COMPUTER GAMES LABORATORY
WINTER TERM 2017/2018

Hikari no tō

tower of light

Artem Bishev - Jonas Mayer - Muhammad Inshal Uddin - Paul Preißner
Team **Pick One**

CONTENTS

1. GAME PROPOSAL

1.1 Game Description

Overview

Summarized in one sentence, the game is a cooperative networked dungeon crawler enriched with native virtual reality support.

The main objective is to create a cooperative dungeon crawler with VR and rogue-like elements. During a game, up to four *Diablo*-like crawlers must work together to survive and work their way through a procedurally generated dungeon, supported by a dungeon master. The dungeon master is a god-like entity that can observe and influence the dungeon from a giant's VR perspective as well guide and support the other players. The crawlers and the master working alongside each other to achieve a common goal implements the **"Together"** theme for this project.

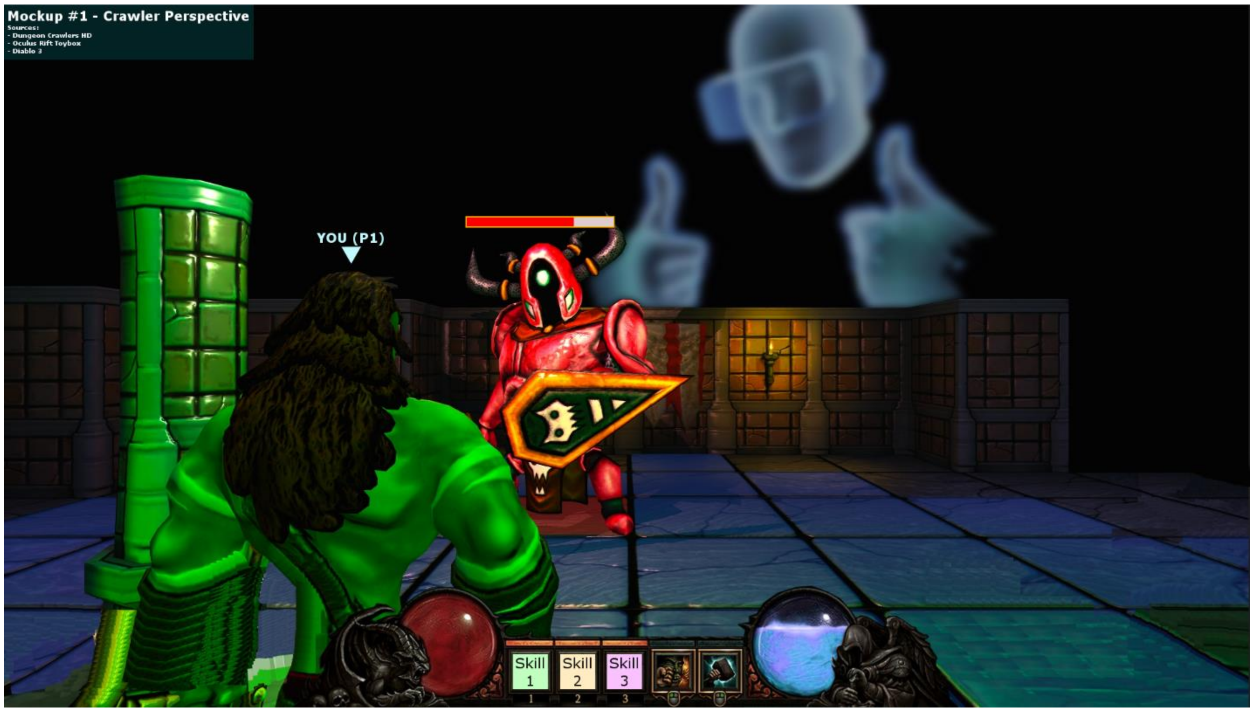
This document will discuss the basic structure and mechanics of the game in question. As such, it is divided into the following sections: *Crawler and Master Gameplay*, *Overall Mechanics*, *Style and Setting*.

Crawler Gameplay

The crawlers play on a standard PC using keyboard and mouse. Each crawler has an individual perspective of the map containing a real-time view of their immediate surroundings and a structural view of the dungeon rooms they have explored thus far (i.e. Fog of War). This necessitates the presence of an intermediary party (the master) to act as a coordinator between the crawlers. Their primary tasks are to defeat foes, interact with objects inside the dungeon, reach certain locations, collect loot and the like.

Each crawler has special abilities depending on their class like melee, ranged, support, etc. with its own set of strengths and weaknesses. Thus, a single crawler will not be able to master the dungeon alone. These classes not only provide a different experience for each player but also give them a specific role in the team. A squad of multiple players will ideally pick different classes to complement each other's strengths and weaknesses.

No levelling system, or a very rudimentary one, will be implemented to avoid programming complexity. The objective of classes and abilities is to provide the feel of a role-playing game without creating an entire "*Dungeon and Dragons*"-like stat levelling system.



Master Gameplay

The master plays the game using a VR headset and hand controllers. The dungeon is mapped to the VR space of the system, so the master can overlook the entire dungeon and focus on areas of interest intuitively. Floating above the dungeon, the master cannot take damage. He can see the crawlers running around but only has a limited understanding of precise enemy activity. From the crawlers' perspective, the master will appear as two hands and an abstract face representation floating above the dungeon.

The role of the master player can be summarized in two words: "Guide" and "Support".

As a guide, he leads the crawlers to their destination. This can be in the form of helping isolated crawlers avoid combat, bringing them together to regroup, coordinating their individual movements and so on. To do this, his set of actions will include pointing in certain directions or placing physical markers. As such, the master player functions as the strategist of the group.

In his supporting role, the master has different abilities to benefit the crawlers or harm enemies either directly or indirectly. Indirect measures include altering the dungeon structure to benefit the crawlers, spot enemies, give temporal speed, damage and health boosts to crawlers directly, nerf enemies similarly, set up traps for enemies and supporting structures for crawlers. The master can also directly heal crawlers and damage enemies but is limited in accuracy in the scale differences and the clumsiness of VR direct manipulation. As an example, the master can toss a physics-based fireball that will cause a large area damage that can also damage the crawlers, or a heal ball that will also heal enemies. The challenge for the master is therefore to assess the usefulness of naturally inaccurate but powerful direct interaction.

Despite being god-like, the master is limited in power. Since abilities are limited in their usage by cooldowns, resources or consumables, he will have to manage his abilities carefully to be able to use them when they are needed most. This includes having to collect resources or collectables or have the crawlers supply them by picking them up or finishing side quests.

Overall Mechanics

Initially, all crawler players are dispersed randomly in the dungeon. They are isolated from each other and have no knowledge of the others' location. The dungeon master must micromanage navigation and support in this phase. They explore the dungeon either following the master's directions or of their own will. Although "going rogue" is possible, players are additionally incentivized to form groups and cooperate by scaling enemy difficulty and/or employing debuffs on isolated players.

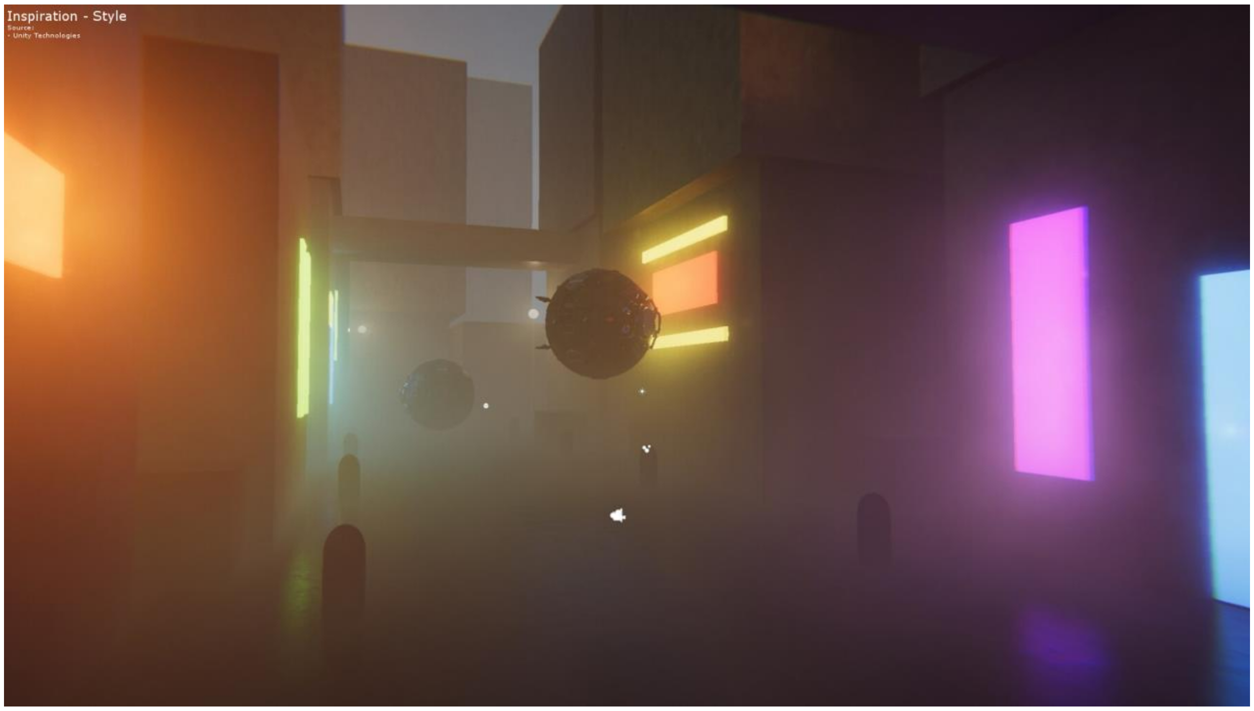
Once the crawlers are together, all players must work **together** to accomplish a given goal such as defeating a boss, solving puzzles, finding treasure, eliminating all enemies in the dungeon, etc. They receive rewards upon completion of the goal after which they may continue to the next level of the dungeon and start again.

Setting and Style

To not end up with yet another dungeon crawler with a medieval/fantasy setting, the decision was made to set the game in a modern Asian city. Instead of a bunch of adventurers having to crawl down through different levels of a dungeon, a squad of diversely skilled street warriors has to fight its way up through the different levels of a fortified high-rise tower in order to rescue their friend from a rivalling Yakuza clan. While he's being kept for his supernatural powers and is unable to escape by himself, he uses his powers to guide and support his friends on their way up.

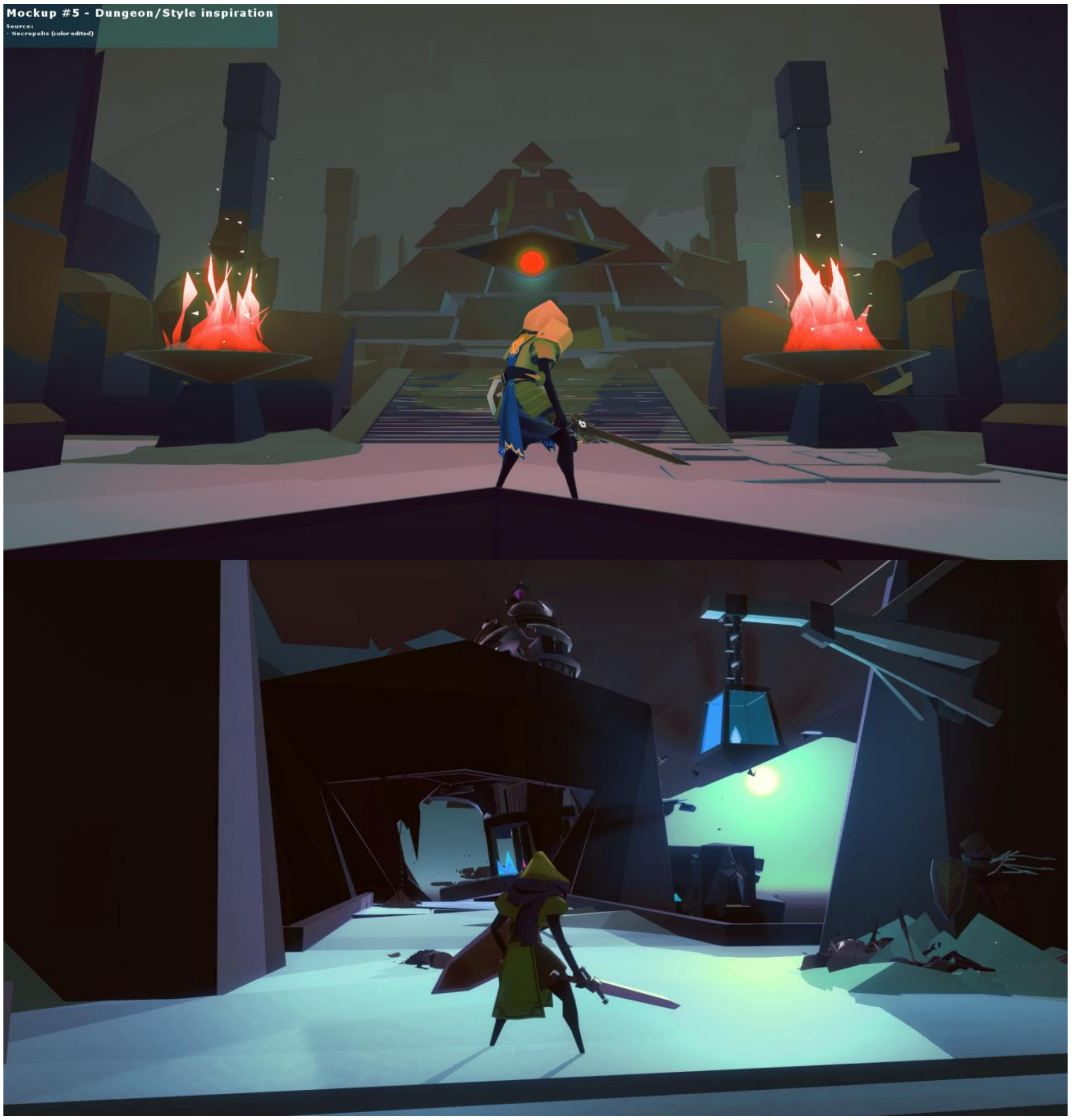
The graphical style will be a minimalistic 3D style (à la Superhot). This decision was made due to limited artistic resources and a focus on technical aspects and gameplay. To keep it visually interesting, the visuals will be dominated by strong contrasts between shady, foggy darkness and colourful, bright "Neon" lighting and particle effects, inspired by nightly streets in Asian cities like Tokyo that are dominated by illuminated street signs.

Mockup #4 - Tower/Style inspiration
Sources:
- Necropolis
- Shanghai Tower
- NEON
- Oculus Rift Toybox
- Shadowrun Hong Kong
NEON
NEON


Inspiration - Style
Source:
- Unity Technologies

Xavier Portela

Inspiration - Style


Mockup #5 - Dungeon/Style inspiration
Source:
- Necropolis (color edited)

1.2 Technical Achievement

The game is to be implemented using Unity3D, using Unity networking for player synchronization and virtual reality controls for the master.

From the gameplay and setup description, the following are the main technical challenges to tackle to implement the game:

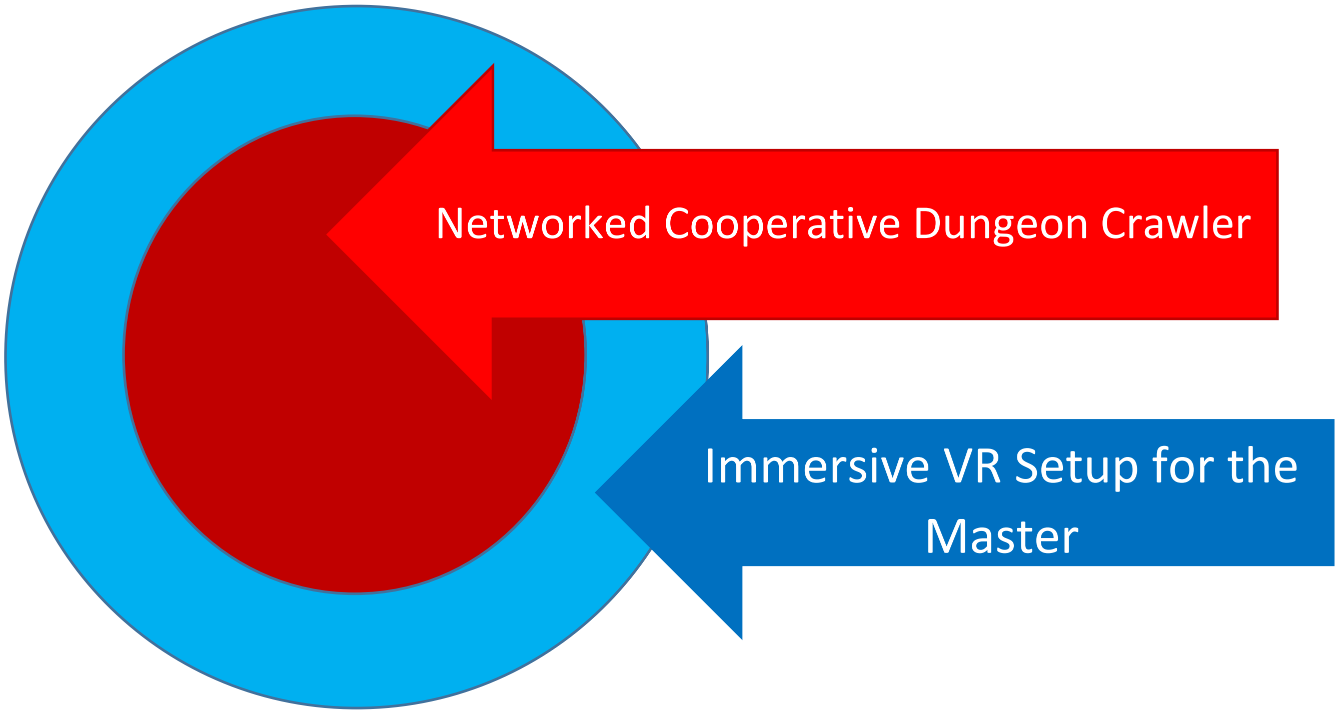
1. Set up stable and efficient networking between all players
2. Set up and tweak virtual reality with hand tracking
3. Procedural dungeon generation

1 and 2 look to be the primary tasks as they are the backbone of the game. Networking is to be constructed off of Unity's built in networking functionality, where the main challenge lies in correctly handling server/client communication, tweaking latency and link load, continuous synchronization and possibly error correction.

VR setup and tracking are planned to rely on the Oculus Rift CV1 headset and included "touch" controllers. While camera setup is simple with recent Unity versions and only requires some tweaking to camera parameters, hand tracking might turn out to be a larger challenge for correct translation, accuracy of control and error (both digital and physical) correction/avoidance.

Procedural dungeon generation is still an unknown. The most likely candidate is to use the well documented Cellular Automata algorithm and premade level modules. Nevertheless, considerable effort is expected.

1.3 "Big Idea" Bullseye


Networked Cooperative Dungeon Crawler
Immersive VR Setup for the Master

1.4 Development Schedule

Functional Minimum

- Basic crawler
  - third person player controller
  - box
  - no classes
  - can deal and take damage
- Basic VR master
  - box for head
  - two sphere hands
  - hand-tracking
  - buff and debuff by button press (no physics and ray casting)
- Basic enemy
  - red boxes
  - can deal and take damage
  - primitive movement (no pathfinding)
- Basic map
  - Hand-made
  - square shaped plane
  - box obstacles (“walls”)
  - spawn points
  - enemies
  - end-goal to restart/win
- Networking
  - world objects synchronisation
  - crawler and master synchronisation

Low Target

- Classes for players
  - Four classes with two abilities each
- Physics-based abilities for the master
  - heal orb
  - fire ball
- Four types of enemies
  - different attacks
- One primitive Boss
- Better models
  - humanoid models for crawlers
  - hand model for master
  - whatever models for enemies
- Communication mechanics between players
  - Creating a mechanism for crawlers to signal master
  - More communication options for master
- Prefabs for 10 rooms and assets for later use in dungeon generation
- 4 handmade examples of levels with enemies and bosses

- Moving to next level after reaching end-goal
- Win screen after completing all example levels
- Primitive UI for crawler and master

Desired Target

- Total of 6 classes for crawlers
    - Main + Side attack
    - 3 special attacks
- 6 different Enemies
    - advanced pathfinding
    - advanced idle behaviour
    - random spawns
- 2 fleshed out Bosses with different behaviours
- Hide enemies from master
- Higher variety of skills for master
    - abilities to alter the dungeon layout (like creating and destroying the walls)
    - make enemies visible for short durations, spot enemies and traps
- Basic eye candy (lights, particles)
- Basic SoundFX
- Main menu
    - Server lobby system
- Mini-map for crawlers with fog of war
- Semi-stat based system for crawlers (strengths and weaknesses)
- Procedural generation of dungeon
    - more props and tiles
- Better models (enemy models, variety)
- Basic story elements
- Add crabs as enemies
- Collectibles (Boxes)
- Win screen

High Target

- Cooler technical stuff, such as volumetric lighting for fog and neon lights
- Loot and artifacts that can make each run distinct and interesting
- Better level generation, puzzles
- Collective AI for enemies, different behaviors
- Music
- Sounds of enemies and atmosphere
- More fleshed out story
- Levelling system for players
- Separate levels by themes (10 levels dark and gloomy, 10 tropical, etc)
    - Create enemies according to the theme of the level
- Final boss fight, end scene
- Completely randomized spawns of enemies and Bosses

Extras

- Deep Story
- Matchmaking
- Side-quests

- Steam Integration
- Steam Greenlight

Detailed schedule here:

https://app.agantty.com/sharing/6350e3c98f45772e54ad99f881c67321

1.5 Assessment

The game's main point is to provide the players with a unique cooperative experience within a familiar concept. They will feel right at home in the setup of a dungeon crawler, but the master player, the vastly different viewpoints and the distinct task distribution give a new spin to the genre. Crawlers and master must play together, build on each other's strengths and come up with a shared strategy in order to win the game. Each player should feel that they have an equal purpose in the team.

The crawlers will be get a rush from the closeup action and seeing their comrades fight alongside them, while the master player will be immersed by the sense of scale and direct hand tracking in VR, leaving each side with a unique experience.

Additionally, the setting and style of the game, the winding headquarters tower of a modern Asian underground organization in the hazy nights of a metropolis flooded by the contrast of bright neon signs, offer a game environment seen less often in dungeon crawler games.

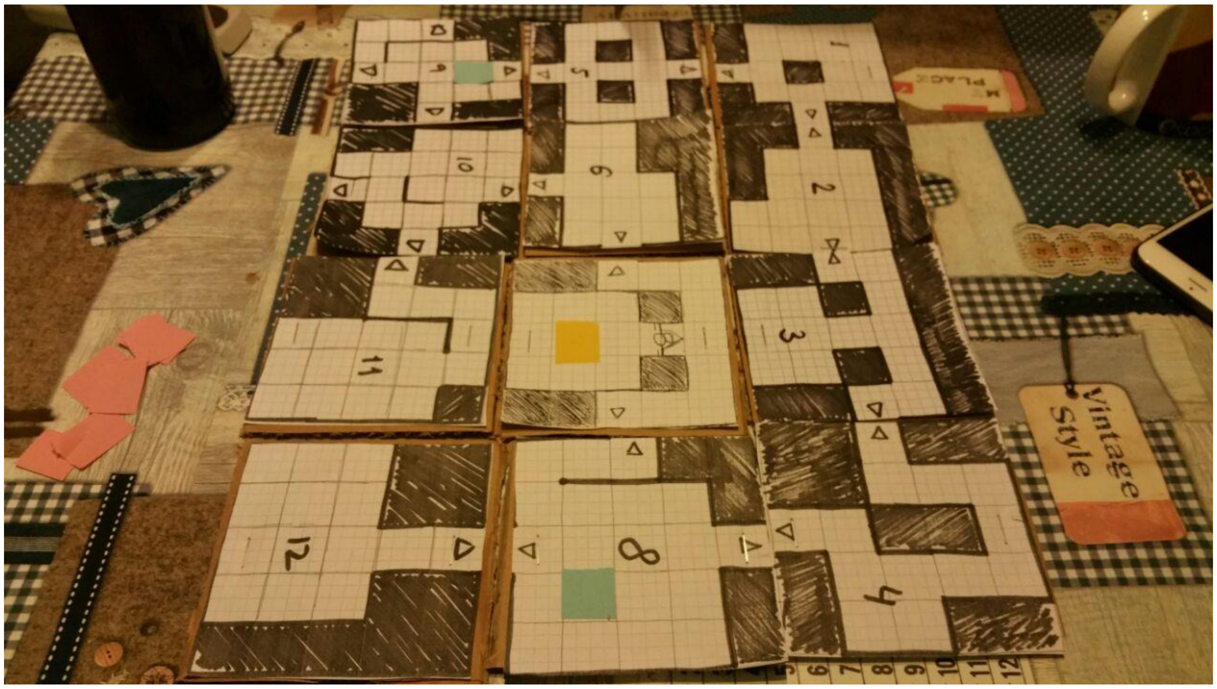
2. PAPER PROTOTYPE

2.1 Protoype Description

Basic Setup

There are three players in our prototype: two crawlers and a dungeon master. Additionally, a “game master” handles the tasks that will be automated in the digital game like enemies and environment alteration.

The crawlers and the dungeon master each have their own play-board composed of a 4x4 module grid. The modules are drawn on one side of the cardboard cutout. Each module contains a 5x5 tile grid representing one room of the dungeon.



Each of the tiles within a module can be either a wall (filled) or a floor. Adjacent floor tiles can also have walls between them to prevent movement from one to the other. Floor tiles at the edges of a module can contain a door connecting it to the next module.

A T-shaped divider separates the dungeon master and crawlers. This is to hide the maps of the crawlers from each other so they each have their own perspective of the dungeon. All boards are in the same configuration and orientation with respect the master’s point-of-view.

The dungeon master has four ability cards, featuring a symbolic representation on the front and a number from 1-4 on the back. Further props required to play the game are two dice (D6 & D4), a black marker and post-its in four different colors.

Entities on the map

Four distinct entities can take up a floor tile on the map. They are represented by differently colored post-its.

Crawlers are represented by blue and the numbers 1 or 2 depending on the player controlling them. Each crawler can move up to three tiles per turn, pick up collectibles and attack one enemy in an adjacent tile.

Basic enemies are represented by red. They deal 1 point of damage per attack, have 5 HP and can move up to 2 tiles per turn.

The boss enemy is represented by yellow. It deals 2 points of damage, has 20 HP and can move up to 3 tiles per turn.

Lastly, loot, represented by green, can be picked up by crawlers by walking over the tile. It will recharge a random missing ability of the master.

Game Setup

Before the game starts and the players can see, the game master prepares the playboards. They place players, enemies, loot and the boss on the boards. As mentioned in the basic setup, all boards are oriented to be the same when looked at from the dungeon master's perspective.

To simulate fog of war, crawlers can only see the module they currently occupy. All other modules are flipped over to hide their contents. The dungeon master can view all modules on his map but he can only see the location of the boss enemy, crawlers and loot.

Game loop

The turn order is:

1. Dungeon Master
2. Crawlers
3. Game Master

Before each of the other players' turns, the game master has to make their visible modules globally consistent. If a player moves to another module, it needs to be updated so that it represents the current state of enemies and loot e.g. the other player may have picked up the loot or killed the enemy there.

The dungeon master can perform one of two actions per turn: gesture to a crawler or use an ability. This restriction is placed upon him to represent the multi-tasking constraints the master will face in the final game where he will have to guide four players at once. The dungeon master also has markers that he can place on his map to remember enemy positions as well as keep track of crawlers. This is considered a free action for him.

If the master chooses to gesture to a crawler player, the master may ignore the divider to point to a position on the crawler player's map. This simulates the perspective of the crawlers where each of them can see the dungeon master from within the dungeon but cannot see what exactly he is pointing to unless it is close to them.

The master has four different abilities representing both direct and indirect skills: Buffs, debuffs, throwing a fireball or a healing orb. The master may not have more than one of the same ability. He will start out with none of these abilities and will only get one when a crawler picks up loot. Upon such an event, a d4 dice is rolled and an ability corresponding to the result of the dice roll is awarded to the master. This design decision was made to create some dependency on the crawlers for the dungeon master as well as to simplify loot placement through randomness.

A buff gives one targeted player double attack damage for that round. Meanwhile, a debuff will prevent one targeted enemy from attacking that round. A fireball will damage all entities within a module including crawlers. The damage will be based on a d6 dice roll. The healing orb will heal all entities in a module to full hit points. The last two abilities are module-based in order to emulate VR-induced inaccuracy.

The crawlers start by moving up to three tiles. Crawlers can only move through empty floor tiles or ones containing loot. Crawlers cannot move diagonally. When moving to another module, the crawlers have to move over the tile containing the door. The neighboring module is then flipped to reveal its contents. On leaving the door tile, the previous module is flipped face down. When moving, crawlers move their post-its to the corresponding tile.

When collecting loot, the crawler throws dice until one of the aforementioned abilities of the corresponding number can be awarded to the master. After this, the loot sticker is removed. If the master has all four abilities, the loot stays in place.

After moving, a crawler can attack an enemy in an adjacent tile. In this case, the player throws a D6 dice and deals that amount of damage to one specific enemy. The damage dealt is indicated on the enemies' post-it by marker lines. If an enemy takes damage greater than or equal to its health, it dies and is therefore removed from the board.

After dungeon master and crawlers have finished their turn, the game master will play the enemies in the players' tiles. Enemies will move towards the closest player. Enemies cannot move to other modules. After moving all visible enemies, each crawler will receive damage from adjacent enemies based on the damage of each enemy. Each enemy will only attack one crawler. As the enemies are played by the game master, she may use them as she sees fit. This emulates enemy AI in the final game.

End Conditions

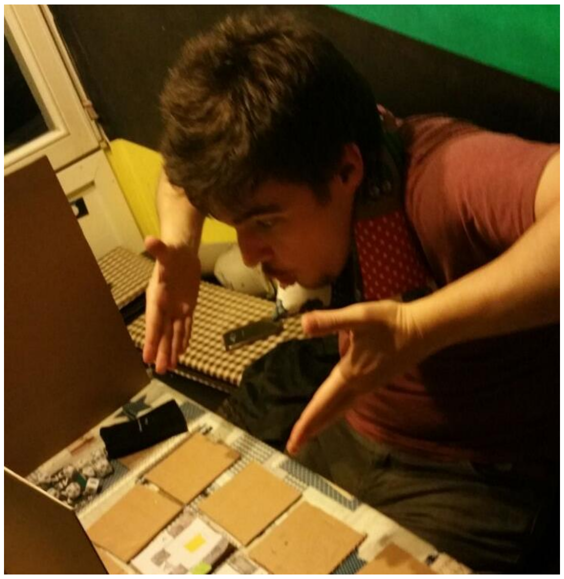
The game is lost when both crawlers are dead and won when the boss is killed, irrespective of remaining enemies.

2.2 Results of Prototype Testing

The current design of the game ensured co-operation between the crawlers and the master. Crawler players depended on the master for advice. However, crawlers still moved and explored the map of their own free will. The crawlers co-operated with the dungeon master to locate enemies and helped each other avoid them. The master players initially focused on getting a loot item so they could be of assistance in case of emergencies.

Once the crawlers were united, things became easier for them as they could support each other directly in combat. As such, they easily went through the dungeon and defeated the boss.

Fun Elements



The initial phase of the game, where the dungeon master was guiding individual players and helping them sneak around the map, seemed to be fun for all players involved. There existed a strong co-operation between the crawlers and the dungeon master in this phase leading to interesting interactions and ability uses. Most immediately recognized the need to serve each other's needs to win.

Dull Elements

Once the crawler players met up, the role of the dungeon master became less meaningful. Crawlers became more confident and rushed head long into combat. The challenge of surviving in the dungeon was significantly decreased and the master was mostly only needed to locate the boss and provide minor assistance in combat. In the meantime, he would skip rounds simply because there was nothing to do. Even though we pinned the problem down to the turn-based system, adjusting enemy threat, the master's abilities and map complexity may reintroduce elements that made the initial phase more enjoyable. Further testing will be done in the digital prototyping phase as the real-time mechanic will allow for a closer representation of the final game as opposed to the turn-based paper prototype.

Design Revisions

What we quickly noticed during design of the prototype is that we were not specific enough in our original pitch idea regarding exactly which entities the dungeon master should be able to see and how – that it should be boss enemy, crawlers and loot, but only enemies that can be seen by crawlers, how we could make sure they weren't abusing their abilities in combination with gesturing – that this would be taken care of by the natural constraint by micromanaging up to four crawlers and the inaccuracy of VR tracking, or how we should skew the balance between exploration and combat – that it is better favor exploration to promote communication with master.

We also noticed that it might be problematic for the master to see the boss from the beginning and being able to kill him directly with fireballs or directly regroup the crawlers there. Therefor it was decided to make the boss immune to damage when no crawlers are in the room. In addition, it might be useful to not show the boss to the master before the crawlers haven't regrouped.

Unfortunately, the nature of a paper prototype felt quite restricting when attempting to translate our core gameplay to a physical representation, be it by having to change from real-time interaction to a turn-based system, switching to a very strict "fog of war"-esque map system or needing a cumbersome game master player to act in place of a digital synchronizing server.

Nevertheless, the gameplay of the game is now more fleshed out due to our deeper discussion of individual game mechanics and player relationships.

3. INTERIM REPORT

The full dev log that this interim report is based on can be found here:

https://wiki.tum.de/display/gameslab1718/Dev+Update+1%3A+Interims

It also contains videos and gifs of our project.

3.1 On the way to a functional minimum - Nov 25, 2017

In the past two weeks, we got almost all of our functional minimum features done. As can be seen in the (already outdated again...) video above, Hikari no to so far has a working lobby system (based on the Unity Technologies lobby asset) that is set up for both local area play and online matchmaking using Unity's UNET services. A lobby can hold up to 5 players (1 master, 4 crawlers), with the host having "admin" capabilities like kicking players. For now, we assume that the host is also always the VR player, on one hand because the master is the unique core feature, as well as because it is likely that the VR machine inherently has enough processing power to handle the additional server work.

Next to the networked lobby, we of course also find networked gameplay. This too is powered by Unity's built in networking framework, albeit not based on a particular sample asset. As of this stage, the server handles AI computation, hit detection, and a handful of broadcast calls, while the clients mostly just synchronize entity transforms and only actively communicate with the server to handle player attacks. We try to always handle network communication efficiently so we can keep the traffic and latency requirements as low as possible.

The enemy AI went through two iterations by now. Our initial design was merely based on range checks and line-of-sight raycasts in order to detect the nearest visible and viable player target to attack, without regard for any more complex behavior like patrolling. But exactly that is what we felt was missing, perhaps not for the functional minimum, but for peace of mind. The second iteration now theoretically supports more modular behavioral patterns and in its current form offers detection, patrolling and attacking. There is still the question of how e.g. patrolling will work in tandem with random level generation, however we postpone those issues to a later date.

The crawler players are a rather simple setup for now, they can move for-/backward and strafe, freely rotate the camera - which also rotates the player and attack enemy entities.

Both enemies and crawlers use Unity's handy NavMesh feature set to facilitate pathfinding and level collision.

The VR master player is coming along nicely as well. We prioritize HTC Vive compatibility for now, but have at least verified that Oculus Rift is also compatible with our setup and should only need some tweaks to improve controls and adapt to possible constraints in >270° tracking. We found room-scale and hand tracked virtual reality play to be a very enjoyable and captivating experience, even when devoid of any meaningful gameplay in this early stage. Our Vive controller control setup as of now consists of a selection radial for master abilities and direct hand-pointing at intended targets - with a certain allowed deviation range to account for sensor jitter and player inaccuracy.

We are still heavily tweaking scaling of both the master and the dungeon, as we are not yet set on how gigantic the VR master should ideally be to be able to overlook a significant area yet still retain the ability to lean down to focus on encounters (and which potential locomotion requirements this entails). Additionally, we feel the dungeons need to be far wider and more open than the sample in the video in relation to the ground entities.

Overall, what is missing for us to reach the functional minimum target is to complete basic VR master interaction with ground entities, a more appropriate sample dungeon, a more properly tweaked sense of scale and a basic end condition/handling - instead of just endlessly respawning the crawlers. While working on these tasks, we are of course also starting to work out more detailed specifications for the low target, such as crawler classes, abilities, collectible items, master skills, and a more complete game framework, e.g. offering a more "complete" UI.

Unfortunately the embedded video does not showcase all the latest progress, but due to the daily rate of progress, it is futile to try and get every new change in there.

3.2 Low Target? Eeeh not quite - Dec 5, 2017

Dev Update - Jonas

My focus over the last week was to refactor and polish the Dungeon Master in VR. In addition I wanted to finalize the functional minimum which includes a buff/debuff ability for the master and a synced appearance.

The master can now select either "buff" or "debuff" from his radial menu, and apply it to crawlers or enemies as an effect (Artem and Inshal have more on that) by pointing at them. In order to implement a soft selection, targets will be chosen based on who is closest to the pointing direction and still in range. The selection is visualized by a bezier curve (blue for buff, orange for debuff) that's being shot from the Master's hand to the target. To give the master a better feeling for his actions there is also force feedback for most of his actions.

In addition, the master is now visible to the crawlers (and he's so cute! :3) which should enable primitive communication. The headset and the controllers are now tracked by a network synced visual representation, buff and debuff rays are also visible to players.

Next week I'll try and find a fitting hand model for the master, preferably a rigged hand mesh that suits the art style. This will allow us to implement advanced master-crawler communication using different hand postures and some better visuals for abilities in general. Additionally, physics based abilities (heal, fireball, etc.) are to be explored.

Dev Update - Artem and Inshal

We were working on the baseline framework that supports:

- Different AI behaviours for enemies
- Different weapons for enemies/crawlers

- Various effects: buffs, debuffs, abilities
- Character stats, classes
- Simple UI for crawler

While we already gave some comments on AI patterns earlier, now we would like to dive into the effects system and the abilities and classes for crawlers. After a fruitful discussion, our team came up with four classes for our future release: soldier, infiltrator, berserker and... tank? Anyways, we don't know the final names for our classes yet, but we outlined the main skills and gamestyle that every class would have. In the interim demo we are ready to present two prototype classes, each of them has two active and one passive ability.

Soldier + Berserker

Soldier has a rifle that shoots. Being ideal for killing enemies at a distant range, she can greatly damage the incoming army before it gets too close. Additionally, she possesses the unique ability to emit destructive laser beam that goes through walls and enemies!

Since we haven't been very certain about how to implement other abilities for the soldier, for the purposes of the interim demo we spiced up the soldier class with some of the berserker's abilities. Berserker can gain adrenaline for killing enemies and when the adrenaline bar is full, she can go RAGE: katana replaces the gun for a while, and instead of fighing from afar the crawler runs and destroys everything that she can reach. Neat!

Infiltrator

He has a short sword and he is not strong at all. But his abilities are superior: firstly, he can go invisible and sneak past the enemies; secondly he can detect enemies hiding behind the walls. It makes this class perfect to complete the game killing as less enemies as possible. However, there is a reason why this class is called "infiltrator" and not, let us say, "pacifist" or "enemies lover": his special passive ability allows to sneak on an enemy and back-stab it with 4x damage!

For the enemy detection, several passes of rendering are used, when some enemies are drawn again with different shaders and the images are blended. At the and, we implemented a nice utility that allows us to highlight labeled parts of the scene in various ways (or apply more complex effects to them) with only a couple of scripts.

Technical notes

Up to this moment, we have a flexible system (with modular approach) that allows to add various types of buffs, debuffs and abilities. From the architectural point of view, everything that can be applied to a crawler or to an enemy (or to any other hypothetical character in the level) is basically one entity: effect, which can be enabled or disabled. So, an active ability is nothing but just an effect that crawler can apply to itself under certain conditions. Buffs and debuffs that dungeon maser gives out to crawlers and enemies are also effects. Passive abilities are effects that are enabled in the beginning of level and never disabled. Traps and some enemies can also, theoretically, apply effects. Every effect, its cost and duration, as well as its icon and name, can be set up independently from the game logic, which makes it easy to add new effects and tweak the game balance. Another good thing is that we made some effort to sync enabling/disabling effects over the network and reduce the technical burden while designing and implementing our game ideas. Effects are stored as the Unity Assets.

For now, every effect applied to a crawler, is automatically shown on the UI. It helps with debugging and looks good. Some stats, like health and adrenaline for the berserker are also exposed in the UI.

See you on interim demo.

Dev Update - Paul

Lobby work + networking

Half my work since the last update pertains to the lobby system and UI. For starters, the HMD checks were broken and out of sync before. Now they properly detect if and if so which type of VR headset is connected and displays as much in the lobby player list. The point of this, theoretically, is for everyone to see who is using a VR headset and for the host to be able to pick who of the eligible players will be the VR master. However, for the foreseeable future this pick will not actually be used, as there are still inconsistencies in the networking for a non-host VR master. Additionally, this might open up avenues of potential VR crawlers as well, which are absolutely not planned in the current schedule, but are a possible extension for the future.

Second, we needed some sort of class selection for the crawlers. Now the design decision was to put this selection into the lobby instead of the game level, primarily to simplify development as it didn't need an entire new UI for the selection. Secondly, we feel it is advantageous to figure out a team composition before the match starts, both to save time and to avoid potential dropout delays after people are unhappy with the picks and leave a match. There are still a few troubles related to accurate hiding and showing of other players' selection depending on VR master status etc, but these are basically just relevant if at some point we decouple the VR master from the match host. Plus, the backend implementation is not particularly clean with respect to how the UNET system usually handles player spawning, but it works pretty sleek for our purposes.

Third, a basic recolor of the lobby UI made it a slightly better fit for the theme and setting of our game.

As a side activity, there was a constant lookout and troubleshooting for possible networking bugs and sync issues.

End condition + UI

Another addition to the game was a basic end condition setup. Now if all crawlers die or all enemies are killed, the match is lost or won respectively. In either case a simple message is displayed on the screen and the player gets the option to return to the game lobby. An equivalent setup for the VR master is actually still missing as we did not manage to implement a VR-based UI interaction system on time.

Interim demo map + scaling

The third big task was to tweak scaling of the dungeon with respect to crawler and master size, and once a suitable scale is found, to build a small demo level for the interim. Overall, hallways and rooms are now about 4 to 6 times as large as in the first prototype video. The master was also scaled up by about a third. The demo level is an attempt at creating a believable office floor structure, with a larger cubicle area, a conference room, a recreational section, a large server room and a central court with some nature objects to provide a nicer work atmosphere. The remaining rooms don't have a specific function, but in light of our plan to replace the fixed structure with dynamically generated levels, there is little point in fleshing out a lot of detail in this demo level. One aspect that fell short, however, is giving the level a high-rise feeling with large windows and perhaps a steep view down onto a larger city. As it stands, the "dungeon" gives a more realistic sense of local scale and should be a good starting point to cut modules from for our planned level generator.

Target practice

To recap, we believe we have certainly completed the goals of our Functional Minimum and are on a good way to our Low Target. Our core gameplay system are mostly in place, the majority of Low Target tasks ahead are related to creating more enemy variety, complete four different crawler classes instead of two, more fitting 3D models for our entities, more direct crawler signalling and a first foray into a wider level setup. In short, mostly visual and diversity efforts.

4. ALPHA REPORT

The full dev log that this alpha report is based on can be found here:

https://wiki.tum.de/display/gameslab1718/Dev+Update+2%3A+Alpha

It also contains videos and gifs of our project.

4.1 Testing, Bug Fixing, Refactoring - Xmas 2017

It turns out our game concept, a networked multiplayer game with VR concepts, is quite a pain in the a$$ to test and debug.

To test the networking now, one needs multiple instances of the game running, before we added in VR this could be done on one machine by starting multiple instances of the built game. The problem is that SteamVR (and probably any other VR API for that matter) needs to be accessed exclusively by one application. So when we want to test VR networking, starting a new instance of the game will automatically close the other one. As a consequence, we can only run one instance of the game (either editor or build) on one machine at once. Making matters worse, Unity networking seems to be pretty fragile when it comes to local differences between two client versions, differences that both Git and Collab seem to ignore, so in practice master networking testing will often be handled by building the game and sharing the build over network or portable storage with the other tester. The fact that thanks to fancy lightmaps and assets the builds can be almost a gigabyte now doesn't really make it easier on us either. But enough whining already.

We spent pretty much the entire week around the interim finding, tracking down and fixing bugs that occurred in weird or hard-to-test situations, like 5 players playing at once or players leaving and entering.

Most of December then was spent implementing the missing features for the low target (more on that in the next update) before sliding into our (well deserved?) Christmas break. :3

4.2 Baby Steps - Jan 9, 2018

Dev Update - Paul

Visuals have changed considerably since the last demo, plus a selection of changes to the UI and smaller fixes

Good old Kleinvieh

Among the progress on smaller issues was the addition of VR pointing, currently only used to return to the lobby after a match has ended. This required setting up a 3D pointer as a Unity GUI compatible pointing device.

The crawler camera got a small offset to achieve a sort of "over-the-shoulder" look, as well as a crosshair so players actually know where they're shooting.

What's on the menu?

One of my bigger additions besides the core mechanics that aids in making the game feel more "complete" is a main menu that extends beyond just a lobby. This main menu is relatively simple but brings all core items a player can expect. There is a main screen from which the user can pick the three different panels:

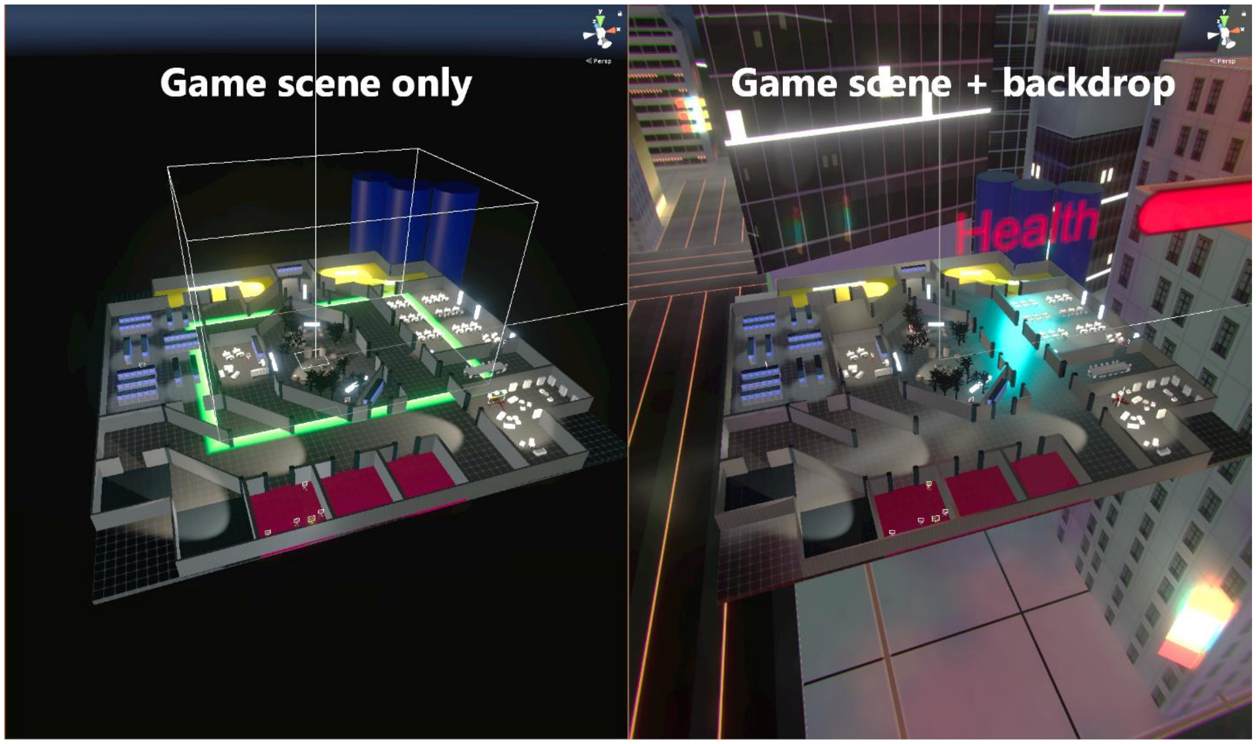
lobby, settings, controls. The lobby panel is the previous lobby as it was before. The settings panel offers the player a small selection of visual settings to change if they want better visuals or better performance, including resolution and quality preset. The controls panel simply shows the control mappings for all three supported input methods: mouse and keyboard for crawlers, Rift or Vive for master.

Visual feedback

One issue we kept seeing while playing as crawlers was the lack of visual feedback when hitting an enemy or getting hit by one. To mitigate this to a degree, visual hit effects were added, simple particle effects that visualize where an entity was hit, as well as screenshake. The screenshake does exactly what the name implies, making the camera shake whenever an "event" like a shot or explosion happens close enough to the crawler. While another layer of visual feedback on the UI would be very helpful, the two additions already give a much better idea of when and why the player is losing health.

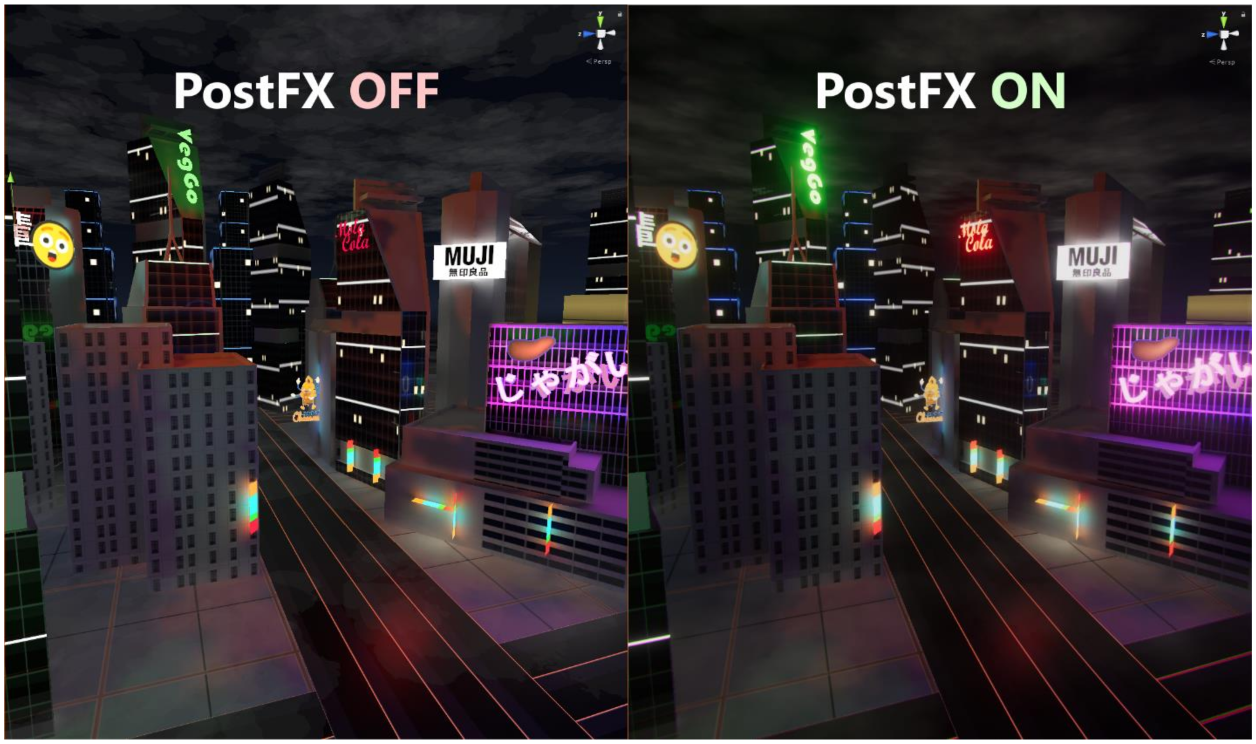
Props to the level... Well, visuals in general

Visuals were another big change for the feeling of the game. The interim demo still only had the level without any surroundings, unrefined materials and simple lighting. For the alpha, I added a more sophisticated backdrop. For one in functional nature: the backdrop would be the same for any level, so integrating it directly into the actively played level would be a waste of memory, mapping times and storage. Instead, Unity supports so-called additive loading as well as "object persistence". The idea was to load an arbitrary game scene and then additively load the backdrop scene. Turns out that additive loading breaks lightmap assignment and completely nukes lighting of the backdrop. But object persistence doesn't, so now the system is a bit more clumsy and less optimized, but works the same on a high level and lighting remains functional.



The other part of the equation is the actual backdrop. Our setting is that of an asian downtown district, so the backdrop needed skyscrapers, neon lights and a dark, broody sky. The skyscrapers were taken from a few packages in the asset store, but materials were adapted to include more emissive detail. A variety of neon signs were created, some designed from the ground up, some taken from existing logos. The skybox was changed to convey a dark night and a slowly advancing cloud layer added. While a soundscape and some ground movement for the backdrop is still missing, it already gives a much better idea of what the setting of the game is. This step also included some experimenting with lightmapping setups and emissive mapping.

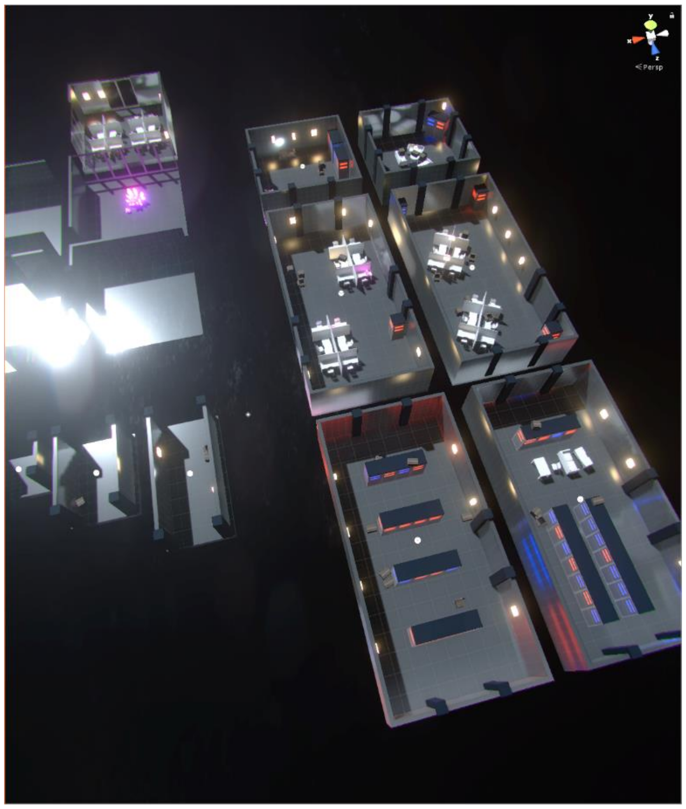
Another core component to the visual style is post-processing. Using the Unity post FX stack, I added tweaked bloom, lens effects, AO, HDR mapping, filmic tonemapping, screenspace reflections and depth of field. This went a very long way in conveying a fitting look for neon signs, fire, and other bright effects as well as enhancing contrast to give a stronger darker look to the game.



Room prefabs for random map generation - inb4 wreck

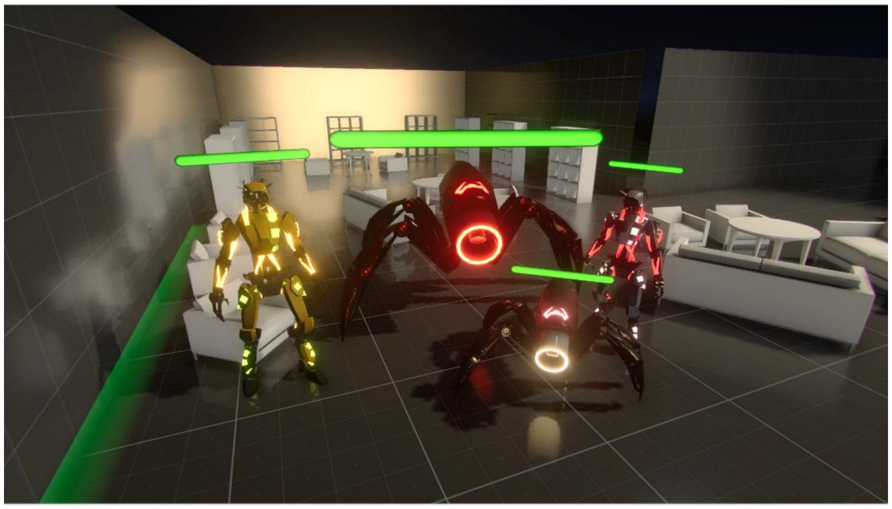
To elevate the random map generation from a prototype to a usable tool, we needed prefabs for the modules that the generator uses. Rooms, corridors and junctions. The next issue that presented itself was that these rooms would need to be filled with props: furniture, lights, decorations etc. Manually placing these would be a chore, especially if we want some variety, so instead of doing that, I introduced a Random Object Instancer. It's a gameobject to be placed in a room, a certain maximum boundary volume is specified and a list of allowed objects is added. Then as an editor tool, an arbitrary number of these ROI can be selected and at the press of a button they place a random object from their assigned lists into the scene. This way the designer only needs to specify the basic layout of items in a room but visual variety can be changed with a single click. Some of the individual props for the ROI were taken from the asset store, some simpler ones were created manually.

Unfortunately, it turned out that the prefab rooms and the map generation tool in its current state are not quite compatible, and couldn't be fixed in time.



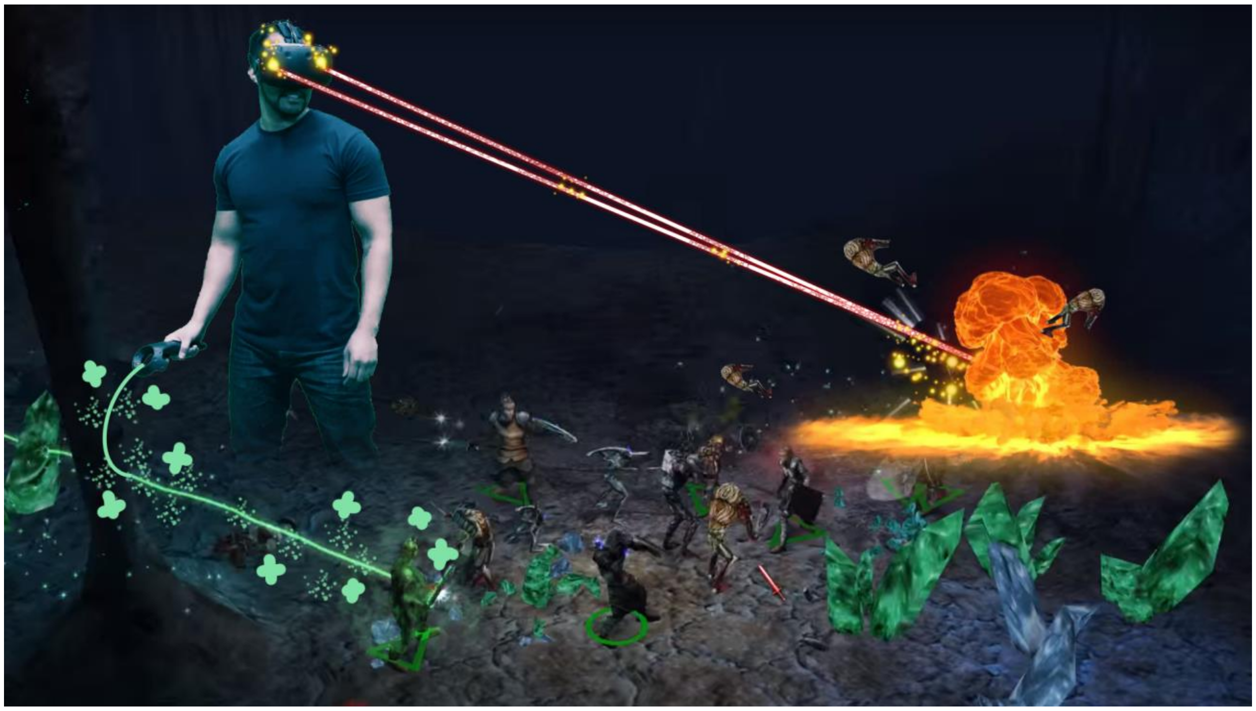
Gameplay balancing

Some changes to gameplay balancing were introduced, such as a significantly higher movement speed and somewhat higher attack rate to make crawlers feel more agile and better equipped to evade enemies if necessary, as well as increasing the level of excitement a player experiences. To mitigate the fire rate somewhat, damage numbers for the crawler weapons were tweaked. ((Artjom:)) Additionally, more enemies and a second enemy type were added so the average player would actually need the help of his fellows and the master. On top of that, the level now contains a boss with more health and more range.



Dev Update - Jonas - "They grow so quickly!"

The master has come a far way from my first concept mockups, he now is even prettier, can show crawlers de wey and burn enemies or heal the crawlers' deep wounds.



VR MASTER - NOW IN RGB!

Instead of the old additive particle material, the master now has a bunch of glossy emissive material for each usable ability that he will smoothly interpolate between.

With a new, carefully selected colour palette for abilities, this gives both the crawlers and the master an intuitive visual feedback and it looks extra sweet with Paul's new post processing effects in place. In addition, buff and debuff have been enhanced with cute little particle systems and some more smaller visual tweaks have been applied to the master. For some reason the new systems cause weird flashes when the master blinks sometimes.

DU YU NOW DE WEY????

Initially, we planned to give the the master a pair of rigged hands, that allow him to perform different gestures and postures like pointing at certain targets or flipping the bird (here at Pick One we believe in flaming).
This idea was abandoned for multiple reasons though. First and foremost, nicely rigged, low poly hand-only models that are suitable for VR are pretty hard to get your hands on (pun intended), the good ones are in the asset store with double-digit price tags and everything you can find for free is pretty much useless. With the limited artistic skills in our team, modelling and skinning a pretty hand mesh and animating nice gestures in postures was also not feasible. On the other hand, we only really need to point at stuff, hence most other postures and gestures would pretty much only be nice and funny gimmicks. In addition, the concern arose that humanoid hands would not quite fit the boxy/digital cuteness of our

master. Therefore the decision was made to implement a simpler and more robust solution: A ***3D*** ARROW!

Surprisingly, 3D arrow assets are just as hard to ~~steal~~ find as hands. Because of that I took on the deed of spending an afternoon teaching myself Blender and modelling this beautiful mesh. In game, the master can turn his hand into yellow glowing arrows by pressing the grip buttons. When we're adding sounds, the arrows will also emit a loud "**PING**" when appearing.

The idea is to use this minimalist concept to make crawler-master communication both simple and exciting.

Throwables

The big master feature for the low target was throwable abilities, more specifically fire balls and heal orbs. The idea was to give the master powerful area effects that are limited in their usefulness by low precision and the fact that they affect crawlers and enemies equally. In practice this means, fire balls will also damage crawlers and heal orbs will heal enemies. The master therefore has to decide strategically when to use these abilities and be extra careful when throwing.

To avoid spamming, throwables need to be charged before usage. Throwables with lower charge will be smaller and less powerful. When selecting a physics based ability, the master's off-hand becomes a "pool" that he needs to suck the throwable's charge out of. Visual and haptic feedback facilitate this process for the master player. The idea is that once collectibles have been implemented, the master will only have limited total charge for his abilities, intuitively represented in the pool's size.

The strength of the effect on an entity is dependent on three different factors: the aforementioned charge of the throwable, the distance to the detonation and finally the velocity of the throwable. The latter was introduced to encourage masters to throw the abilities properly instead of just dropping them right next to where they are needed. In addition to the applied effect and a badass explosion there is now some immersive screen shake thanks to Paul.

Next steps

With the alpha realease this week and about one month of developement left, focus from now on will be more on polishing and bug fixing rather than adding a bulk of new unfinished features/abilities, ending up somewhere around the desired target.

For me this means that I want to end the infinite spam of abilities for the master by adding an automatically recharging resource for buff and debuff and pick-up based resources for the throwables. Limiting the enemies' visibility for the master also seems like a doable and gameplay-relevant task. In addition, a simple off-hand-touch-pad-based teleportation solution might become important for larger dungeons. The idea is to have a less overpowered master that makes gameplay a bit more challenging for both crawlers and the master.

Dev Update - Inshal

The Sumo

A new player class was added to the game. Do you feel those enemies are too tough? Do you want extra health to be able to tank some hits for your friends as they run away and abandon you? Do you want to be able to erupt in anger and their lack of anger? INtroducing, the Sumo. The primary gameplay purpose of the Sumo is to act as the vanguard of the group and keep his allies from harm by attracting attention of the enemies. His primary attributes are:

Extra health due to having more mass than any other entity in the game (perhaps even the master!)

A basic attack that deals damage to everyone in front of him

Sumo Blast! Burst forth in anger and deal distance based damage to enemies around you!

Randomly Generating Maps

Creating a new map layout can be tiring, stressful, boring, tedious (insert more negative adjectives here). So rather than do it ourselves, we got a Unity to do it for us. We have a simple agreement with it: We give it prefabs of the map pieces we want to use and it generates the layoutr of the map and connects those pieces together. Then we can apply some light modifications if we feel like it and play the game! If you feel like nerding out a bit and getting a sense of the algorithm, read on!

The algorithm requires map objects. There are three types of map objects at the moment: rooms, corridors and junctions. The default connection rules are as follows:

- A room may only connect to a corridor
- A corridor may only connect to a room or junction
- A junction may only connect to a corridor

These rules can be changed on a per map oject level so you can have a room that connects to other rooms if you'd like (say, a trap room connected to treasure room).

Each map object also has to have exit markers attached to it which are basically empty game objects giving the location and orientation of where to connect the next rooms. These usually come with walls that can be enabled or disabled based on whether the exit is being used or not.

Finally, map objects must also have a collider covering the entirety of their geometry for overlap checks when instantiated by the algorithm.

The algorithm itself is simple to understand. The main steps are:

- Create a list of all exits that are not in use
- Try to add a map object to that exit
- Based on allowed connections of the map object
- Randomly selects a map object from allowed list
- Connection chance
- Overlap checks
- Map boundary checks
- Rinse and repeat

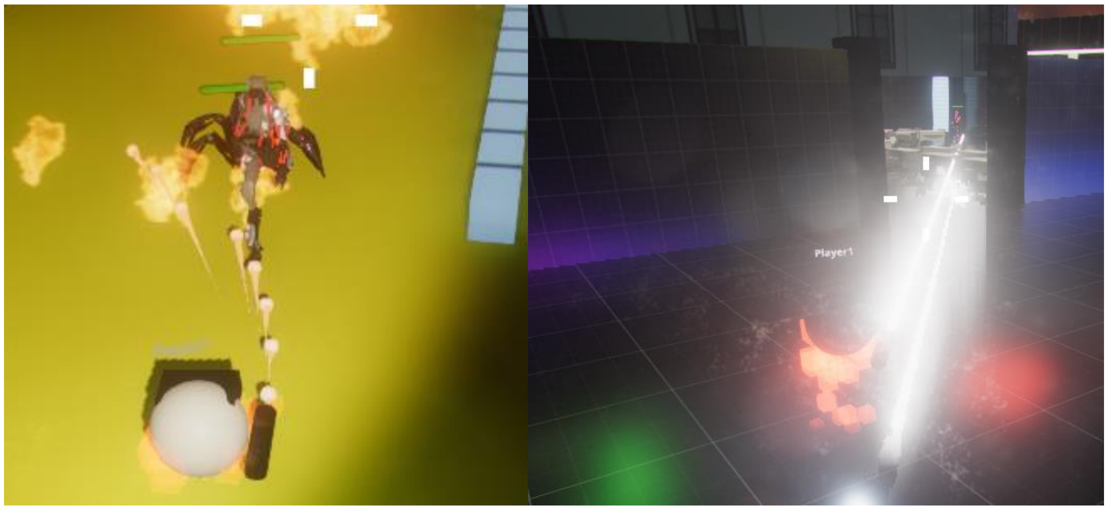
We added several features on top of this simple algorithm such as chance for the algorithm to simply drop the exit and make the room a possible dead-end.

Dev Update - Artem

Classes

Inshal already told you about the sumo class, I would tell about the rest. Remember, we had adrenaline soldier class in our interim? So, now it is split in two separate classes with different roles: **soldier class** and **adrenaline class**.

Soldier inherited his glorious sniper beam that strikes through walls. What's more, now he obtained a new super-attack that fires a blossom of bullets in front of him, that makes him less vulnerable in close quarters:



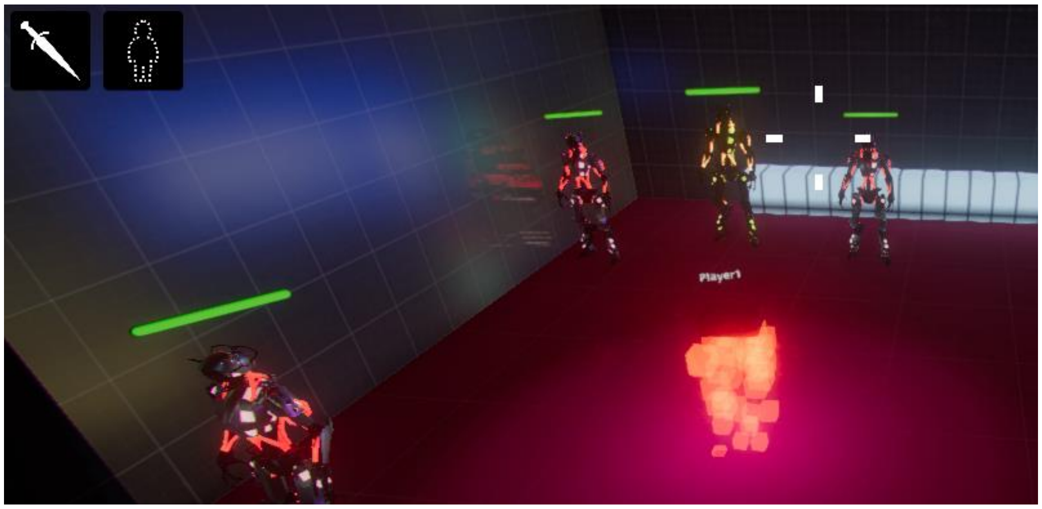
Adrenaline crawler inherited, well, adrenaline! He gains adrenaline from kills. When adrenaline is enough, he can become an overpowered katana master for a while. His basic attack is changed to shotgun to make him more distinctive from other players and highlight his "rage" behaviour.

So, along with sumo and rogue, we now have four playable classes.

Enemies

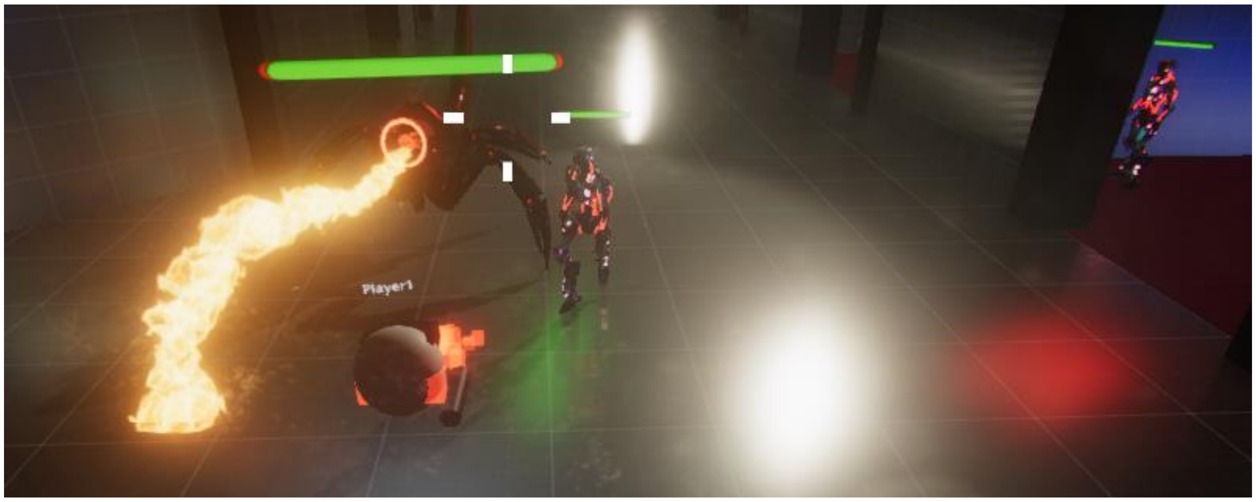
Another thing necessary for a complete game is enemies. We already had a dummy enemy with a simple AI for our interim, but now we have four of them, with two unique animated models.

The most frequent enemy is **small robot-soldier**. It is the weakest enemy that comes close to the player and uses blunt attacks. This simple enemy has a bigger version, with more health and damage, **big robot-soldier**. We are probably going to make a big version of soldier more rewarding for players in the future, allowing them to drop useful collectables.



More interesting are **scorpions**. They look dangerous and use flamethrower as their main weapon, which inflicts serious damage. Fortunately, this attack is pretty easy to avoid.

Finally, there are **scorpion-boss**. To defeat it is the main objective of our levels. It has 3 times more damage and enormous amount of health. Just look at the pic:



Next steps

We have working collectibles; however, we haven't tested them properly yet, and they won't come in the alpha version.

Also, we are almost ready to add bonus enemy that shoots from afar.

Besides that, our priority task is to **balance and tweak our gameplay, fix bugs.**

5. PLAYTESTING REPORT

The full dev log that this alpha report is based on can be found here:

https://wiki.tum.de/display/gameslab1718/Dev+Update+3%3A+Playtesting

It also contains videos and gifs of our project.

5.1 Last Tweaks for Playtesting - Jan 17, 2018

Dev Update - Jonas - Finishing Touches

Over the last week, the master received his last finishing touches: teleportation, ability resources, bug fixes and further visual upgrades.

Teleportation

One major design question was how to deal with varying map sizes as well as different VR room sizes (steam VR play area). The idea was to encourage the master player to walk (actually walking, taking advantage of the large HTC Vive play space) around on the map and support crawlers in different corners of the dungeon. In order to preserve a consistent master player scale, some form of teleportation needed to be implemented. There are many existing solutions to teleportation in VR, most famous the Steam VR pointing based teleportation, which will teleport the player to the exact position that he points to. As this would allow the player to simply spam teleportation instead of actually moving in VR space, this didn't seem like a suitable solution for us.

So finally, we came up with a tile-based teleportation system. It allows the master to teleport to adjacent tiles of the size of his play area. In order to be effective, the master still needs to move around within the tile.

Selection of the teleportation target is done with the radial menu of the off-hand, in order to keep teleportations globally aligned, the radial menu was modified to a "compass radial menu". In addition, radial menus now also support the "no selection" if the selection (either trackpad or analogue stick) is too centred, this is needed to stay in the current tile. The current target tile selection is also visualised by a marking on the floor.

Ability Resources

As the master player was way overpowered before, usage of abilities is now limited by sparse resources. Both buff/debuff rays and physics-based abilities now use resources that are either regenerated over time or are picked up by crawlers. As of now, this means that a fully charged master can buff and debuff for 5 seconds and throw 3 heal orbs and fire balls before needing to recharge. The amount of available resources is intuitively represented in the size of the ability pool or the casting hand.

Bug fixes and further tweaks

Over are the days of the exploding eyes, turns out that any mesh scaled down to 0 will cause weird lighting bugs. Also gone is the "infinite buff" bug, that caused buff effects to stack and not be removed, making the affected crawler an invincible one-hit kill machine. In addition, quite a few non-master related bugs that bugged us for months got fixed.

Further improving master visuals and following the style set with the new particle systems last week, buff and debuff rays now glow more neon-like, the casting hand emits fancy particles as well and the entire master starts glowing when applying a buff or debuff. In addition, there is now a helper ray indicating the aiming direction of the rays in case the player fails to aim at crawlers/enemies properly. Some less visible changes and tweaks have been applied to the radial menus, hand visualisations, particle systems and materials in general.

Next Steps

All in all, i'm pretty happy with how the master turned out so far, but of course there's always room for improvement. Early playtesting this week revealed, that the crawlers are often having a hard time finding the master and grabbing the master's attention is not really supported at all as of now, and vice versa. To solve this, we are planning to add a 3D arrow to the crawlers UI, that always points to the master that makes a ping noise and lights up yellow when the master is pointing somewhere and takes the master's color otherwise. A key to point the crawler camera at the master as well as one for pinging the master back seem like valuable features to implement.

In addition, the master seems a bit mute as of now, with first sounds in the game. Adding some idle, ability and ping sounds will definitely boost immersiveness and localization for both crawlers and the master.

Of course, it goes without saying that there never is too much eye candy. Adding in ground effects for crawlers affected by abilities, more particle systems, more light sources, nicer and more polished, textless radial menus are currently not too high up in the priorities though.

Dev Update - Paul - Touching finishes

Over the last week the menu, visuals and crawler also received some finishing touches.

The days between the Alpha presentation and now consisted mostly of bugfixes, first playtesting and loads and loads of tweaking.

Visual tweaks revision 209124683 'n' a half

I mostly tweaked visuals and kept playing over and over again to find bugs and issues. One of the most prominent topics were visual tweaks. For one doing a new lightmapping pass, but this time with the backdrop loaded additively as well as using reflection probes, which also fixed the additive scene loading. This brought with it the possibility to bake occlusion culling into both levels, which helps improve performance. Other than that, many particle effects and materials were tweaked or redone to increase visual fidelity, such as a sizzling effect for the laser beam or sparks flying when an enemy is hit. Post-processing effects were also tweaked to tone down brightness overshoot.

UI tweaks

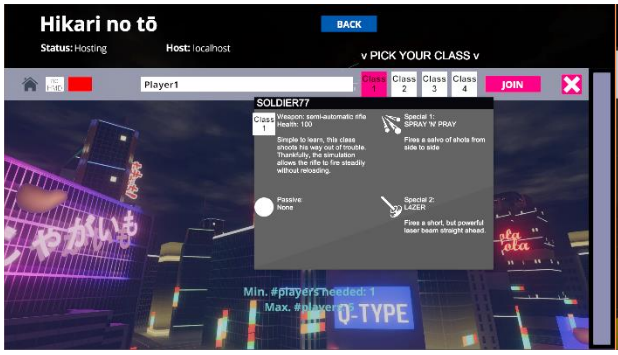
Changes to the UI were made as well.

First, in the main menu, the settings section was expanded to offer audio volume control, the option to show a VR controls reminder when playing VR and a - admittedly gimmicky - system info, which really only shows your machine's core specifications as an orientation. On top of that, the lobby screen now provides a brief rundown of a class' abilities and statistics when moving the mouse over the class icons and a loading screen after the match start countdown has reached 0, as the previous barebones loading could be mistaken for the game freezing.

Second, the crawler UI was revised, moving the ability status icons to the center of the screen and increasing the visibility of the attribute bars. The crosshair was fixed to show up in the correct position and was changed to a dot instead of a cross since we noticed that the cross suggested to players that this was a gun spread indicator when really it is just an indicator of gun direction. The cursor was fixed in that it is now hidden while playing and only shows up in menus. One noteworthy addition was an arrow at the top of the screen that

always shows where the master is, as playtesters complained it was hard to figure out where the master is at all times. I also introduced a death screen, since up until now dying as a crawler simply meant your control over the character would stop and a single nametag change would appear. Now, a message about your death is displayed on the screen and the camera zooms out into a total overview of the level.
At the start of a game, each crawler gets a short notification on what the objective of the game is and a brief hint on how to achieve it. The master now sees a large circle around each crawler and their nametags and healthbar are enlarged, making it easier to spot them in a crowd.

Additionally, there is now rudimentary UI handling of disconnects. Our game, unfortunately, cannot gracefully handle mid-match dropouts, so up until now if someone disconnected from an ongoing game, no one else would notice until AI or statistics eventually broke and the game couldn't be finished. With this change, if a player disconnects during the match, all other players are notified about this event and informed that the current match cannot be finished and they would need to return to the lobby. If the server disconnects during the match, players are put back into the lobby and shown an error notification telling them that the connection was lost.



SOLDIER77

Class 1

Weapon: semi-automatic rifle
Health: 100

Simple to learn, this class shoots his way out of trouble. Thankfully, the simulation allows the rifle to fire steadily without reloading.

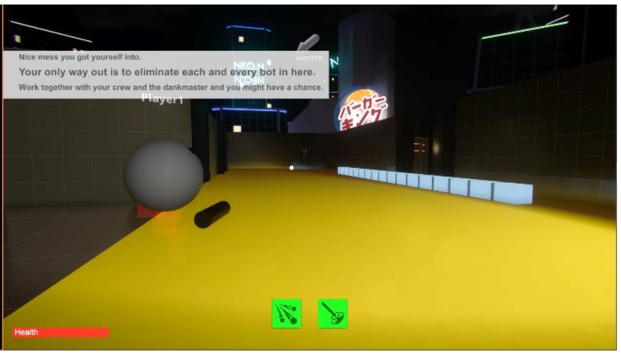
Passive:
None

Special 1:
SPRAY 'N' PRAY

Fires a salvo of shots from side to side

Special 2:
L4ZER

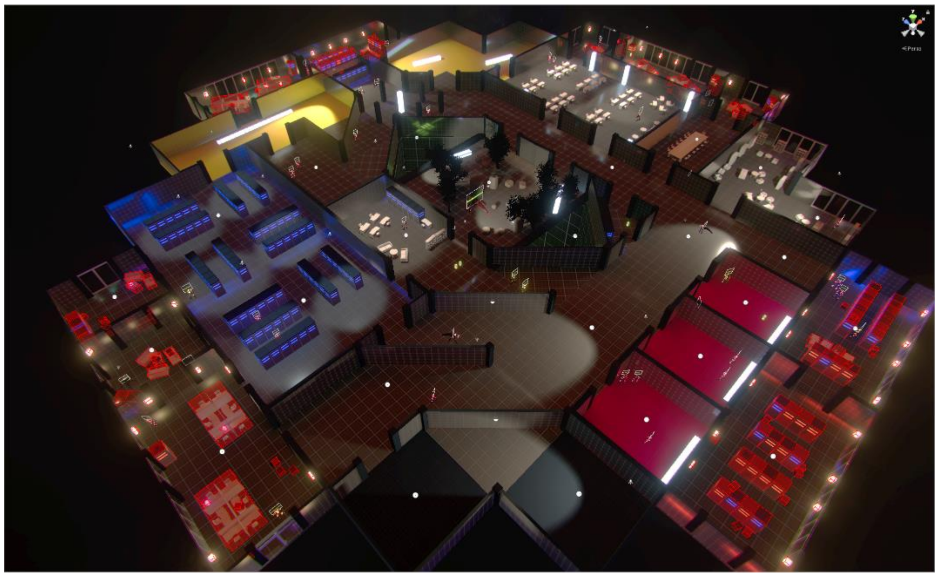
Fires a short, but powerful laser beam straight ahead.



Nice mess you got yourself into.
Your only way out is to eliminate each and every bot in here.
Work together with your crew and the dankmaster and you might have a chance.

Scene adjustments

What we noticed when playing the Alpha version was that, for more than two players, the existing level was definitely too small, so it was expanded by one row of rooms in each direction, this time even using the updated room prefabs. A simple VR controls reminder was added to the backdrop level which, when enabled, simply shows the VR controls inside the game view, useful for novice players.



Audio, at least halfway there

One important addition was sound effects (all lifted under various licenses from online sample databases). The game already had simple background audio for a while, but no sound effects for anything else. This milestone sees the addition of ambient sound for the backdrop: traffic jams and some howling wind, and some sizzling of neon lights in the main scene. Every crawler attack now also has a sound effect, from gunfire sounds of the soldier, over supercharged laser effects, singing lightsaber hum of the infiltrator to the satisfying pellet spit and reload of the berserker's shotgun. Similar goes for enemy hit effects, which now randomly fire off one of several damage sounds.

A caveat to sounds is that not all things have one. For example, enemies are still completely silent, not even their attacks give off the slightest note. Neither are there any sounds to specific match-specific events such as starting or winning, nor for the clicks in the menu. This is unfortunate but is unlikely to change drastically.

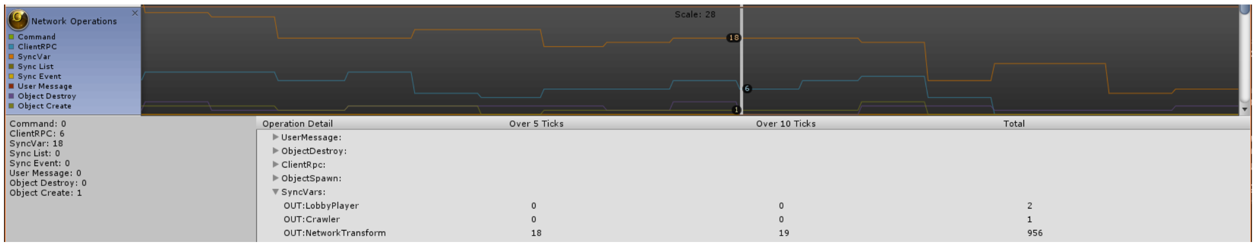
Player balance

A few tweaks to player balancing were made, including a change in camera offset and distance as well as some weapon statistic changes and movement speed.

Network performance

This is a tricky and nerve-racking topic. Here's some necessary background on networking as we use it: The Unity UNET framework uses relay servers to transmit data between clients and servers. These relay servers have a bandwidth limit imposed on user traffic, likely to avoid abuse. This bandwidth limit is realized as a "pool". The added bandwidth limit is 4KByte per second per connected client. So every second, 4KByte of "data volume" per player are added to the pool. Additionally at the start of a session, the pool is initialized with roughly 2 minutes worth of limit. If at any point the growing pool is exceeded, all clients are disconnected from the relay server. This means that a UNET enabled game usually needs to keep their traffic at or below 4KByte per second per client.

Obviously where I am going with this is that it turned out we were not within that limit. We don't have precise numbers, but our traffic exceeded the limit so much that, with a full match of 5 players, we would see reproducable relay disconnects after less than 4 minutes. Ironically enough, in our first attempt to lower bandwidth usage, we instead increased traffic tenfold and disconnected after 25 seconds. The second attempt actually saw reduced traffic. Now we still don't know precise numbers, but we believe we are close enough to the limit to not see disconnects during an average match. These attempts consisted of reducing synchronization rate for certain network elements, reducing the amount of data sent (for example only synchronizing the Y axis rotation instead of all three axes), compressing some of the data and more. One experimental attempt - that is not actually deployed in the current release - sees a more compute-heavy approach of selecting the bare minimum of necessary position and rotation data, converting it to "smaller" data types, synchronizing, and then reverting the process on the other end. In theory, this could reduce minimal message content size per transform update from 16 Bytes to 3 Bytes. However, the message headers still remain as they were and actually represent an even larger fraction of traffic. We have no solutions currently in development for this issue, we basically simply hope that our efforts so far suffice.



```
Command: 0
ClientRPC: 6
SyncVar: 18
Sync List: 0
Sync Event: 0
User Message: 0
Object Destroy: 0
Object Create: 1
```

Operation Detail	Over 5 Ticks	Over 10 Ticks	Total
▶ UserMessage:			
▶ ObjectDestroy:			
▶ ClientRpc:			
▶ ObjectSpawn:			
▼ SyncVars:			
OUT:LobbyPlayer	0	0	2
OUT:Crawler	0	0	1
OUT:NetworkTransform	18	19	956

Performance testing

Lastly, a lot of performance testing and analysis was done. The game was tested on a wide range of hardware (from an Atom x5-8350 + HD Graphics up to a Ryzen 7 1700 + GTX1080Ti) to see how well it would perform and how well it scales. A number of performance issues were encountered, analysed and mitigated to varying degree, in certain cases improving performance by up to 60%. Unity's built-in profiler proved to be a very valuable tool for this purpose.

At this point I would classify the system requirements of the game as follows:

Minimum (1024x768, Low, 30fps, some stutter):
- AMD Phenom X3, Intel Core 2 Duo E8400 or equivalent
- 4GB RAM
- ATi Radeon HD5570, Nvidia GT440 or equivalent DX11-capable GPU with at least 512MB of VRAM
- Windows 7 x64

Recommended (1920x1080, High, 60fps, mostly smooth):
- AMD FX4300, Intel Core i5 670 or equivalent
- 8GB RAM
- AMD Radeon HD7850, Nvidia GTX660 or equivalent DX11-capable GPU with at least 2GB of VRAM
- Windows 10 x64

Serious business (2560x1440, Very High, 60fps, very smooth):
- AMD Ryzen 5 1500X, Intel Core i5 3570 or equivalent
- 12GB RAM
- AMD Radeon R9 290, Nvidia GTX780Ti or equivalent DX11-capable GPU with at least 3GB of VRAM
- Windows 10 x64

Overkill (3840x2160, Ultra, 60fps, very smooth):
- AMD Ryzen 7 1700, Intel Core i5 7600k or equivalent
- 16GB RAM
- AMD Radeon R9 Fury X, Nvidia GTX980Ti or equivalent DX11-capable GPU with at least 4GB of VRAM
- Windows 10 x64

Dev Update - Inshal - Touches Finishing

I'll just nerf this character so it will not be overpowered... Aaaaaaaand now it's useless.

Balancing

The initial design of the game was that crawlers would tend to avoid enemies and look to gathering their strength before taking head-on fights. However, for testing purposes, we had "artificially inflated" the damage and health values of crawlers so it would be easier to test their abilities and attacks. This led to single crawlers going through the entire level, killing all enemies and the boss without so much as a scratch in some cases. It was high time to fix that!

The *S0ld13r* with his rifle and laser abilities could easily take down all enemies in his path before they could even get to him. He was rebalanced to be a long distance fighter. The overall damage of his class was reduced so enemies could still reach him and deal damage before getting annihilated.

- Low bullet damage
- Fire rate set to 5 shots/second
- Scattershot ability per bullet damage reduced
- Laser damage per second reduced

The *1nf1ltrat0r* was able to stay invisible for a duration and get backstabs on enemies for as long as his invisibility lasted. This was a problem as all enemies ignored him while he dealt quite a lot of damage to them, leading the class to easily kill even the boss. By removing invisibility, however, he quickly got overwhelmed and killed at the start of the game. A balance has been achieved in the current state. The 1nf1ltrat0r still needs to be careful about avoiding combat but whenever possible but now he is able to hold his own in a fight.

- Above average sword damage
- Backstab yields x2 base damage
- Additionally, going invisible and backstabbing deals x4 base damage
- Attacks no longer break invisibility
- Invisibility lasts 8 seconds
- Health reduced so as to encourage stealthier play

The *Sum0* was always a simple character balance-wise. He'd deal slightly more damage than everyone else and take more damage as well.

- Increased basic attack damage
- Large, distance-based Sumo Blast ability damage
- Twice as much health as average (as getting close to deal damage leads him to get hurt quite often)

Finally, the *B3rs3rk3r* had been a bit underwhelming despite his ability to switch weapons. A bit of playtesting revealed that his ability was like a double-edged sword (pun intended). He dealt more damage in his enraged state but also took quite a lot, leading to the ability as a whole to be risky. To counteract that, during the rage state, his armour was increased so he takes less damage from enemy attacks.

- Average shotgun damage
- Significant sword damage in rage state
- Increased armour in rage state
- Rage adrenaline requirement set proportional to circa. 5 kills.
- Health set to 1.2 times that of other classes (due to shotgun usually leading to close quarter encounters)

Loading Screen

A loading screen was added to the game with an animation so as to hide the game stuttering for a few seconds when loading in the level. This helped transition to the game easier and was later "prettified" by Paul so as to have a fade in!

Ability Notifier

A common annoyance was that when a crawler used an ability, it was always a guess as to when that ability would be available again. We had timers in working in the background to make the ability available again after X seconds but it was never ready when one would think it was (and really needed it). Well, now we can proudly say: "Despair no longer beloved playtesters! We, your humble developers, have added a UI element with pretty icons to display when an ability is on cooldown and when it is ready to rumble. So now you can go into that fight knowing your invisibility is there to help you run away! 😁"

Bug Fixes

- Fixed the Sumo simply obliterating an enemy with his basic attack because it registered multiple times
- Fixed Berserker rage issue where the katana would get stuck to his side if the ability ended during a swing

Dev Update - Artem - NullReferenceException

Collectibles

While our main goal were to playtest the game, fix bugs and tweak balance, we still managed to add one essential feature that makes our game complete: **collectibles**. From now on, not only crawlers depend on master, but also master depends on crawlers in a way, since crawlers now need to collect items in order to restore master's strengths.

There are two types of collectibles: **green** and **orange**, where the first one corresponds to master's ability to heal characters, and the second one stands for his fireballs.

Final placement of enemies, players and collectibles

After the collectibles are added to the game, and the demo level was significantly expanded by Paul, we had to prepare it for the final series of playtesting. My work was to arrange a proper placement for all in-game stuff to our level: player spawn points, collectibles and enemies. Finally, our level meets the conditions we outlined in the very beginning of our game practicum:

It has lower number of weaker enemies closer to the edges of map, especially where the players spawn. This allows the crawlers to slowly get used to the game before they might be accidentally killed.

It has stronger enemies in the center of the map, which increases the importance of cooperation on a later stage of the playthrough.

It has 'bad' dead ends full of enemies and 'good' dead ends with collectibles. Therefore, the crawlers have to obey their master's guidance if they want to avoid troubles and play their round smoothly!

BOSS in the middle of our level. Make him angry and enjoy running from him across the entire map!

New enemy detection shader

After a lot more details were added to the level, making the final picture contrast and complex, visibility of certain things became significantly lower. For instance, it was almost impossible to use **Enemy Detection** ability of our glorious infiltrator, since the highlights were hardly visible. For the sake of a better visibility, a new shader is introduced! It also looks more visually appealing and... highights the boss with a menacing red color.

Performance optimization and profiling

Another thing that had to be done to establish smooth gameplay even on middle-range laptops is a **performance optimization**. After a deep dive into Unity profiler and examining every called function, we detected several performance issues and fixed them. The results much cooler than we could've expected and merely overwhelming: overall FPS increased more than by 50%! What's more, we managed to get rid of performance spikes that tend to occasionally slow down our game to few FPS.

Thanks to Paul, who tested the game after the performance tweaks, we now can be sure that everything runs smoothly on a wide specter of machines.

Additionally, we tweaked network performance by significantly reducing the synchronization burden for syncing animations of players and enemies.

Bugs, bugs, bugs...

As everyone I our team fixed numerous amount of bugs and polished a lot of small features, I also want to expand this list:

- Fix bug when berserker cannot shoot after his RRRAGE fades.
- Fix bug that player spawn positions were corrupted (so that they could've even appeared in neighboring room).
- Provide missing icons for some of the active and passive abilities.

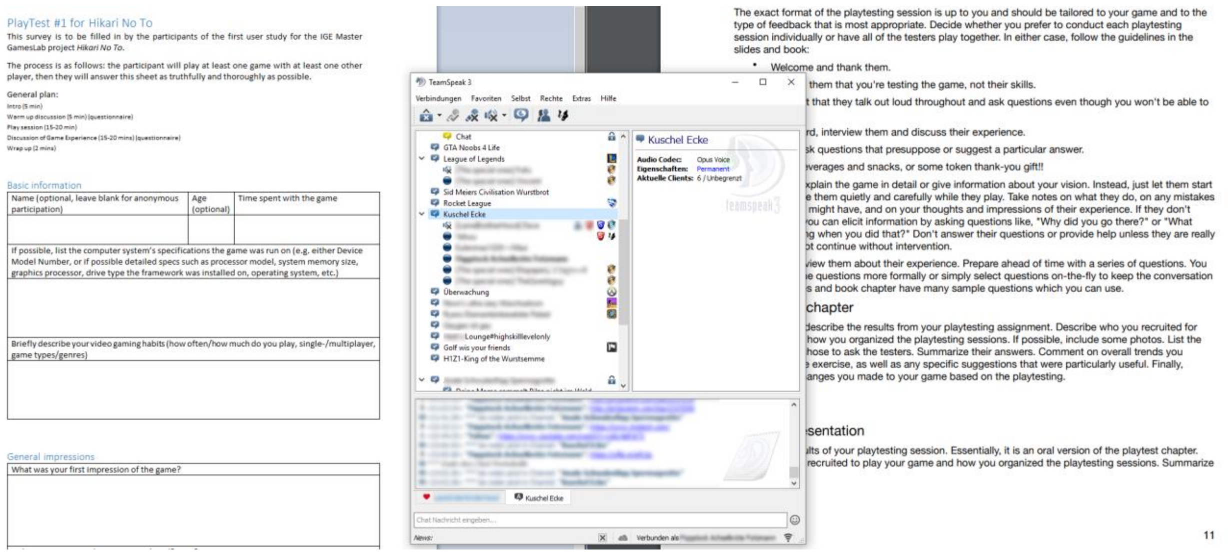
Experimental: manually sync movements

After researching what is still the most unpleasant part of our game in terms of visuals, we came to conclusion that it is a laggy movement of enemies on client. Unity does not interpolate synchronized positions and orientations of objects and it is clearly visible that update happens only 10 times per second! To solve this, we tried replacing the vanilla Unity NetworkTransform by our own sync framework. As a result... BOOM! Thee game breaks in the beginning of our playtesting! Invited testers waiting 10 minutes until we run an older version is a certainly not the best experience for us, and despite the fact that at the moment the framework is fixed and gives pleasant results, we are still thoroughly testing it and the feature is excluded from the current release.

5.2 Playtesting! - Jan 22, 2018

And yet again, our game concept turns out to be a huge pain in the a$$. Having a 5-player global multiplayer that includes one VR player, just grabbing one of your friends and placing him in front of your laptop for a couple of minutes doesn't quite do the trick, so we had to get creative.

We decided to split up the playtesting, starting with the crawlers, as they have the largest part in the game.


PlayTest #1 for Hikari No To

This survey is to be filled in by the participants of the first user study for the IGE Master GamesLab project *Hikari No To*.

The process is as follows: the participant will play at least one game with at least one other player, then they will answer this sheet as truthfully and thoroughly as possible.

General plan:
Intro (5 min)
Warm up discussion (5 min) (questionnaire)
Play session (15-20 min)
Discussion of Game Experience (15-20 mins) (questionnaire)
Wrap up (2 mins)

Basic information

Name (optional, leave blank for anonymous participation)	Age (optional)	Time spent with the game
If possible, list the computer system's specifications the game was run on (e.g. either Device Model Number, or if possible detailed specs such as processor model, system memory size, graphics processor, drive type the framework was installed on, operating system, etc.)		
Briefly describe your video gaming habits (how often/how much do you play, single-/multiplayer, game types/genres)		

General impressions

What was your first impression of the game?

The exact format of the playtesting session is up to you and should be tailored to your game and to the type of feedback that is most appropriate. Decide whether you prefer to conduct each playtesting session individually or have all of the testers play together. In either case, follow the guidelines in the slides and book:

- Welcome and thank them.
- them that you're testing the game, not their skills.
- t that they talk out loud throughout and ask questions even though you won't be able to
- rd, interview them and discuss their experience.
- sk questions that presuppose or suggest a particular answer.
- everages and snacks, or some token thank-you gift!!
- xplain the game in detail or give information about your vision. Instead, just let them start e them quietly and carefully while they play. Take notes on what they do, on any mistakes might have, and on your thoughts and impressions of their experience. If they don't ou can elicit information by asking questions like, "Why did you go there?" or "What ng when you did that?" Don't answer their questions or provide help unless they are really ot continue without intervention.
- view them about their experience. Prepare ahead of time with a series of questions. You e questions more formally or simply select questions on-the-fly to keep the conversation s and book chapter have many sample questions which you can use.

chapter

describe the results from your playtesting assignment. Describe who you recruited for how you organized the playtesting sessions. If possible, include some photos. List the hose to ask the testers. Summarize their answers. Comment on overall trends you exercise, as well as any specific suggestions that were particularly useful. Finally, anges you made to your game based on the playtesting.

sentation

lts of your playtesting session. Essentially, it is an oral version of the playtest chapter. recruited to play your game and how you organized the playtesting sessions. Summarize

11

Early Small-Scale Crawler Testing

To get a first impression of what's fundamentally wrong with the game in general, Paul and Jonas recruited a gamer friend to test the game with them in a late-Wednesday-night TS session. The participant was sent our latest build version of the game and only told to start the game, have a look around and get familiar with the controls before joining our server. During the hour-long playtesting session, Paul and Jonas would take turns hosting servers and be playing as the VR master, giving us a solid 3 player setup. The participant was asked to think out loud and describe things he liked and disliked during the game and finally to fill out the following questionnaire.

The playtesting session not only revealed quite a few previously unknown bugs but also gave us an unbiased opinion on design choices and implementations. In the following, you will find a list of the worst critique points:

- The music is horrible, sounds, in general, are too loud, making things worse, there is no volume slider in our settings menu
- The particle effects and the light on the crawler are too bright, quite blinding actually, in addition to the excessive amount of bloom
- The camera is too far away, camera controls feel both raw and sluggish, the crosshairs are off-centre
- The different crawler classes are poorly balanced, due to the nature of the gameplay some classes don't make sense
- On a dual-monitor setup, the mouse pointer would leave the screen leading to loss of focus when clicking
- Quite a few game-breaking bugs that drastically reduce fun (infinite buff, friendly fire, random deaths)

In the following day, quite a few of the bugs and short-comings were addressed and resolved.

Large-Scale Crawler Stress Testing

"Hey, let's just broadcast the download link for the game in a populated Teamspeak server, what could go wrong?"

Ok, to be fair, it was a bit more thought through though. After having addressed the most obvious flaws of our game, we wanted to get some more feedback from other people, test how the game would perform in the unforgiving environment of real-world gamers and test the game's performance under previously untested player counts. We ended up having two new testers, that went through the official playtesting process, got a warmup, minimal instructions and were asked to fill out the questionnaire.

The other ~5 participants had a less formal, but yet not less helpful, process, they literally just downloaded the game because they found it in the chat, did neither know that they were supposed to playtest it nor that Paul and I were involved in the development. As a consequence, we expected the most unfiltered form of direct feedback, and yes, they tore the game into pieces.

Again, we told our official participants to join our servers after having a look around, while the rest of the Teamspeak server just basically did whatever they wanted. As a consequence a bunch of servers were hosted simultaneously, our server was also fully populated for the first time with distinct, real people, putting the entire networking architecture to a good test, revealing new bugs and flaws.

From playing more ourselves, testers' comments during the hour-long playtesting and filled questionnaires, we could identify quite a few more bugs and poor design choices (incomplete list):

- The lobby and the main menu are bugged, Ability to duplicate oneself in the lobby, Mouse pointer disappears after unplanned disconnect
- Weird disconnects, Win/lose conditions not triggering
- Gameplay boring when dead, No death effect on crawlers, No revivability, Master or player spectator desired
- Crawlers have no indication of Master's position, No ability to communicate with master, Not aware of master's abilities, Sometimes master too high => Indication arrow and button to focus on master desirable
- Class 2 (Infiltrator) useless, Friendly fire is still in for some crawler attacks, General bugs for crawlers (Berserk can't shoot anymore after going rogue)
- Class 1 (Soldier) overpowered
- Buff kind of equal to Debuff, Crawler names not readable for master
- Update rate of master visuals too low
- Annoying SFX (too much honking, "neon tube sounds"), extreme bypass effect, poor mastering
- Camera still janky
- UI scaling issues on 4K monitors
- Game objective unclear, Running around looking for the last enemies unsatisfying, More linear map desired
- Visual Indicators of low hp desirable

Some of these issues were addressed, some are still unchanged.

Master Testing

This part turned out to be the trickiest. VR setups are still quite rare, generally quite cumbersome to set up and due to its performance-hungry nature anything but portable. In addition, useful testing of the master requires four other crawler players to take part in the session.

We decided to conduct the playtesting session in the GamesLab because it provides one working Oculus Rift based Steam VR Setup during our weekly team meeting. We had two participants (one gamer, one non-gamer) come in and test the master player while the team was playing as crawlers. The first thing to become pretty obvious was how uncommon VR still is, most people have probably never had a VR experience and therefore are unfamiliar with basically everything, including the controllers, their buttons and standard control schemes. As a consequence, both our participant had no idea what they were doing or how the controls worked (they also ended up getting stuck in the Steam VR and Oculus home screens), despite the existence of a control screen in the main menu and the VR controls being displayed in the game scene as a billboard. In addition, despite the "Please take off the HMD" screen in the main menu, it was confusing to players when to use VR or the standard Monitor/Mouse/Keyboard setup.

To tackle this, we will need to think about implementing a full VR main menu and an extensive master tutorial including tool tips in the later game. In addition, it might be useful to reintroduce the visual representations of the controllers to give new players the ability to inspect different controller and button layouts. Whether or not all of these changes can be realistically implemented before the final release is questionable, since some of them will require substantial changes to the project structure.

After the players had figured out the controls (~10 mins), they actually started to play around with the game and interact with crawlers. Generally the testers were quite happy with the gameplay, nonetheless, some bugs and suggestions for improvement made were reported:

- Resource levels of abilities not always clear, size-based approximation not too intuitive
- Movement with the Oculus restricted (Occlusion)
- Teleportation unaware of play area size, Teleport Text not visible over fire hand, Teleport text unintuitive
- Some kind of screen with condensed crawler statuses desirable
- "Please take off the Headset" screen creepy