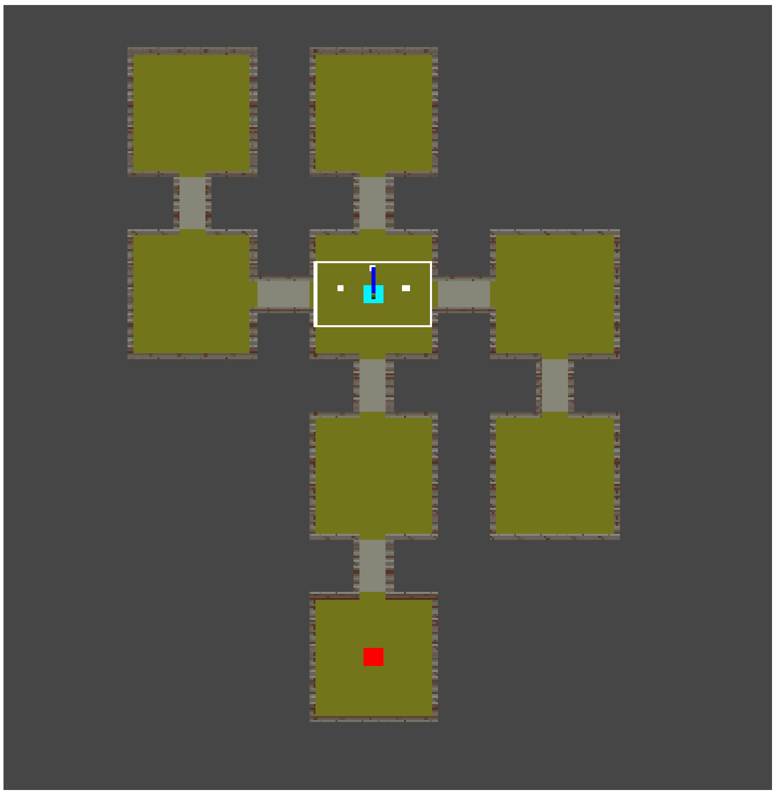
Prototyping

Prototype 1

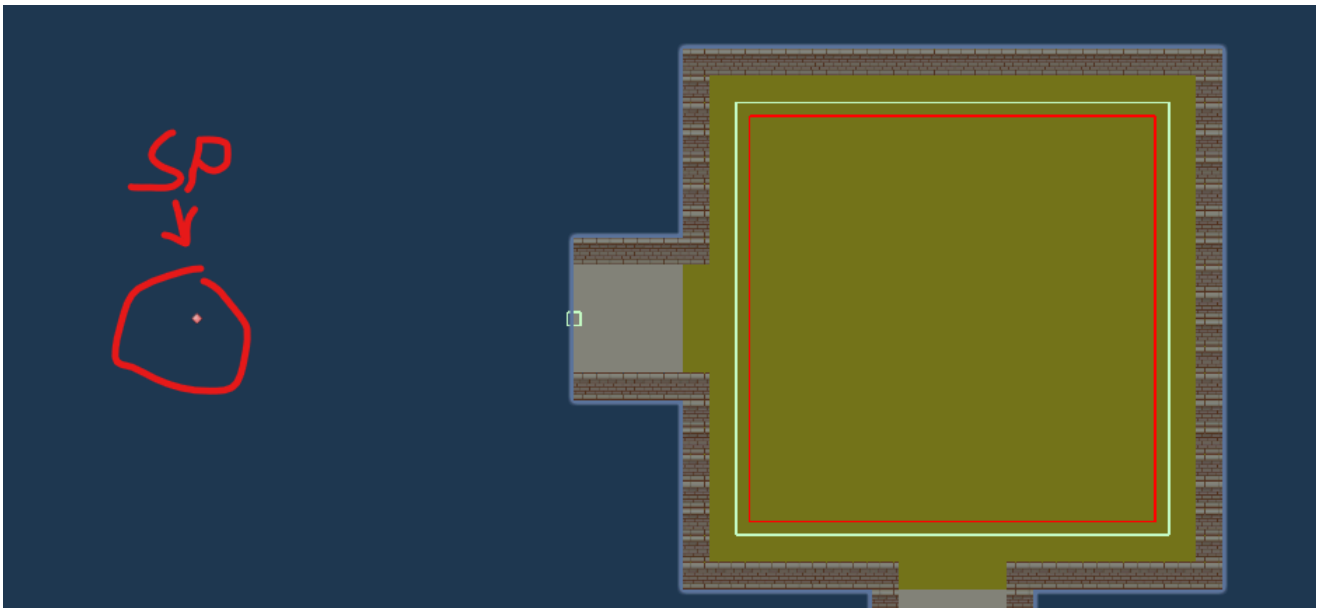
Sort of like an endless runner, but instead it goes down. There is a wall on each side and the player can jump between those walls. The random topic here is the random obstacles and the obstacles will be enemies who spawn on the walls. The player can kill those enemies by jumping to the other wall. There are 3 enemies in this version. One that just hangs on a wall and shoots projectiles at you, one that is running on some surface back and forth between the walls and an enemy that homes in at you and you have to jump to kill it.

Week 2 (8.10. >)

After a week we decided to not do 2 separate prototypes. Although we understand that the separate prototypes serve to try different things, we have 2 reasons. We would really love to do a roguelike and also quite big time constraints and we would be behind if we started the roguelike later. So I got to thinking. The first and biggest thing in the roguelike is the procedural environment. I am doing this for the first time so doing a completely generated dungeon would be hard. We also want some control of the rooms we make. Obviously there is still control in complete generations. But for use it would be easier if we create the rooms and then use them to create a grid like dungeon, where there would be one room in a grid cell with corridors leading to other rooms.
So once again I am pained by the time constraint. We have to do a prototype pretty much every week and that doesn't give the time necessary for me to dive into the algorithms one would use for something like this.



I created a very basic dungeon generation. There are multiple rooms created with corridors already built. So there are rooms with combinations of corridors on the right, left, top and bottom. Each room has a spawn point on each side there is an opening. And there will spawn the next room from an array that has rooms with this kind of opening.



This would spawn a random room with an opening on the right
So while this works and creates a pretty nice dungeon of what I want there is a bug. Sometimes there is a room spawned either going into nowhere or a wall going into a room that doesn't have an opening there. Also it generates rooms until it can't because it is bound to spawn closed rooms. Obviously because the spawner is not checking for this kind of thing. But this is all I could do this week.

Week 3 (15.10. >)

I sort of fixed the bug. I unfortunately had to write a new script for checking if there is an opening to the rooms. I did it by placing wall checks on the edge of the colliders and checking if there is a different wallcheck object. If there is it does nothing. If there is not it replaces the current room with a room that is closed and has only one opening. This kind of opening only works because all the rooms have maximum of 2 corridors. This wall check script is bigger than the room spawner.
I had a revelation while working on this bug. The way I have the dungeon spawning set up is very very inefficient. It doesn't allow for any easy expanding, let's say for some special rooms to spawn, or just controlling easily what rooms spawn. I realized I would have to rewrite the dungeon generation after the prototyping phase ends in a better way. Currently I have to keep it like this because I have to work on the other prototype iterations. I have to again emphasize on the lack of time we have for quite big things. I would love to experiment and figure out stuff myself but it's impossible.
I also used this bug fixing to make a limit to how many rooms spawn instead of until it can't spawn anymore. So the arrays containing the rooms don't include the closed rooms anymore and instead it checks how many rooms are spawned and if there is more than the limit it will not spawn. And since the wall check is there it will close up the the open rooms, so it's nice that it combined like this into a different problem.

Also this week is for the topic Random Obstacles not just bug fixing. So basically the way we want it is that when the player enters a room that is a combat encounter random enemies spawn in random positions.
And here comes the combat loop we thought about. It will essentially be a bullet hell where the player mainly focuses on dodging all the projectiles that there are flying against him. The attacking will be that it automatically shoots at the closest enemy. Why? The game will already quite hard by itself and if we add manual shooting on the top of dodging a lot of bullets it might be a lot. However for the attacking there is fortunately more time to experiment in the future.
Also we thought about the health system. At least to me it is quite important because it is a bullet hell with focus on dodging, therefore focus on not being hit. So having a basic health bar I feel like is just disappointing. I thought about having just health points, basically hearts, and when the player gets hit it will decrease by one point (heart) and put emphasis on the hit so some kind of flashy animation.
I was thinking of Hollow Knight while thinking of the health system and because we want the player to heal somehow we can add also the charge healing. Where there is a container and it fills up whenever the player hits something and when the player want's to heal, he will hold a button it will stop the player in place while it's charging and after a while it will heal the player by one point.
This is something that will definitely make the game pretty hard but to me it seems like something that benefits game where there is a focus on not being hit and I presented it so to my team.
Mike who unofficially joined our team, created a simple enemy on my request. It is a very simple stationary enemy who constantly shoots at the player in range. It serves for the current prototype and time constraints.

Week 4 (22.10. >)

This week we got Sophie and Mike (Antonio Lazarov, officially now) on our team. Both great additions, Sophie who wants to be more on the design side of the game and Mike on programming who will lessen my load a lot.
So while did make a functioning enemy script it was made for 3D, so it didn't shoot at the player who is 2D. He did it on purpose, apparently because he was once told that making a 2D game but still with 3D objects eliminates a lot of problems mainly with collisions but no concrete examples were given. I asked Hagen who substituted this week and he said he doesn't see a reason to do so and just use 2D for 2D games. Mike then agreed to change the script for 2D.
I talked with Hagen, how I could go about doing the upgrades for the random abilities. We talked for a while and basically pointed me in the direction of Scriptable Objects so I have to look at those and see if I can do them this way.
Also I created the basic combat loop of shooting at the closest enemy.
Now that we have a design person on our team (Sophie) we can have some good input. She said that the health system is depending on how hard we want to have our game. I also asked her to create some ideas for upgrades for this week. I also talked with her a little when

asking about what she has come up with. And it does seem like she might make us more grounded because she did push back on asking us on more concrete stuff we're able to do before she can come up with the upgrades. So it's great to have someone like that. And to think I doubted her skills before, I was proven wrong.

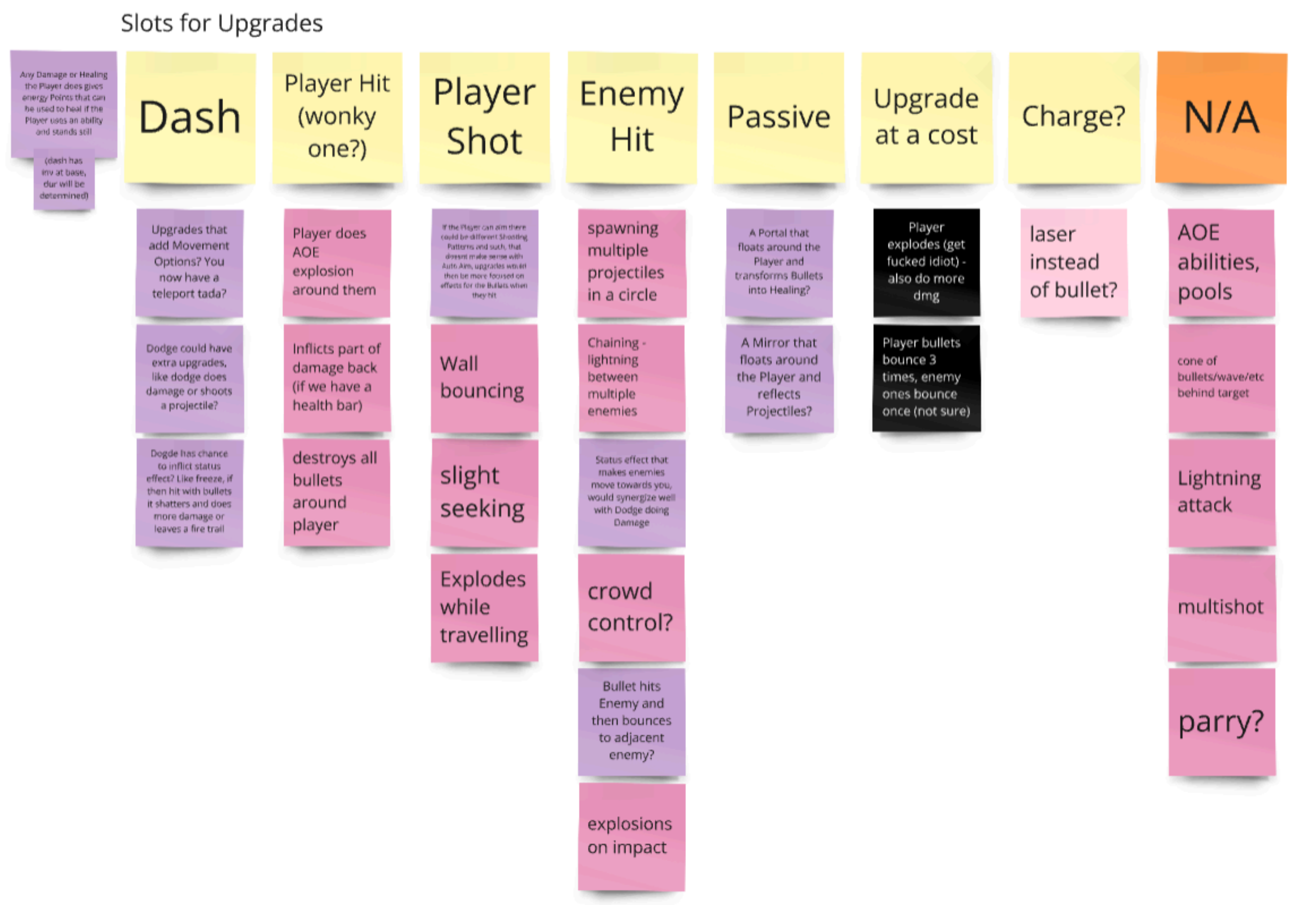
I created a simple Upgrade system. I went with Scriptable Objects, something Hagen told me about. Unfortunately since I didn't have that much time, I couldn't implement the choice to pick which upgrade to get from as a reward. So instead I made something I already know and it drops the upgrade on the death of an enemy.
I quickly found out how beneficial Scriptable Objects are, to create a lot of similar but different objects and store data. So I created a Scriptable Object template (abstract) and then made specific upgrades that inherits from the main template. Then I made an object that drops on the death of an enemy and is like an empty container. On death it just sets a random upgrade from an array of the upgrades and sets some stuff like the name.
Although while this works I have no idea if the way I set it up is a great way and I have to ask Jorg about it.

Week 5 (29.10. >)

In this week I did 2 main things. The health system and the shooting modes. Because Mike insisted that he would like to have manual shooting instead of auto shooting, instead of deciding on one mode I implemented so that the player can toggle the between the auto and manual shooting modes and we decided that we will let people try this out on the playtesting evening and see what they like.
I also created the health system and luckily I was successful. It is how I described, player has health points, when the player gets hit with a bullet the time stops to simulate a hit stop. Because there is no animation right now the game just stops for a second. I also am curious how people like it.

One big thing that concerned me a little also was the scale of the sprites. Because Unity Tilemaps scales down the sprites to 100x100 it made the tiles created by Clara so much smaller. We were struggling to figure out how to go about it also with the idea of how to make it as modular with the procedural generation. As of right now the procgen is not that modular and I am planning it on rewriting it. But we are struggling to figure how much the camera sees, how big dante is and how big the rooms/environment tiles have to be.

On Monday, the day before playtesting evening, we had a meeting. We mainly discussed the gameplay and the upgrades.



Slots for Upgrades

Any Damage or Healing the Player does gives energy Points that can be used to heal if the Player uses an ability and stands still

(dash has inv at base, dur will be determined)

Dash	Player Hit (wonky one?)	Player Shot	Enemy Hit	Passive	Upgrade at a cost	Charge?	N/A
Upgrades that add Movement Options? You now have a teleport tada?	Player does AOE explosion around them	If the Player can aim there could be different Shooting Patterns and such, that doesnt make sense with Auto Aim, upgrades would then be more focused on effects for the Bullets when they hit	spawning multiple projectiles in a circle	A Portal that floats around the Player and transforms Bullets into Healing?	Player explodes (get fucked idiot) - also do more dmg	laser instead of bullet?	AOE abilities, pools
Dodge could have extra upgrades, like dodge does damage or shoots a projectile?	Inflicts part of damage back (if we have a health bar)	Wall bouncing	Chaining - lightning between multiple enemies	A Mirror that floats around the Player and reflects Projectiles?	Player bullets bounce 3 times, enemy ones bounce once (not sure)		cone of bullets/wave/etc behind target
Dogde has chance to inflict status effect? Like freeze, if then hit with bullets it shatters and does more damage or leaves a fire trail	destroys all bullets around player	slight seeking	Status effect that makes enemies move towards you, would synergize well with Dodge doing Damage				Lightning attack
		Explodes while travelling	crowd control?				multishot
			Bullet hits Enemy and then bounces to adjacent enemy?				parry?
			explosions on impact				

(sorted by priority)

We thought of an upgrade system similar to the one of Hades. Where on the action of a player there would something happen. The player can choose what upgrade he has in each slot and replace them. Basically the whole meeting we brainstormed what upgrades the player could have and what sort of actions there could be performed by the player. Basically at the end we came to the conclusion of having these actions (viz. image above). When player dashes, when he gets hit, when he shoots, when an enemy is hit, a passive, risky upgrades (high risk, high reward), maybe a charge ability (low priority), and lastly upgrades that we might implement if there is time. the charge is also undecided because there is already a lot of actions the player can do and we don't want it to be confusing.