RogueGen

A PCG ADVENTURE
INTERIM REPORT

Andreas Leitner
Florian Buschek
Jean Paul Vieira Filho
Johannes Runkel

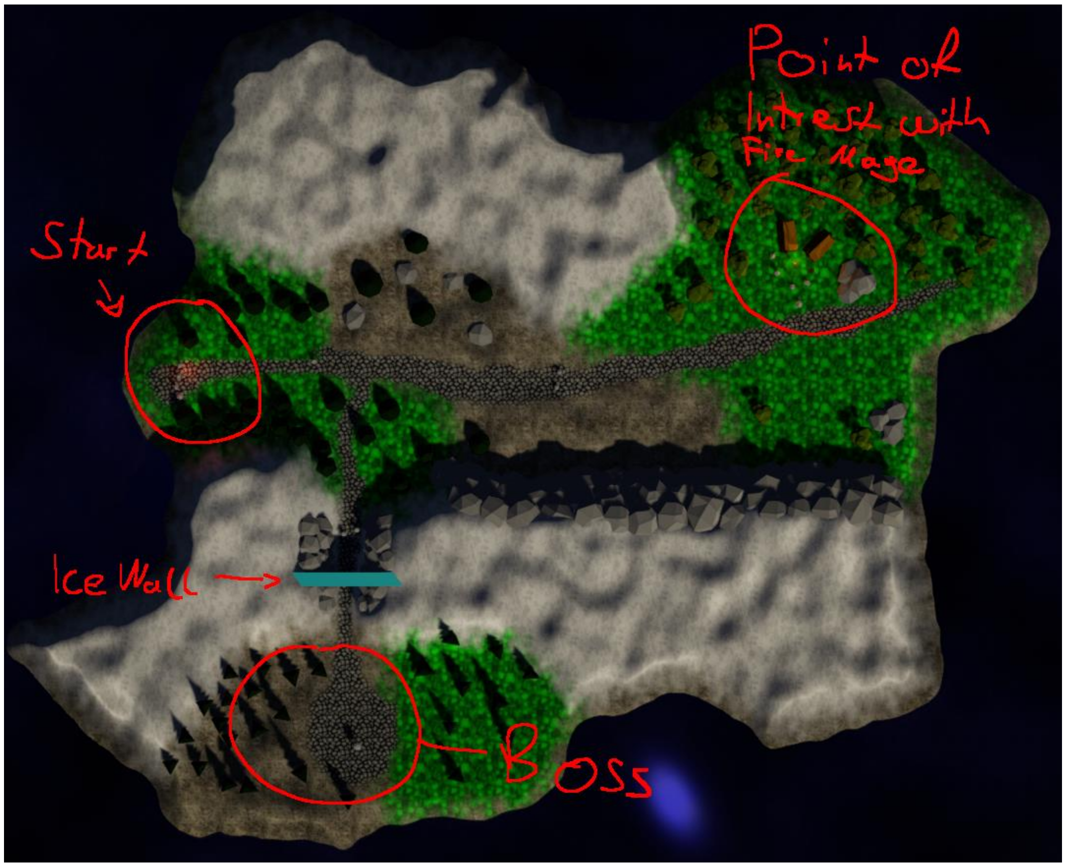
GamesLab WS 17/18

Artwork and Handmade Map

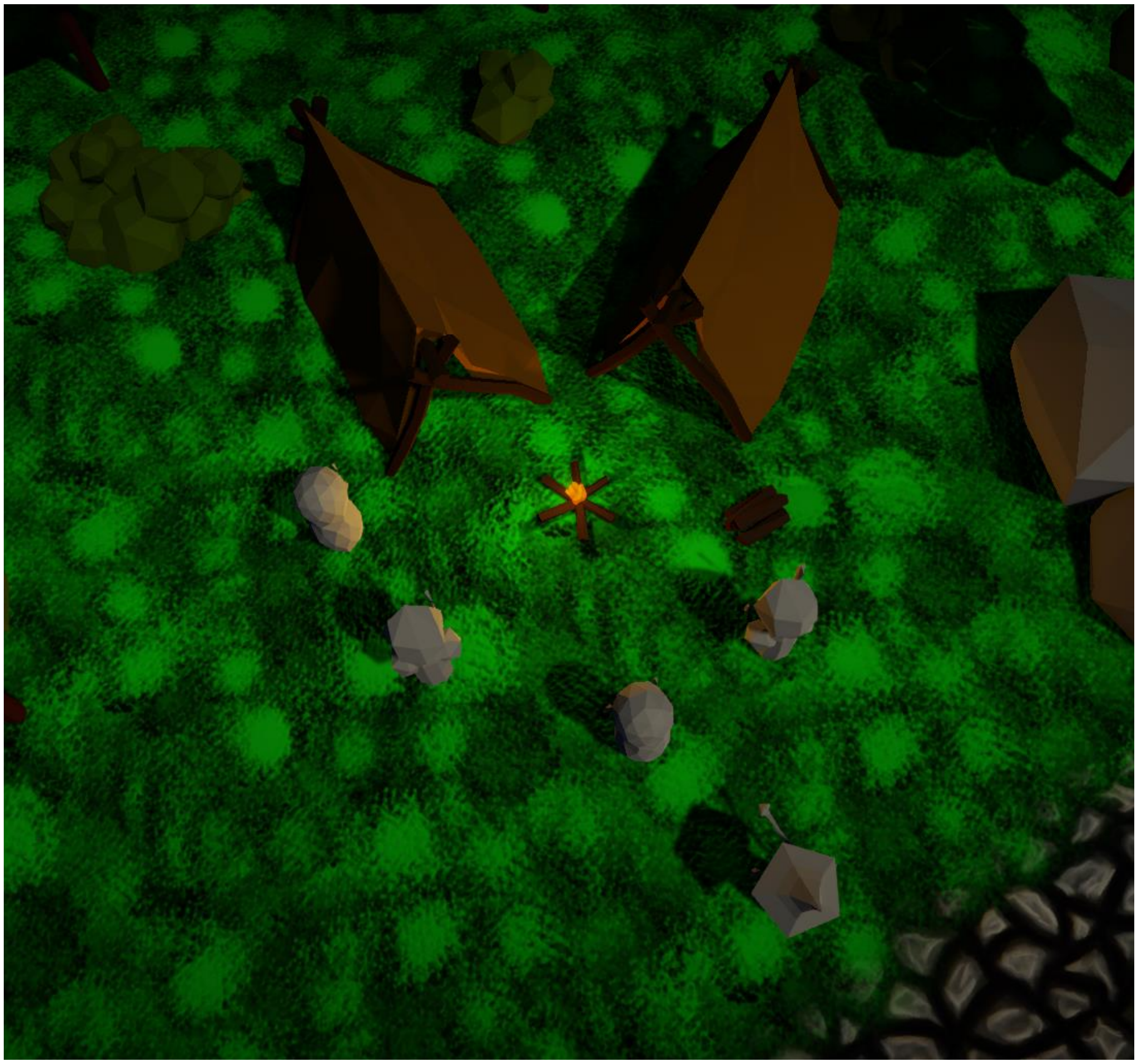
For there interims report there are currently three character models, intended for both the player and the enemies. One more playable character is planned so every of the four players has his own character.

For the environmental assets there are multiple types of trees for different biomes. Then there are rocks that can fit in many biomes, but can also be used to define cliffs.

For the handmade map multiple simple terrains with only few biomes were generated until one was found that suited the general idea for his map. For this simple and small map the idea was to have two separated areas. The bigger one is the starting area, and the smaller one is a boss area. They are separated by an ice wall. In order to destroy this wall the players have to find the fire staff.



The special point of interest in this map is a campfire with two tents and a stack of firewood. The campfire is surrounded by multiple normal enemies and the fire mage who obviously drops the fire staff.



In the early prototype stages the lighting will mostly be done with one bright directional Light (Sun), so that it is possible to see everything, even on laptops. For the final version it is intended to be darker, and mostly lit by point lights in the environment, and the lamps the players can carry around. It will probably be necessary to have a brightness slider for laptops.

Combat System

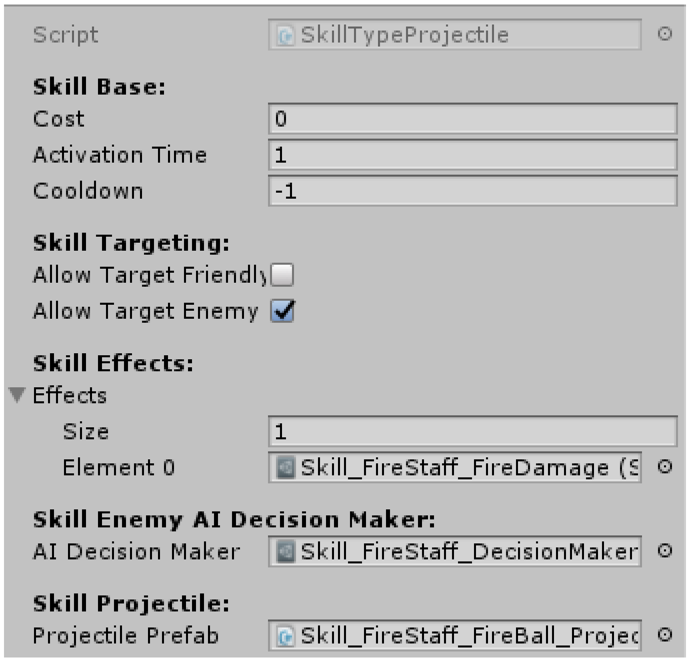
The main idea of how we wanted to implement the combat system was to make it as easy as possible to generate new content for it. Thus, we implemented mostly the systems behind characters, movement, weapons, skills, and AI. This means, that we don't have that much content in each category yet, however adding more content in the future should be easy and fast.

Characters share a common pool of statistics, such as health, resistances against different types of damage, and their equipment. Characters are moved through a physics based system, allowing nice interactions with other characters, the environment, and skills.

Each character can equip up to two one-handed weapons or one two-handed weapon. Each weapon grants the wielder up to two (or four for two-handed) skills.

Characters can activate their equipped skills. These represent some combat action the character can do in relation to the equipped weapon. Currently, there are 3 different weapons with one skill each in the game: a sword that can be swung, a throwing knife, and a staff that shoots a magical fire ball.

These skills are made up of two parts: a skill type and multiple skill effects. The type determines the general form the skill takes on, such as swinging the weapon or creating a projectile. Once the weapon or projectile hits it's target, all skill effects are applied. These receive various information about the owner, target and skill. So far, skills can apply damage, but other effects, such as changing resistances, can easily be implemented. They can also check if the target reached a certain health threshold and then decide whether to apply the damage or not. With conditional effects like that, skills can be build in a tree structure allowing complex interactions.



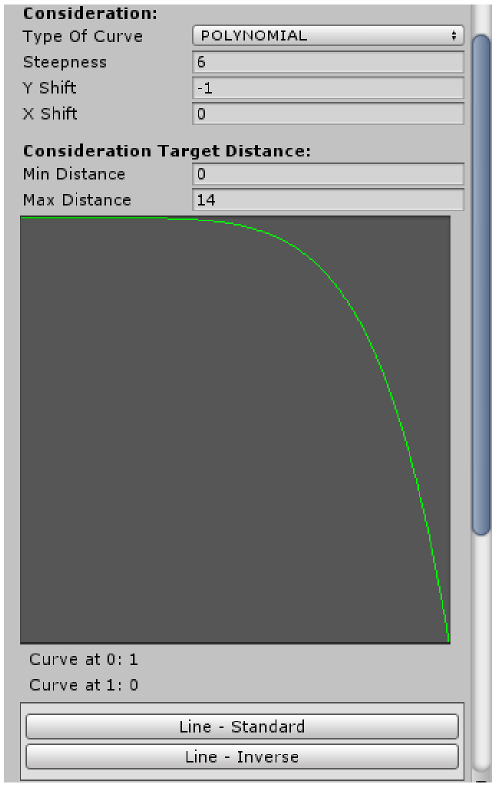
Script	SkillTypeProjectile
Skill Base:	
Cost	0
Activation Time	1
Cooldown	-1
Skill Targeting:	
Allow Target Friendly	☐
Allow Target Enemy	✓
Skill Effects:	
Effects	
Size	1
Element 0	Skill_FireStaff_FireDamage (S
Skill Enemy AI Decision Maker:	
AI Decision Maker	Skill_FireStaff_DecisionMaker
Skill Projectile:	
Projectile Prefab	Skill_FireStaff_FireBall_Projec

Skills also have various attributes, such as the time it takes to activate, the time it takes until the skill can be used again, and an animation that plays while activating. Here we had some problems at first getting the animation properly mirrored for the other hand due to some problems with the original rotation. In the end, we decided to rework the animations we had done from the ground.

There is also a functioning resource system called energy, however it is not used for the demo yet.

For the AI of enemies we decided to implement a utility based system, where each potential action is evaluated based on various considerations and then the best is chosen. The possible actions of an enemy are to activate a skill and to decide how they want to move. For this, each enemy has a list of movement patterns, functions that evaluate how the character moves, and one is always active. Currently, enemies can move to or away from the player, stand still and look at the player, or do nothing. Which movement pattern the enemy uses is decided in fixed time intervals.

To decide which skill or movement pattern to use, each skill and movement pattern is assigned a list of considerations. Each consideration takes a single value as input that is evaluated with a linear, logistic, or polynomial function. The result of that function is always between zero and one. Then, the results of all considerations are multiplied with it's base score and a compensation for the number of multiplications is added. Considerations that require a target, for example the player, are evaluated for every legitimate target and the best is chosen. Then, the skill or movement pattern with the highest score is chosen and activated.



Consideration:

Type Of Curve	POLYNOMIAL
Steepness	6
Y Shift	-1
X Shift	0

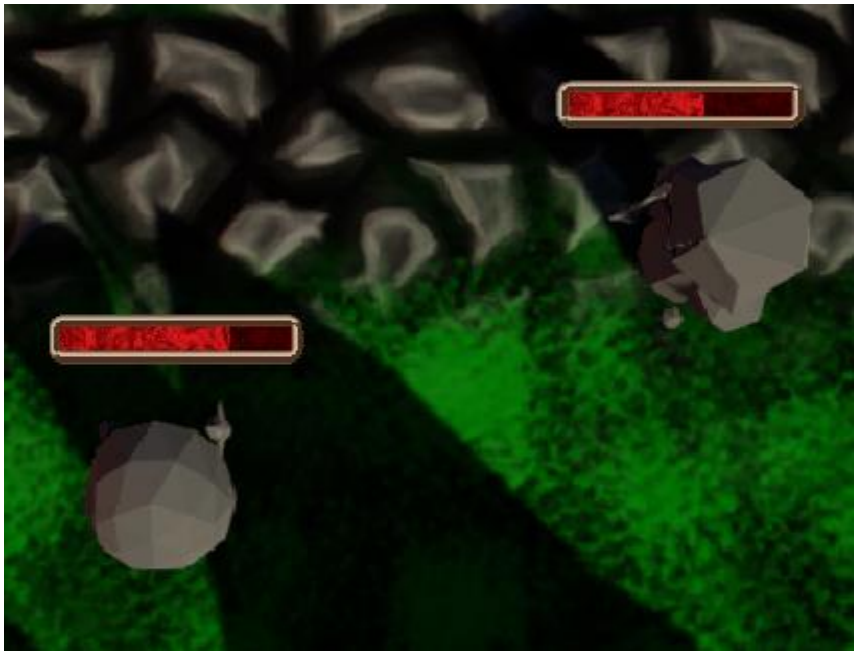
Consideration Target Distance:

Min Distance	0
Max Distance	14

Curve at 0: 1
Curve at 1: 0

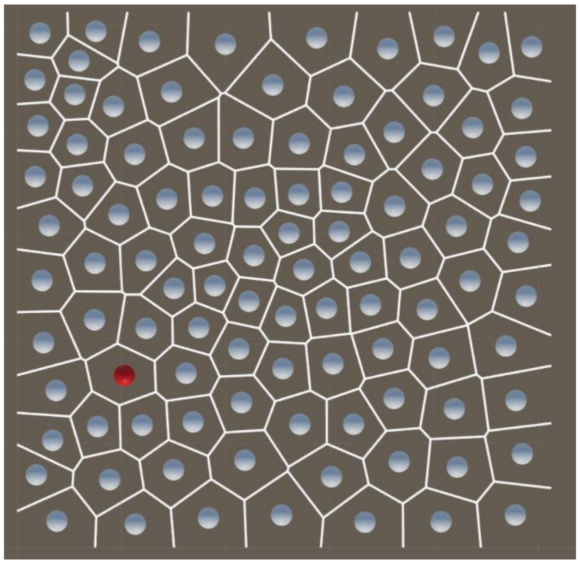
Currently, the only consideration fully implemented is the distance between the enemy and the players. This is used for a melee enemy that walks towards the player and attacks if close enough, as well as a ranged enemy that starts attacking as soon as the player is in range, and will try to escape should the player get too close.

We also made a simple GUI in the form of a health bar floating over the characters.

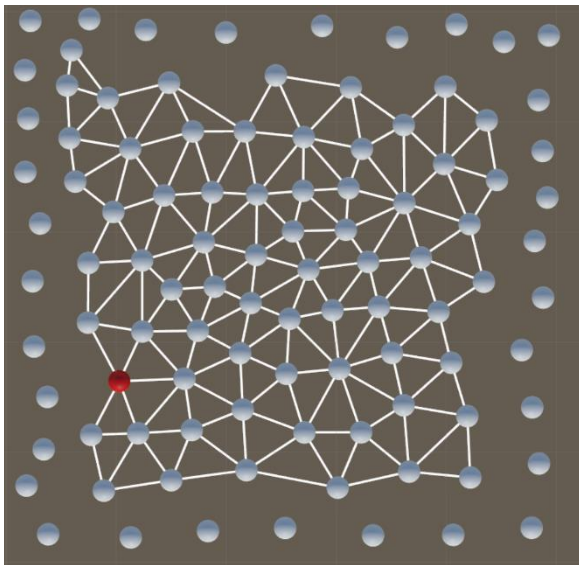


Terrain Structure

As a starting point, the biome system was modelled. We wanted to create a system where each biome would influence the landscape in our game. At first, we tried to use a Delaunay triangulation between the biomes to create an overlay mesh and use barycentric interpolation.

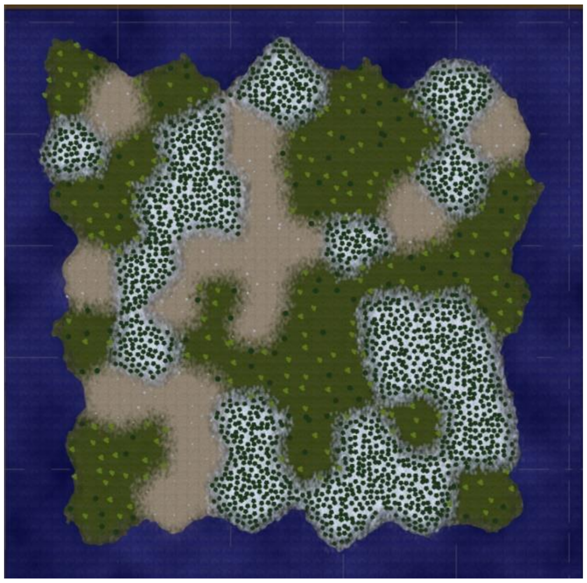


Voronoi Diagram



Delaunay Triangulation

However, the performance was not optimal, and we changed the system to use Voronoi diagrams. Each Voronoi cell influences the region it covers. The Delaunay triangulation was kept to know the neighbors of a biome.



Textured and filled heightmap

To create a map, a list of biomes is given to the map generator. Each biome contains information about Perlin noise parameters, ground texture, conditions (humidity and temperature) and neighbor blending parameters.

At first, the biomes are randomly distributed through the map and pushed away from each other with some iterations of Lloyd relaxation. The border biomes are set to water.

Afterwards, a heightmap is created and the values are sampled from the biome cells based on their position plus some uniform noise, which gives more detail in the borders between biomes. The same process is repeated for the texturing part.

To achieve a better visual quality and prevent texture stretching in great height differences, we also implemented a triplanar shader for our terrain.

Lastly, the biome polygons are extracted and filled using Poisson Disc Sampling. This last part is not permanent since it doesn't add any extra detail to our world other than "filling". But the function to fill a 2D polygon with samples is certainly going to be used later on.

The final result looks good but it is still lacking interesting elements such as grass, roads, structures, etc..

This is what our terrain generator is currently capable of: