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.

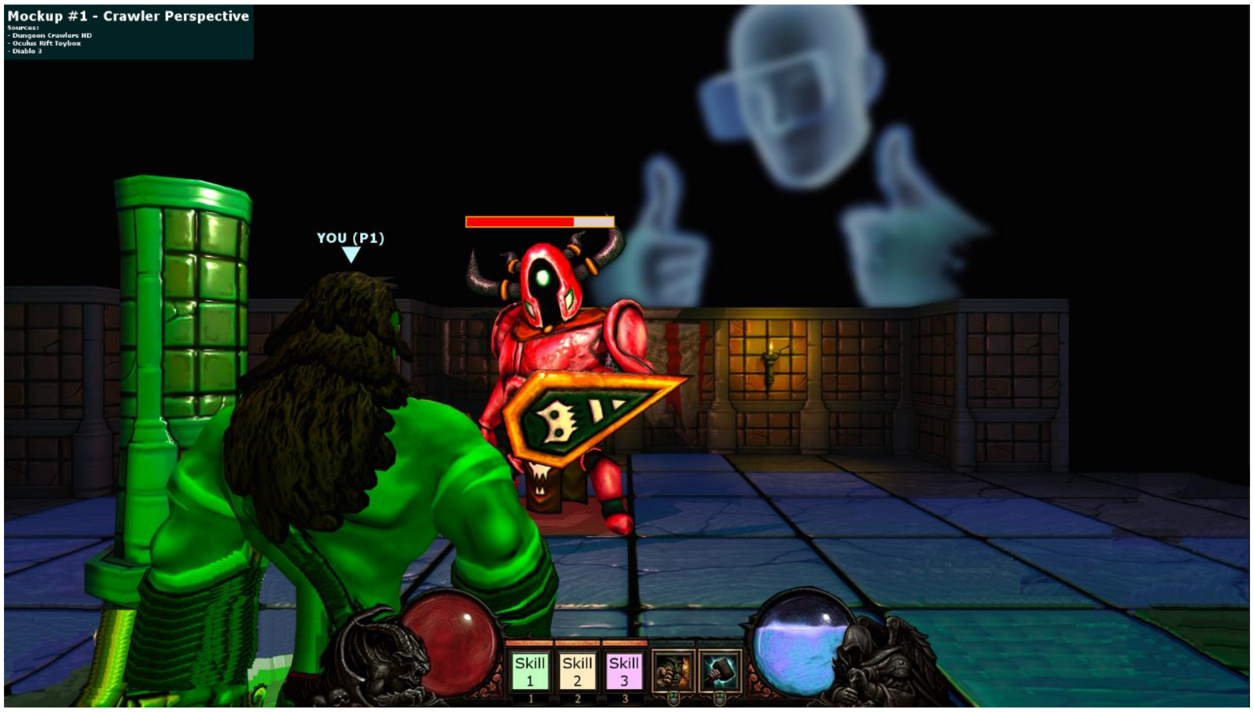
This document will discuss the basic structure and mechanics of the game in question. As such, it is divided into the following sections: *Crawler and Master Gameplay*, *Overall Mechanics*, *Style and Setting*.

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

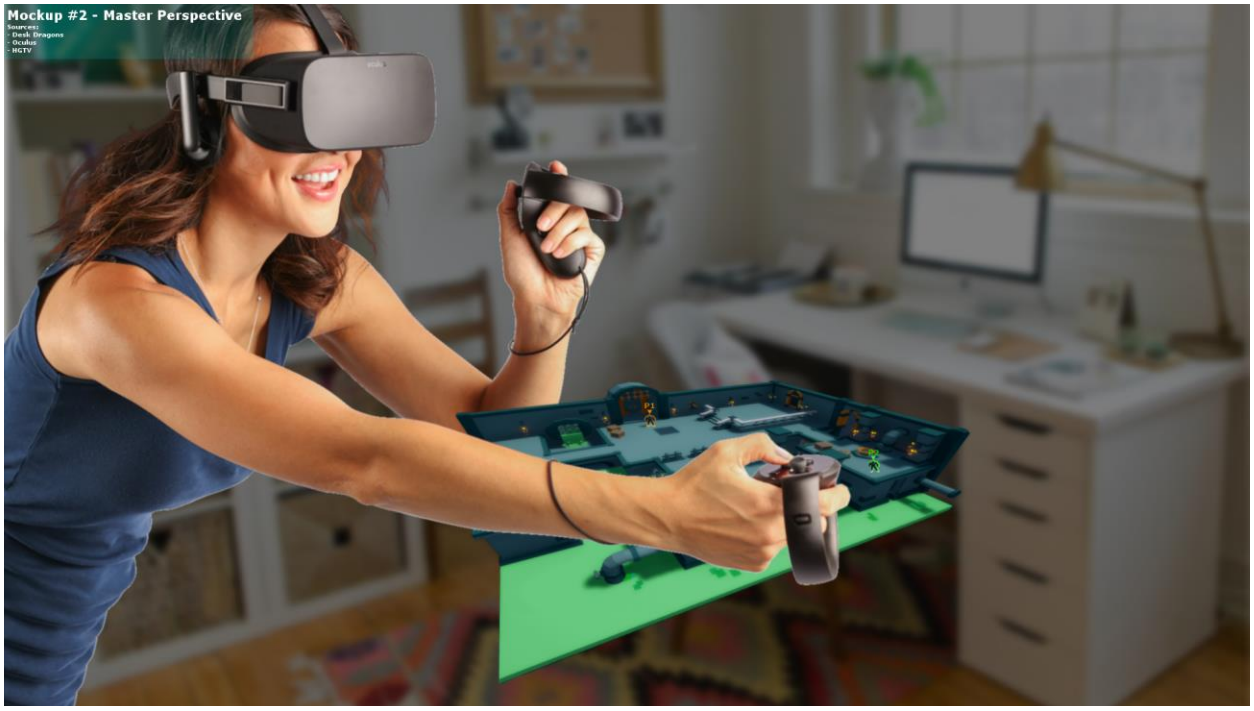
The master plays the game using a VR headset and hand controllers. The dungeon is mapped to the VR space of the system, so the master can overlook the entire dungeon and focus on areas of interest intuitively. Floating above the dungeon, the master cannot take damage. He can see the crawlers running 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.

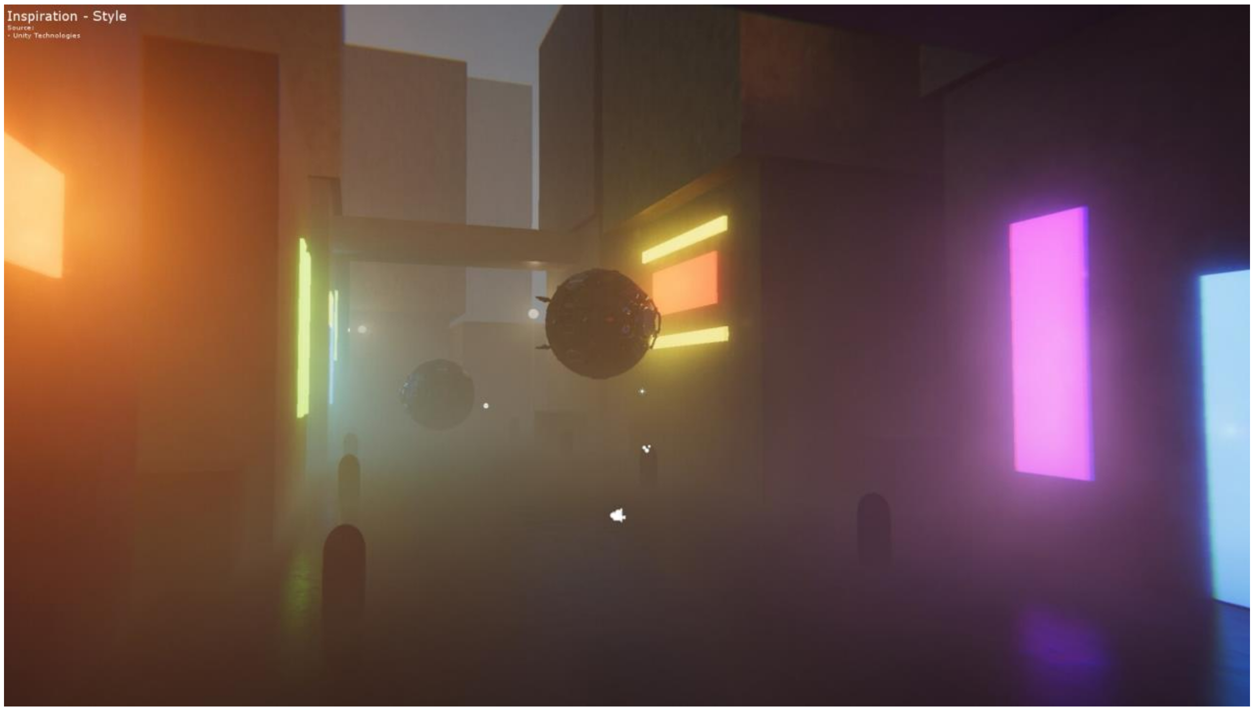
Once the crawlers are together, all players must work **together** to accomplish a given goal such as defeating a boss, solving puzzles, finding treasure, eliminating all enemies in the dungeon, etc. They receive rewards upon completion of the goal after which they may continue to the next level of the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

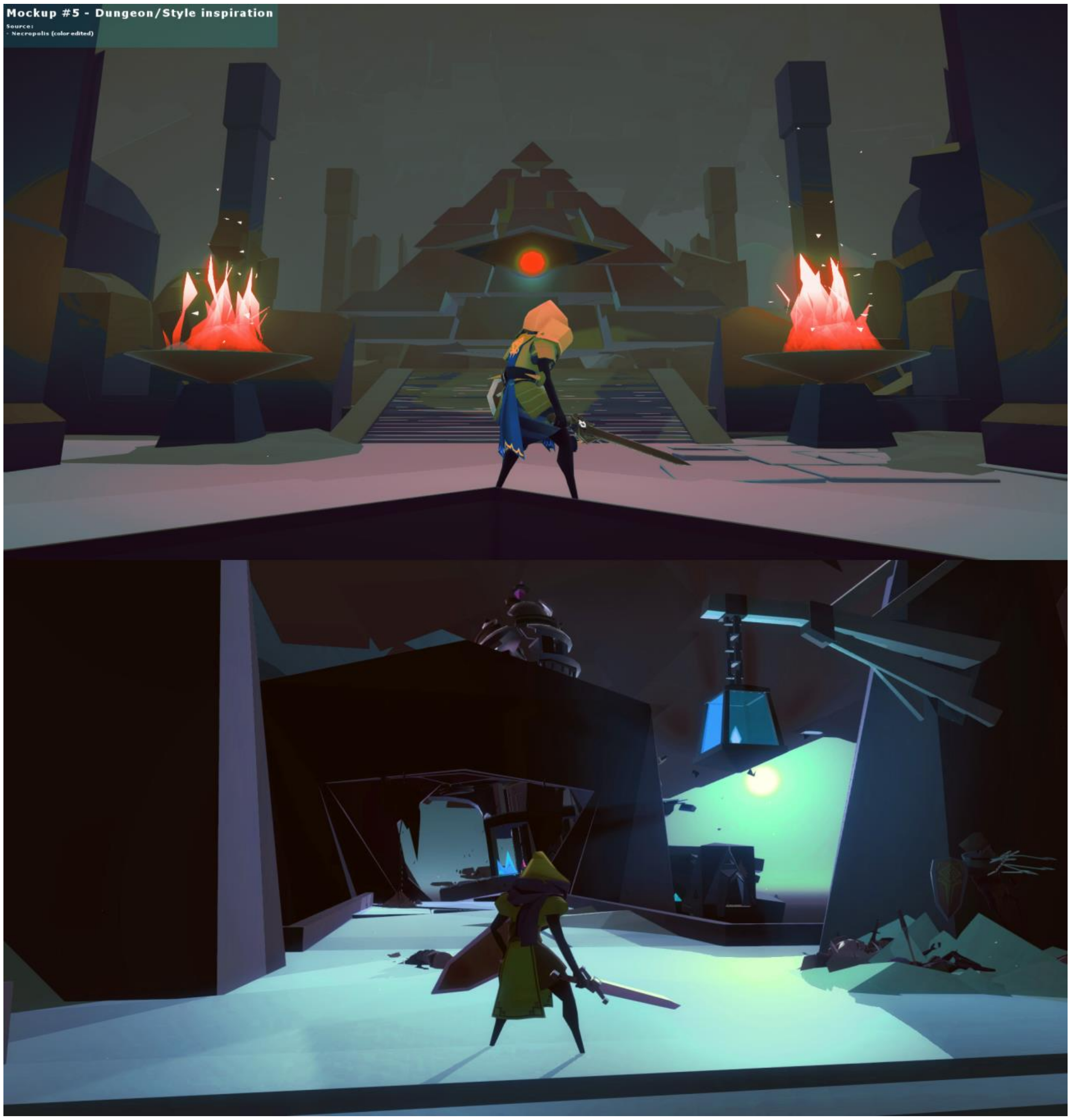
The graphical style will be a minimalistic 3D style (à la Superhot). This decision was made due to limited artistic resources and a focus on technical aspects and gameplay. To keep it visually interesting, the visuals will be dominated by strong contrasts between shady, foggy darkness and colourful, 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:

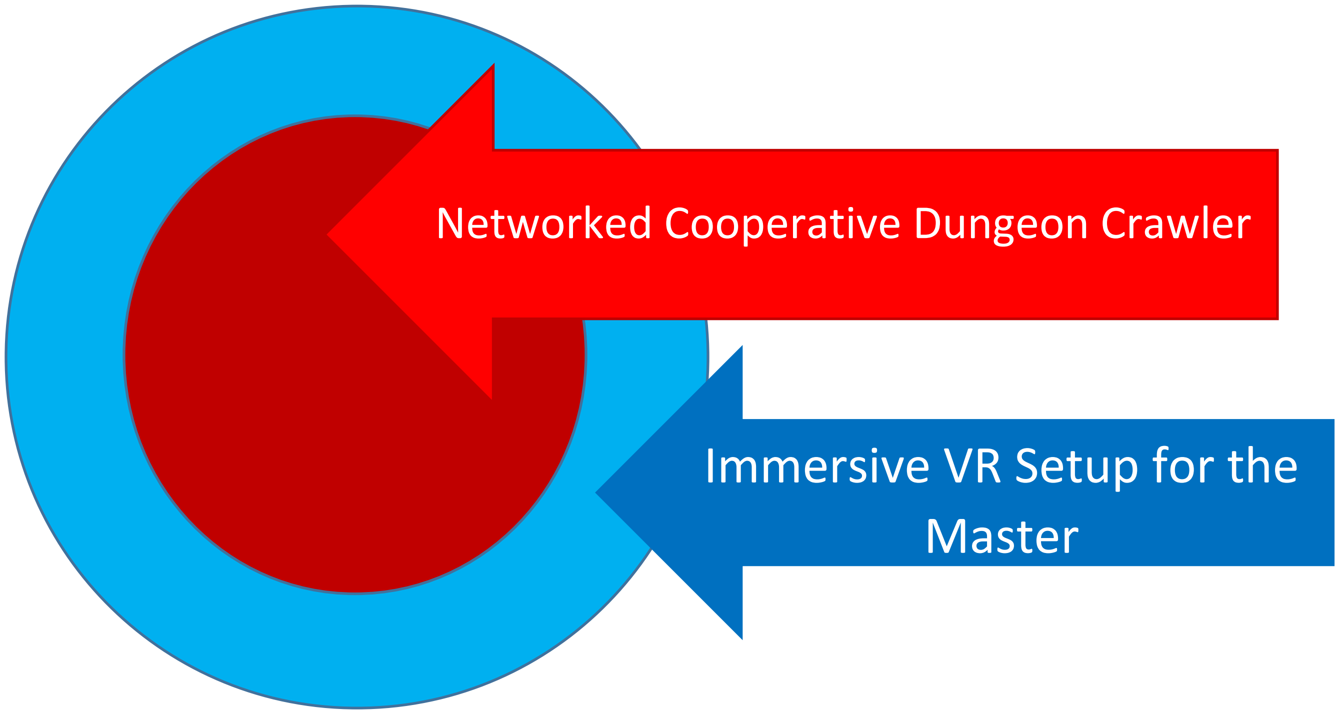
1. Set up stable and efficient networking between all players
2. Set up and tweak virtual reality with hand tracking
3. Procedural dungeon generation

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.

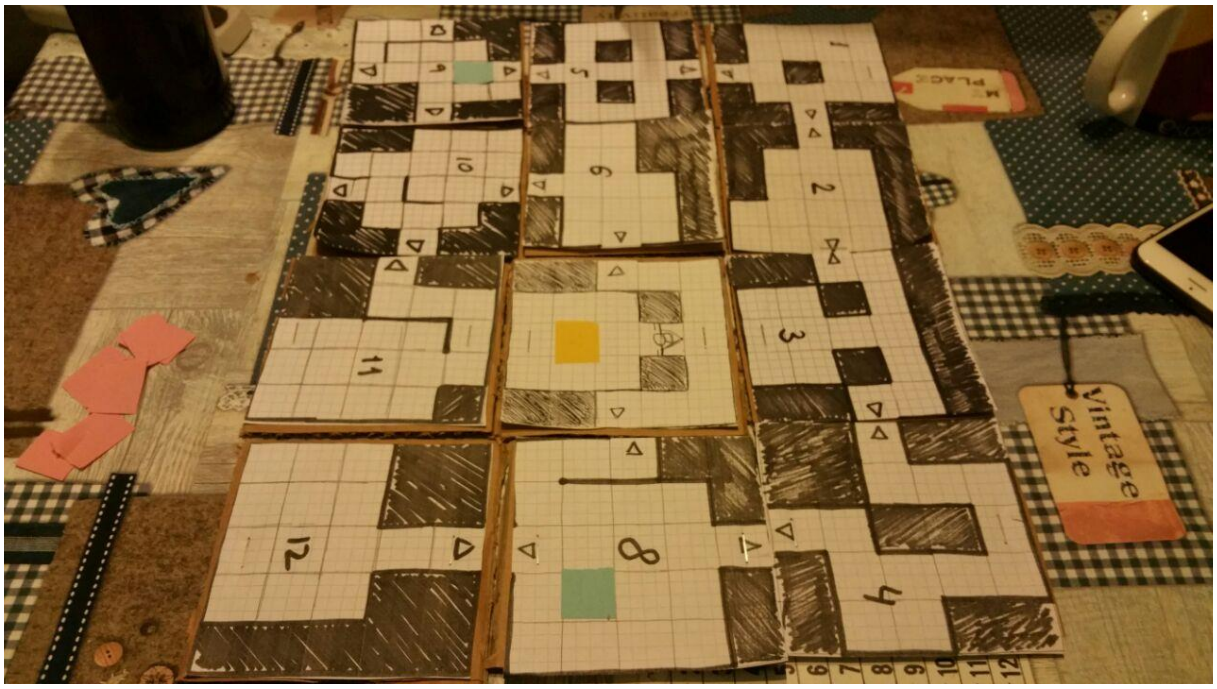
2. PAPER PROTOTYPE

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.



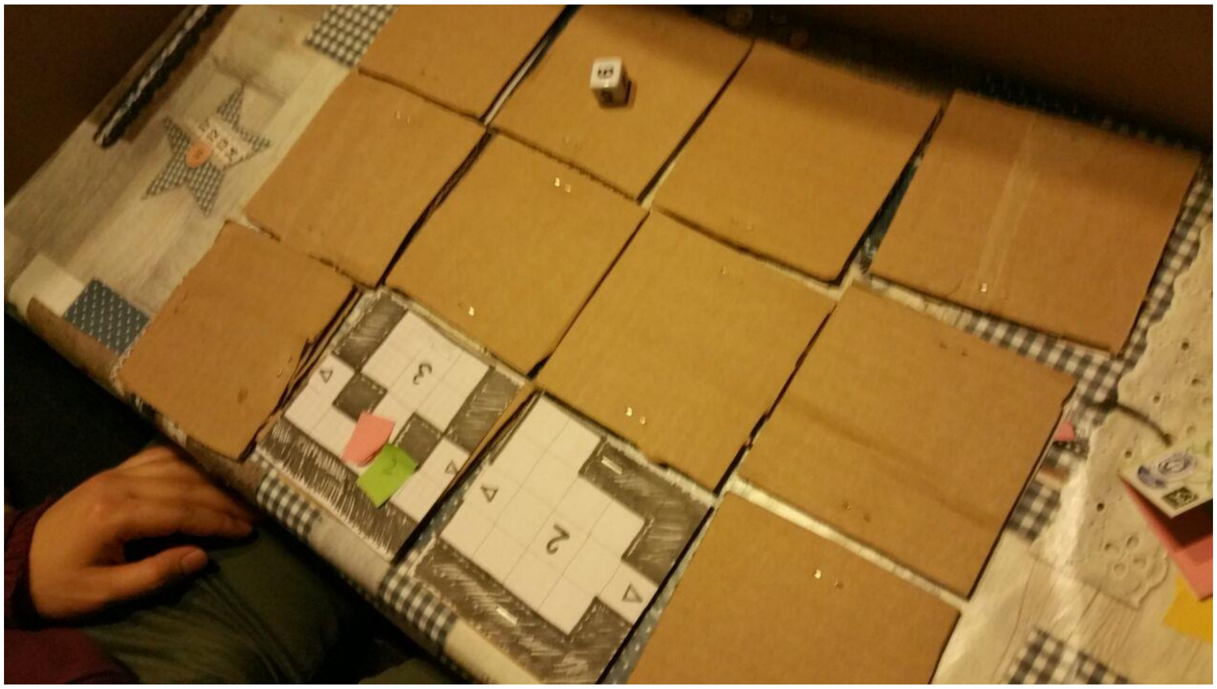
Each of the tiles within a module can be either a wall (filled) or a floor. Adjacent floor tiles can also have walls between them to prevent movement from one to the other. Floor tiles at the edges of a module can contain a door connecting it to the next module.

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

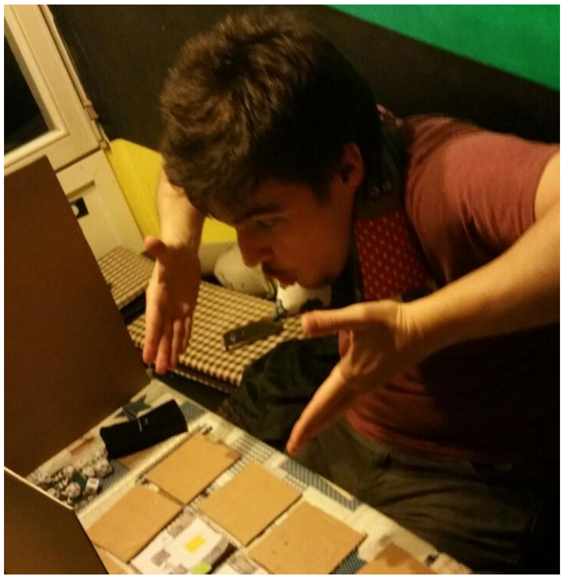
The game is lost when both crawlers are dead and won when the boss is killed, irrespective of remaining enemies.

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/Development+update+1

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/Development+update+2

It also contains videos and gifs of our project.

2.1 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

2.2 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.**