terms of expected code. Things are symbolic enough that each card will only turn on or off some particular script on game objects, but simultaneously I think I've found a sweet spot where the symbolic nature of the game still allows for the player to experience a genuine thrill of running around in a mad mages manor, barely surviving.

7 MOVING MANOR – NEXT STEPS

As of writing this report, I have not begun programming the game. However, I still try to avoid fully committing to writing any production code and instead try to validate aspects of the game by using the same hybrid approach I demonstrated earlier. I will write modules that will allow me to prove that a design decision is valid, without putting in too much effort.

I will also need to create a sort of decision-making procedure for calling when will I fully commit to making the game a reality.



Ideating some of the aspects needed in code for the game to exist.

8 LEARNINGS

Early validation is needed in order to use the limited time one has most effectively. Seeing as game development is very time-consuming, the further in the design process of any one particular prototype you go before making the decision of whether or not to continue, the more you have wasted your time. And also it is important to mention that the further you go in a game development project, the more time it takes to make any one iteration that is significantly different from the previous one in order to test. So early validation rigorously is key.

For me, this usually means validating game concepts before any major production code has been written. As a solo dev with only beginner / intermediate knowledge and experience in software development, it’s easy to find oneself working on tangents in code that may or may not have anything to do with the actual issue or the final game, just because it’s a new and interesting way to solve a problem. Optimising one’s time so that good quality production code is written simultaneously as one is learning is a good way to prune the long time periods it takes to learn to code and to develop a game.

Here I outline some of the tools that helped me with early validation. The first is to take the game concept or idea and strip it down to its most basic and simple form. If the game concept still holds and is fun to play with, it may be then easier to get into developing rules and game logic that are based on simple principles, helping the designer to understand why certain things in the game exist.

Another good way of doing this is to set an artificial constraint of imagining making a board game instead of a video game. This way there is no chance but to ideate ways to prototype on paper. Later procedurality and computation can be added in to enhance the experience. But I still think this is the best way to ensure that as a solo dev the size and complexity of the game project stay reasonable.

Speaking of constraints, having them as opposed to not is better. Spelling out what can and cannot be accomplished gets rid of the mindsets that may lead to aiming for something outside the realm of possibility. Also, setting constraints for design makes it easier to evaluate whether to continue on something, rather than finding out later that something cannot be completed due to inexperience or lack of manpower.

Spelling out thoughts is a powerful way of working through an idea. Logical errors that couldn’t be seen earlier reveal themselves.

Writing a stream-of-consciousness problem-solving document becomes almost like a dialogue between multiple minds. After this exercise, interesting bits can be highlighted and carried over to a separate similar session or on post-its. Otherwise, there is a risk of just creating concepts where nothing gets validated.

For me at least, it’s also important that making decisions feels effortless. Or at least once a decision has been made, it can be tested and its validity confirmed. This gives a sign that there is progress happening with the game, instead of gears grinding to a halt. If the decision-making is thorny and always painful, whether it relates to matching the theme to the mechanics or just coming up with relevant game mechanics that would fit the bill, that can be a sign that the game idea needs a fundamental shift in its core logic. Something isn’t aligned at the most basic level, so it affects everything else. A separate issue relating to decision-making is that sometimes the decisions themselves are not even relevant for validating the game further. Brain gets stuck in a concept design loop, instead of trying to analyse what can make the game better now. Here spelling out things also works. What am I deciding on and why is it important? How will making the decision help me finish this game?

Another point about the feeling of game design is to listen to one’s own ambitions as well instead of only prescribing some set idea of what the game should be about. Making games is a long endeavour, and the process should ideally be enjoyable. If the game concept manages to encapsulate perfectly some ideal game experience, but there is no excitement, should that game exist? In the end, game development requires constantly also playing that game daily for possibly years. On the other hand, if there are viscerally positive reactions to game ideas or design choices, in my opinion, they should be followed up on.

If there is a feeling of being tied to a knot, it’s good to look for a fresh perspective by just taking a break and regaining self-confidence. While taking breaks from projects is not something I’ve seen advertised a lot anywhere, I think it gives a fresh set of eyes and a more objective lens to look through.

Finally a bit about the coding trap, that is an ever-present threat for me at least. If I feel like there is no other way than to jump into code, I need to evaluate my own wants and needs at this point in time. Do I just want to develop code? Or is the task actually such, that code needs to be developed? And if so, is the mechanic so important it cannot be changed so it can be represented analogically? It could only be represented in this analogue way for only a few iterations, while still keeping in the back of mind that the actual mechanic would be different computationally. And finally, if actual code needs to be developed in order for you to progress, can it be scrappy and fast code? Could only a simple module aid the paper prototyping process further rather than turning the entire project into a digital production?

9 RESOURCES AND LINKS

FIGMA BOARD LINKS

- Hegemony Interactive Prototype
- Moving Manor Figma playtest file

GITHUB REPOS

- Space Saloon repository
- Concierge repository

COLAB

- Moving Manor Cards CSV processor
- Moving Manor Map Layout Matrix Generator

ITERARY LINK

- Moving Manor Player Deck Tool