RogueGen

A PCG ADVENTURE
ALPHA REPORT

Andreas Leitner
Florian Buschek
Jean Paul Vieira Filho

GamesLab WS 17/18

Combat System

For the alpha release we had the somewhat nebulous goal of a “better” combat system. For this, several big changes and additions have been made.

BASIC COMBAT

To make swinging a weapon feel more responsive than pressing the button and then having a 1 second animation play, most weapon attacks now are charged up as long as the button is pressed down. When the button is released the attack then happens relative quickly. Skill effects such as dealing damage can also scale on the time spent charging up.

Characters with two weapons equipped can now use both of these weapons at the same time. For this a system was implemented that defines how much a skill would hinder a skill from the other hand, potentially disabling the ability to use that skill. For example, a character can't swing most weapons while blocking with a giant shield.

A more specifically defined goal was to add a lot more variety in the weapons and skills than the 3 we had in the Interim Report, to give players more choices and more different enemies to fight. For this, there are now 11 different weapons with a total of 21 different skills to use. All weapons and skills were designed to feel different from each other, for example the basic sword strike is fast and inflicts a bleeding condition that deals damage over time, while the mace is slower, deals more damage directly, ignores a portion of the target's resistances and pushes enemies away.

For these skills, new effects were implemented in addition to dealing damage: Increasing/Decreasing resistances, defenses, movement speed, and offensive capabilities, as

well as pushing and pulling enemies. New types of skills were also added, such as piercing projectiles, rays, blocking, and generally skills affecting areas. More weapons, especially some that heavily encourage team work might be added during the playtesting phase.

ENEMY AI

Enemy AI was also improved. In addition to the distance to the players, the AI can also take into account the target's resistances, defenses, and remaining health, as well as the number and average health of adjacent allies. With this, melee fighters will go into close combat range with the players, but instead of only chasing their closest target they might switch their focus if they find a more lucrative target, such as a player with low health.

Enemy AI can now also take into account a general threat level. This is calculated from players activating skills and projectiles in the close vicinity of the enemy. With this, enemies can use defensive skills such as blocking or dashing away more sensibly.

The AI also had to be changed to allow the activation of two skills at the same time and take into account the possibility of one skill hindering another. Now if both weapons have to choose a skill at the same time, the skill with the highest score of both weapons is chosen. Then, for the other hand the skill with the highest score that could still be used, is also activated.

Another change is that weapons and skills can now contribute movement patterns to the basic ones enemies have. This makes it so that now any enemy can have any combination of weapons and skills and tries to figure out some sensible course of action, for example going into close combat range when a melee weapon is equipped.

ENEMY SPAWNING

To fill the generated world with enemies, a spawner object was implemented that generates and instantiates enemies based on various parameters, such as positions, the biome it is in, minimal and maximal levels, and a general power level for the encounter. The power level is essentially a resource the spawner can spend to generate enemies, so it might result in more, weaker or less, stronger enemies. It is also an easy way to scale the difficulty of encounters with the number of players.

For this, enemies were restructured a bit from the Interim Report. Enemies are now generated at runtime out of several components.

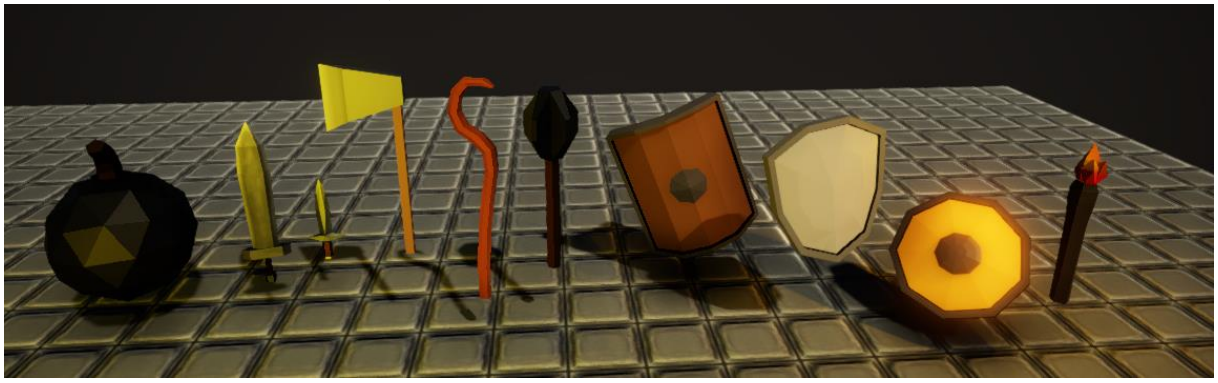
First, a base character is selected (a knight, rogue, mage or wizard). These have differing attributes, for example a knight can generally defend better against physical attacks, while a mage if good at resisting magical spells.

Then, each character gets one or two weapons randomly chosen from a weighted list of available options (unique to each character). The result is a finished enemy, as the AI automatically takes care of how the enemy should function based on the available weapons and skills.

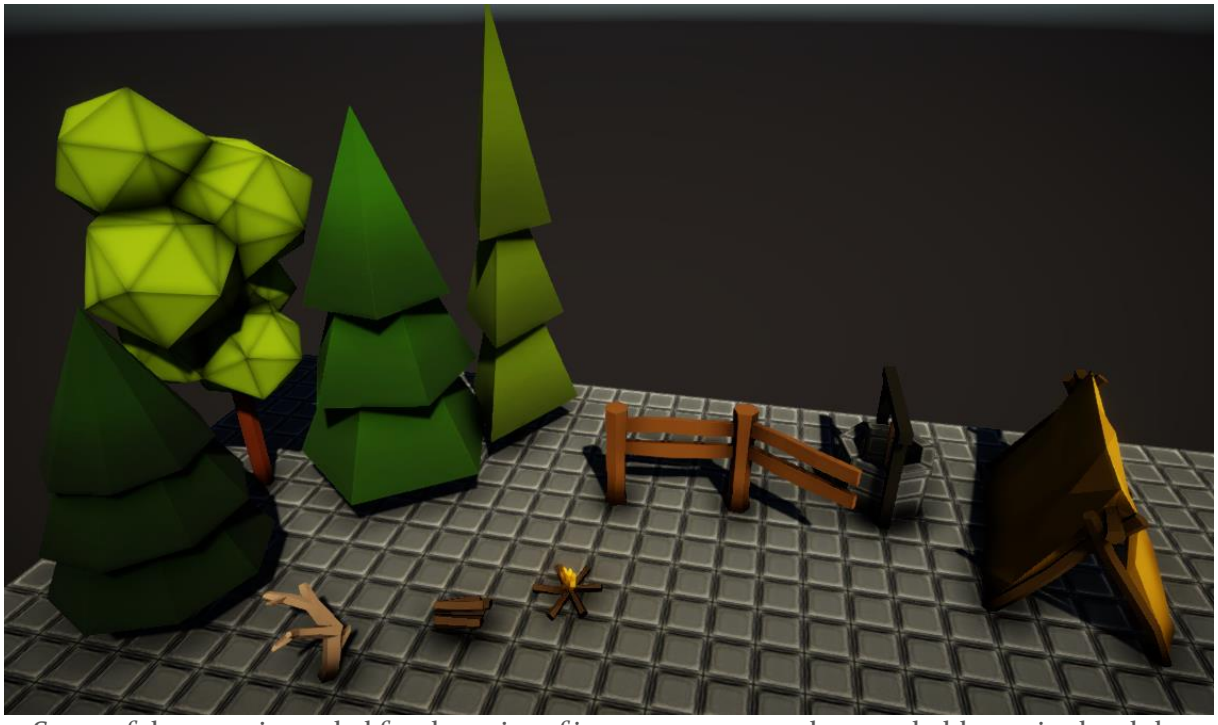
Models and Textures

For the alpha release all desired artistic goals are reached.

There are many different objects the player can pick up, mainly weapons and shields.

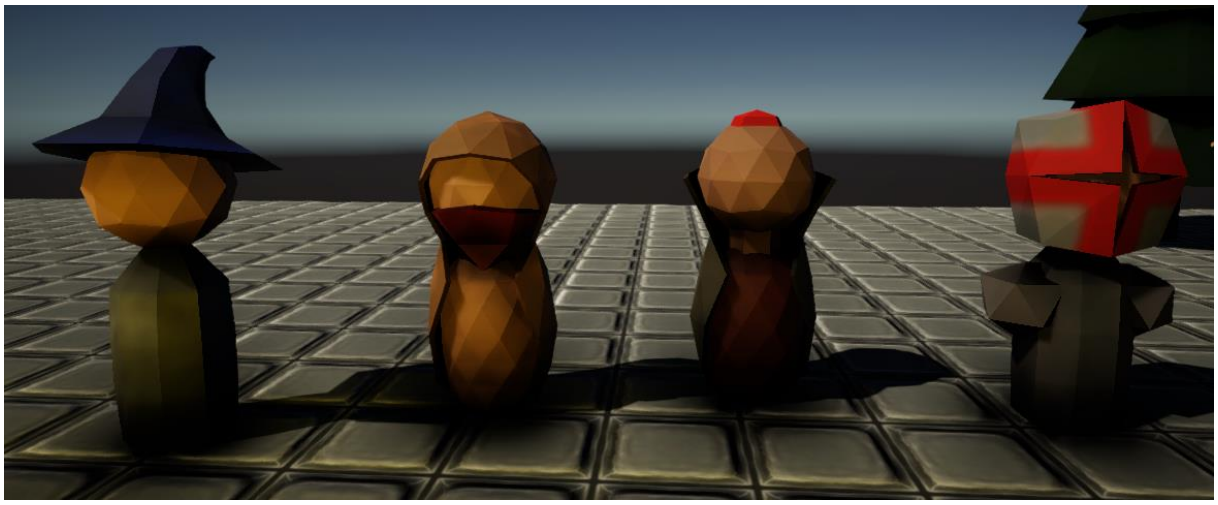


The following objects can be used to fill the environment.

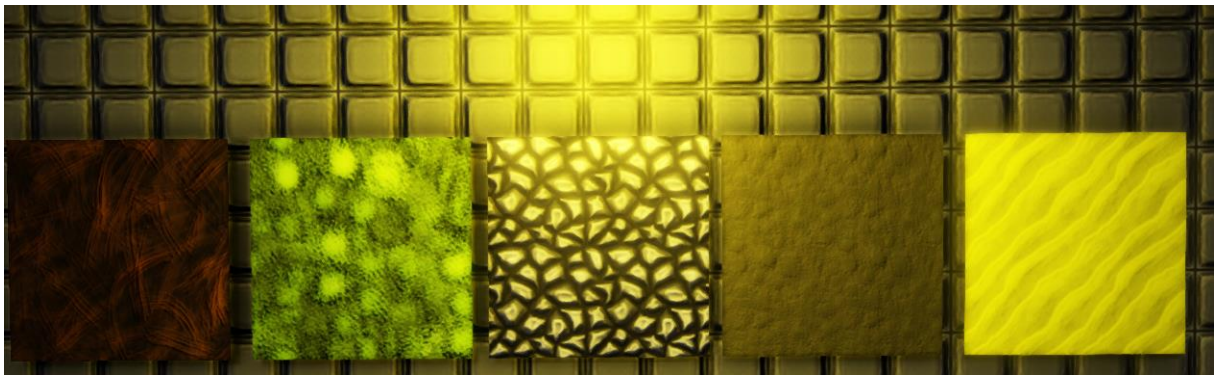


Some of them are intended for the point of interest system and are probably not in the alpha release.

There are now 4 character models. From left to right: Wizard Thief Mage Knight.

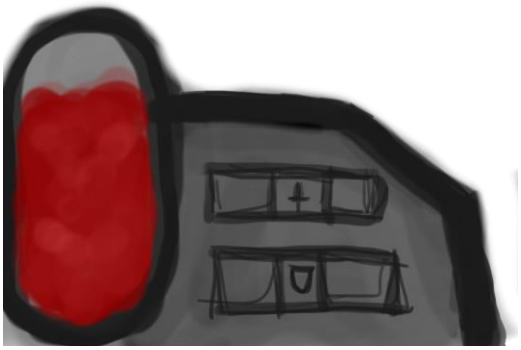


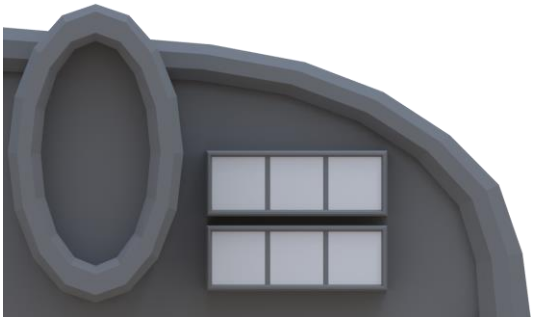
The high target goal of self made biome textures was reached. There are several tile able textures for the different biomes. In order to look better (especially with point lights / torches around) they all have normal maps.



Hud mockup (left) and final version (right). The idea is to have one of this hud elements for every player in one edge of the screen. It shows the health, and small icons for both weapons

and the abilities of those weapons (and cooldowns if they have any). This is not jet in the game and will probably be added before playtesting.





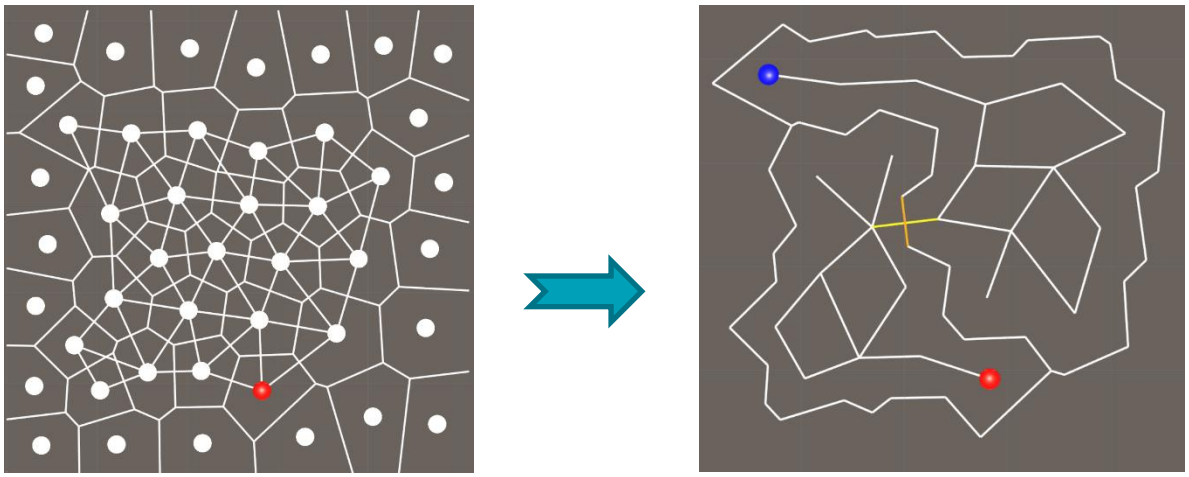
Terrain Generation:

In the terrain generation aspect of our game, we implemented features that introduced a playable state. More structures were added to generate a playable world and support the game logic as well as combat system.

MINIMUM SPANNING TREE

To divide the playable world into “Areas”, in which the players must overcome challenges to progress in the game and win, a minimum spanning tree is generated using the Delaunay triangulation from the Interim release and the Prim’s algorithm, by choosing a random “Start Node”. After the MST is generated, the farthest node from the Start Node is sought and marked as “End Node”.

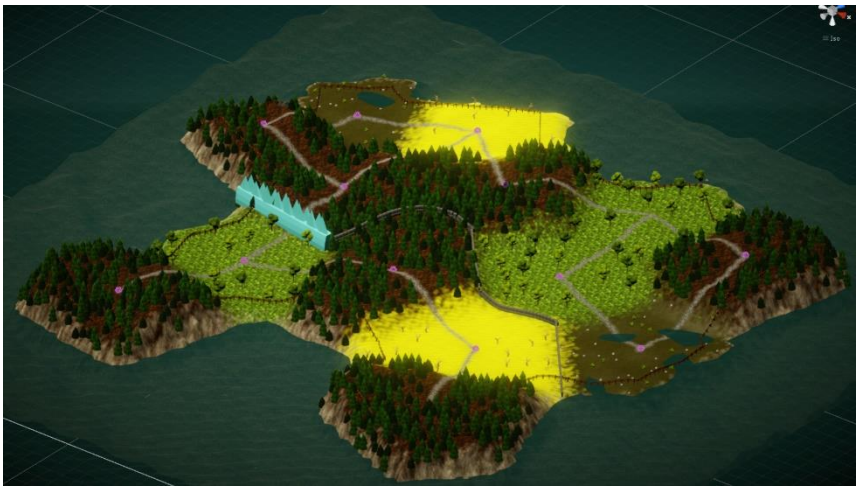
When all this information is gathered, the path is then divided in a fixed number of segments. The nodes that branch out of these segments are grouped into areas.



On the left: The Delaunay triangulation with a start node (red)
On the right: the polygon areas with their respectives path inside, the end node (blue), crossing path (yellow) and crossable border (orange)

AREA POLYGONS

After the areas are defined, we can then generate the polygons for each area. These polygons are used to instantiate prefabs at the borders between areas, and between an area and the coast. Currently, fences separate the playable area from the borders of the map and walls separate the areas from one another.



Current terrain generation state: 4 different biomes, road texturing, enemy spawner placement and confined areas

CLEAR POLYGONS

In addition to the "filling polygons" presented in the interims demo, we have also implemented "clear polygons". These represent areas that should stay clear of prefabs, e.g. roads, points of interests, etc…

With this combination, we can then decide in each area where the players can navigate and where they should not be able to cross.

ROAD LINES AND TEXTURING

Roads are now drawn into the terrain to facilitate navigation.

ENEMY SPAWNER PLACEMENT

In each biome, an enemy spawner is placed and the level of the enemies spawned increase the further the players progress in the game. These spawners are described in detail in the Combat System section.

Game Logic

QUEST SYSTEM

Players now receive instruction on what to do during the game. After each quest is completed, a new one is given. Once the final quest is completed, it is intended that the game ends and the main menu is loaded. This last feature still needs to be implemented

Player perspective with quest instructions at the top

MAIN MENU

Additionally, we implemented a start menu for the players to set specific parameters for the game, such as numbers of players, seed for the map generation, etc...

Current State and Challenges

It proved to be harder than expected to generate certain aspects of the terrain. The first one that comes to mind is the geometry involved. While it is easy to sample the points of a polygon, we still needed to reorder them to check if a prefab is inside it or not. Also, finding a library that provides all the functionalities w.r.t to geometry was hard, so we had to implement a part of the functions ourselves and the rest integrated from multiple sources (different libraries, stack overflow code snippets). Unfortunately, one of our team members left the team, so we had also one person less working on the project and therefore had to downscale our game to deliver a playable experience.

These are some screenshots from our Alpha Release