# CATIATORS

**A humoristic arena-fighter, featuring heroic kittens.**

**Made by**

**Quokka Entertainment**


**Members**:

Imme van der Made 377903

Machteld Dalhuisen 347033

Robin Potze 369482

Cyrill Ozhoga 380619

Class: CMV2E

Minor: Game Lab #1, 2.3.1

Hanze University of Applied Sciences, Groningen

# Game Design Document

## About the game Catiators

**A humoristic arena-fighter, featuring heroic kittens.**

**By Quokka Entertainment**
**Members**

Imme van der Made 377903

Machteld Dalhuisen 347033

Robin Potze 369482

Cyrill Ozhoga 380619

Class CMV2E

Hanze University of Applied Sciences, Groningen

09-04-2019

# 1. Section I - Game Overview

## 1.1 Game Concept

A group of players is dropped in an Arena, where they will have to fight each other and obtain as many kills as possible in order to achieve the highest score. If a player reaches 5 kills, the game will end and the high score will be shown. Every death will count as a negative point and will lower the end score.

At the start of the game, each player chooses a class, out of the 2 available classes (berserker and ranger). Each class has its own set of unique skills, which consists of one normal skill and one skill with crowd control. This crowd control skill grants extra damage and extra crowd control effects.

## 1.2 Features

- Local Co-op without the need for controllers - use your phone as a controller.
- The possibility of creating a game with more than 4 players, in the easiest way, since only extra phones are needed.
- A fast-paced, stylized, 3D party game.
- Cats as main characters.

## 1.3 Style and genre

The style of this game is 3D and stylized. The genre is Local Co-op, casual, social, competitive arena fighter.

## 1.4 Target Audience

The target audience of this game consists of young adults of both sexes (age 16-24), who do not have a lot of money. The main focus is on people who speak English and who are casual gamers or non-gamers, but who are nonetheless interested in playing a social and (slightly) competitive game.

## 1.5 Game Flow Summary

The players will start out in the outer corners of the level and have the choice to directly go towards other players, via the upper or bottom sides, via the left or right sides with the traps, or the choke point in the middle. Each of these paths have their own advantages and disadvantages. From there they will be able to use any attacks to kill the other players as soon as possible.

## 1.6 Look and Feel

As said before, the style of this game is 3D and stylized. The style of the level is made to be mysterious, dark, magical and a tad ominous, since the players are in a player versus player mode and are meant to fight. This is emphasized by the rendering and particle systems in the game, which direct the player's attention or show the player that a certain area is dangerous.

More details on the mood of the game can be found under Aesthetics.

### 1.6.1 Intended user experience

The intended experience, is for the player to experience a fast-paced game, with unexpected traps and other ways the environment as well as the other players can damage the player. All of this in a mysterious environment, that consists of old ruins with small magical aspects, with cats as the heroes of the game. The cats in the game are used in order to make the game feel more casual, with a "cute" and stylized style and very cat-like accessories (with paws and bells on it), to make the arena feel more different and interesting and opposite from the characters of the game, which makes both the arena and the characters stand out more.

# 2. Section II - Gameplay and  Mechanics

## 2.1 Gameplay

### 2.1.1 Objectives

A group of players is dropped in an Arena, where they will have to fight each other and obtain as many kills as possible in order to achieve the highest score. If a player reaches 5 kills, the game will end and the high score will be shown. Every death will count as a negative point and will lower the end score.

### 2.1.2 Play Flow

The game is a simple and fast-based combat arena game. All players will have the same goal of killing the other players five times. Each player will have three attacks that they can use to reach this goal. These attacks are the basic attack, the crowd control attack and the ultimate attack. The game will give clear signs of feedback when a player hits another player with their attacks. The game will be designed in such a way that the players will be able to focus on each other, and the attacks they can use to kill each other within the game.

## 2.2 Mechanics

### 2.2.1 Basics

#### 2.2.1.1 Scene NeverUnload

The game opens from the scene NeverUnload. In this scene there are two GameObjects, which are the SceneManager and the AudioManager.

The GameObject AudioManager has the AudioManager script that manages all the sounds within the game. All scripts can call the function Play(string name) from this script, which then plays the sound with the same name as the string that is given.

The GameObject SceneManager has the script ManagingScenes, and PlayerLoader as its child GameObject. The ManagingScenes script loads and unloads all the other scenes in LoadSceneMode.Additive mode. The NeverUnload scene is never unloaded, because this scene will contain all the GameObjects that need to be send from one scene to the next.

The GameObject PlayerLoader has the script Server. This script opens the server to make sure that the phone application is able to connect to the game. This script also receives and handles all the messages between server and application. It also has the static function LocalIpAddress(), which can be called by any script in the whole project to find the IpAddress of the device that runs it.

#### 2.2.1.2 Scene RobinMenu

When the game is opened the ManagingScenes script of the SceneManager GameObject loads the RobinMenu Scene. This Scene contains five GameObjects, which are the EventSystem to handle the UI of the menu, the Screens that contains the 8 different Canvas of the menu, Scenes that contains all the models that make up the menu and end scenes, the CameraHolder that contains the CameraControl with the Camera and the transforms that indicate the camera positions, and a DirectionalLight which is only culled to the Player layer.

The GameObject Screens contains 8 different Canvas GameObjects, and these are called the ExitScreen, StartScreen, PlayerScreen, CharSelectOverlay, LoadScreen, HeroScreen, OptionScreen and the EndScreen. From these, only the StartScreen and the EndScreen are enabled.

When the RobinMenu scene opens, the EndScreen script on the EndScreen Canvas GameObject checks if there is a Score GameObject in the NeverUnload screen. If there is a Score GameObject found, the script moves the Camera with the use of the CameraControl in the CameraHolder to the end screen position and shows the scores that can be found in the GameObject Score on the end screen. If there is no GameObject Score, the camera will stay at main menu position and the EndScreen is disabled.

The main menu is in the StartScreen Canvas. In this Canvas there are five GameObject Buttons, which navigate to their own linked Canvas within the GameObject Screens. The PlayButton GameObject opens the PlayerMenu and the CharSelectOverlay Canvas. Together these two Screens handle the player menu.

In the player menu you can choose which players, with what controller and with what characters the game will be played. This menu Instantiates Controller GameObjects for each player in the PlayerLoader within the NeverUnload scene, and these Controller GameObjects will store what controller and character the players will use. The Confirm GameObject Button in CharSelectOverlay will first check if all players have chosen their characters before calling the SceneManaging script attached to the SceneManager in the NeverUnload scene to load the Arena2_Elective Scene and unload the RobinMenu scene.

#### 2.2.1.3 Arena2_Elective

The Arena2_Elective scene contains four GameObjects, which are the CameraControl, the Field, the Arena, the prefab LevelManager and the SpawnPoints.

The GameObject Arena contains all the models, lights and other GameObjects that make the level look the way it does.

The GameObject CameraControl has the script CameraControl and the child GameObject Camera in it. The CameraControl script uses the Field GameObject to find all the locations of the player locations. Then it moves the CameraControl to a position where the Camera can see all the different characters.

The SpawnPoints GameObject contains all the spawn points that the game uses to instantiate GameObjects.

The LevelManager prefab has the child GameObject Directional_Light_For_Players which is only culled to the player characters, Characters which is where the characters of the players will be instantiated, LevelUI which is where the UI for the different players will be instantiated, EventSystem for all the UI elements, Message which shows the level mission or objectives at the beginning of the level, and the PickupManager which regularly instantiates pickup items.

The GameObject LevelManger has the LevelManger script. When whatever scene is started that contains the LevelManager prefab, this script will look for the PlayerLoader from the NeverUnload scene. If passed from the menu, the PlayerLoader contains the GameObjects with the controller scripts for each of the players that will play the game. These scripts can be the StandardController script for the pc or the standard controllers or the ClientServerConnection for the phone applications. For each of these controllers, the LevelManager script will call their StartGame() function to start the game for each of these controllers.

When the StartGame() of the StandardController or ClientServerConnection scripts is called, the controller will instantiate the character that has been chosen in the menu at the spawn point that the LevelManager script has assigned them. The character will be instantiated in the

Characters GameObject in the LevelManager GameObject. When the characters are instantiated the controller will store the BasicMovement of that character.

### 2.2.2 Physics

The models in the Arena GameObject in the level scene have colliders where needed to set the boundaries of where players are able to go. The characters have a CharacterController component, and all movements will be done through this component to make sure that the characters will not be able to go through objects.

### 2.2.3 Movement

#### 2.2.3.1 General Movement

Whenever a button of the pc, controller or application is pressed, the controller scripts will catch them and use the functions of the stored BasicMovement script to make the character take the chosen action.

The BasicMovement script will make the characters move around the level through the command MovingCharacter(Vector3 direction, bool jump). The Vector3 direction will decide in what direction the character will move and the bool jump will decide if the character will move normally or dash into this direction.

#### 2.2.3.2 Other Movement

The controllers will also be able to call the HoldBasic(bool hold), UseCCAttack(), and UseUltimateAttack() for the different attacks of the characters.

The HoldBasic(bool hold) will keep track if the basic attack can be used. When the function is called with a true hold the basic movement will call the basic attack of the character every few seconds until the HoldBasic(bool hold) is used with a false hold. The basic attack is called from the function into the Weapon script that is attacked to the character. The Weapon script is the parent script to the LongSword or the BowArrow scripts so that the same function can be called, while the actions of the characters will be different.

The UseCCAttack() and UseUltimateAttack() will first check if the attack can be used. If the attack can be used they will call the scripts of the ccattack or ultimateattack to call the attacks. The timer of the attack will be set to 0 and the attack will not be able to be used until a certain time has been passed.

### 2.2.4 Objects

#### 2.2.4.1 Picking Up Objects

##### 2.2.4.1.1 PICKUPS MANGER

A game object with PickupManager script is present in game scene in addition to several pickup spawn points. Over set period of time random pickup item from Pickups pool is being instantiated on random spawn point. Pickup doesn't appear on certain spawn point if a player is too close to it and if another pickup is still present on this spawn point.

##### 2.2.4.1.2 HEALTH PICKUP

One of pickup items that can appear on arena is health pickup. This pickup is represented through a red heart model that emits light. When player character touches this pickup it disappears and set amount of health is being added to character's health amount. Player's health can't go over maximum health, so picking up health when player has full hp will make pickup disappear, but won't add extra health.

#### 2.2.4.1.3 TORNADO PICKUP

Another existing pickup that appears on arena is tornado pickup. It is represented through a tornado model that emits light. When player touches this pickup, tornado disappears and all players in set radius around it are pushed from it apart from player that picked it up. This pickup has no visible advantages or disadvantages , but it can be used in different in-game situations to push players to a trap or get of a pursuit from faster player.

### 2.2.4.2 Moving Objects

#### 2.2.4.2.1 GRINDER

There are a few grinder GameObjects in the level that are moved around with the Grinder script. It will constantly move around the angle that is given in the inspector. The grinders also have the OnTrigger script that has an OnTriggerEnter. Whenever a player hits the grinder the script will call DealDamage() in the PlayerManager script of that player to deal them some damage.

#### 2.2.4.2.1 SPRINKLER TRAP

Two sprinkler traps are present on arena. Active area of this traps constantly rotates around trap's axis. When player gets into the active area he/she is getting damage and being constantly pushed from the trap up until player leaves active area. More than one player can get in active area of the trap with similar trap effect applied to them. Trap can be activated and deactivated , it's rotation speed and influence area are adjustable, but for now this kind of traps is constantly active with set 360 degrees rotation and set rotation speed.

## 2.2.5 Input of the players

The player can have three types of controls, which are handled through two different scripts. Both scripts use a stored BasicMovement to make the character of the player that uses that controller to move around.

One of these scripts is the StandardController script. This script catches the Input from the mapped Input System from Unity itself and then uses these Input for the functions of BasicMovement script. When this StandardController script is instantiated, the script will also call the SetupButtons(bool pc, int controllerIndex) function to make sure that only the input of that player will be caught.

The other script that sends the input to the BasicMovement script is the ClientServerScript. The buttons on the application will be mapped with the PhoneControl script. This script sends the input to the server with the functions in the Client script. The server then uses the ClientServerScript of that player to translate the message and move the character with the store BasicMovement script.

## 2.2.6 Combat

For each attack there will be a collider that hits the other players. On the same GameObject there will be an OnTrigger script that deals the dmg to the other player. When OnTriggerEnter is called on that script, the script will look for the PlayerManager of the GameObject that is hit. If this PlayerManager is not the same as the player that has used the attack, the function DealDamage() of that PlayerManager script is called to deal the damage. This function updates the health bar, starts the getdamage animation, and checks if this player has died.

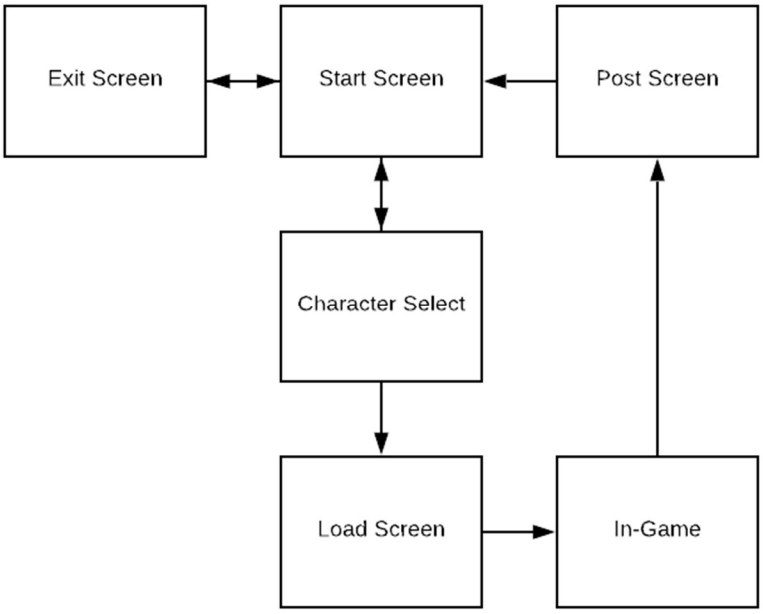
If the player dies, the DealDamage() function will also call the DeathAction() function. This function starts the coroutine BeingDeath(). This coroutine starts the death animation, shows the ghost, starts the ghost animation, hides the character for a few moments, then after a few seconds hides the ghost, moves the character to the start spawn point, reveals the character,

sets the health back to the maximum health, and makes the character invincible for a few seconds.

The function DealDamage() returns a bool to the OnTrigger script, which tells the OnTrigger if the player that has been hit has died. If the player has died the OnTrigger will tell the PlayerManager of the player that has used the attack, that they have killed another player. This will update the PlayerUI of the player that attacked. If the player has killed other characters five times the PlayerManager will call for the LevelManager script to end the level. The LevelManager will instantiate the Scores prefab, in which all the scores of the players are stored. The parent GameObject of Scores will be set to the **SceneManager** in *NeverUnload*. The LevelManager script will then call the SceneManager in the NeverUnload scene to unload the level and load the menu.

## 2.3. Screen Flow

### 2.3.1. Screen Flow Chart




### 2.3.2. Screen Descriptions

#### 2.3.2.1. Main Menu Screen

The main menu screen is the first screen that is shown when the game starts. Players can choose to either start the game, and go to the character selection screen, or quit the game entirely.

#### 2.3.2.2. Character Selection Screen

The character selection screen allows players to enter the game in an assigned colour, according to the order of entry. The players will then be able to select their control mode, as well as the character they want to play. Once all participating players are locked in, the game can be started by the confirm button. If the players decide they want a different character, control type, or want to leave the game altogether, they can press the back button in their selection section. If the players want to return to the start menu, a back button is located on the bottom of the screen, next to the button that starts the game.

#### 2.3.2.3. In-Game Screen

The in-game screen shows all the gameplay, as well as the statistics of the players. The players' health, class, cooldowns, kills, and deaths are displayed here.

#### 2.3.2.5. Post-Game Screen

The post-game screen shows who has won the battle, and who did not. The players' scores are displayed next to their class icons, but of course the winner is now the evilest kitty on top of his altar. This screen leads directly back to the start screen.

## 2.4. Game Options

The game currently does not have an options screen, but the players are offered multiple options for controlling the game, and characters to play.

The controls offered do not change the mechanics of the game, but the gameplay experience does differ per type. The keyboard is slightly more uncomfortable than a controller is, but it does allow full control of the game. The traditional controllers are the most comfortable and allow for multiple buttons per action so the player has a choice of preferred input. The mobile controls are custom to our game, but lack physical feedback and are reliant on a good internet connection.

There are two playable characters in the game, which differ in one being the berserker, a short-ranged, fast moving character, which focuses on getting close to his enemies, and knocking them around. The other character is the ranger, which is a character that should stay at a distance, can trap his enemies in a box, and shoot them with a bow.

## 2.5. Replaying and Saving

From the end screen the player will be send back to the main menu. From here the player can restart the game by pressing the Play Game button again. However, the player will have to redo all the controller connections as nothing is saved at the end of the game.

## 2.6. Cheats and Easter Eggs

### 2.6.1 Kill oneself

Just before a player will die from an other players attack, they are able to walk into a trap to kill themselves. In this manner the other player will not register that they have killed someone, while the first player will get a full health bar. There is no punishment for doing this, so the player will be able to avoid ending the game.

# 3. Section III – Story, Setting and Character

## 3.1. Story and Narrative

### 3.1.1. Back story

While cats are usually quiet, elegant creatures, they have always thirsted for more power. In order to obtain this power, a team of hero cats arose and has taken on the challenge of defeating each other in a battle arena. Every fight is a step closer to cat world domination.

## 3.2. Game World

### 3.2.1. General look and feel of world

As said before, the style of this game is 3D and stylized. The style of the level is made to be mysterious, dark, magical and a tad ominous, since the players are in a player versus player mode and are meant to fight. This is emphasized by the rendering and particle systems in the game, which direct the player's attention or show the player that a certain area is dangerous. More details on the mood and the look of the game can be found under Art.

### 3.2.2. Area #1

Area 1, which is currently the only level, consists of a ruin/castle-like courtyard, with a gloomy feel. This area is where the players come out of the gates of the sides and will fight each other.

## 3.3. Characters

### 3.3.1. Ranger

**Personality**

The ranger is a cat who is fast, with high agility, crafty and one with nature.

**Look**

The main colours of the Ranger are green and brown. He wears leather armor, a hat and a quiver for his arrows. He uses a bow as a weapon.

**Special abilities**

The ranger attacks with a bow and arrows and has a special crowd-control attack where he throws a box toward other players. The box will capture the other player and do damage and apply crowd control.

### 3.3.1. Berserker

**Personality**

The berserker is a cat who is a little slower compared to others (due to the armor), and is someone who first dives into a fight and then thinks.

**Look**

The main colours of the Berserker are grey and black. He wears place armor with a helmet and a uses a sword for his attacks.

**Special abilities**

The berserker's special attack is an attack where he charges another player and hits them with the armor on his shoulder to push them away and stun them briefly.

# 4. Section IV – Levels

## 4.1. Level #1

As said before, in *Game World*, area 1, which is currently the only level, consists of a ruin/castle-like courtyard, with a gloomy feel. This area is where the players come out of the gates of the sides and will fight each other.

More information on this level's look can be found under *Art.*

### 4.1.2. Introductory Material

The introductory material for the level is brief, only consisting of the loading screen (with a small preview of the level) and a short instruction on killing other players.

### 4.1.3. Objectives

The objective for the level is to kill the other players. This can be done with the player's skills directly, but the level can also be used to do this. The water and various traps in the level damage the players.

### 4.1.4. Structure and design of the map

There are various paths around the level for each player to choose. The level design is made in such a way, that each of the paths have their own advantages and disadvantages for the player. For example, the middle of the level is a bit harder to reach because of the choke point, but ranged classes will still be able to hit the other players from this centre. On the sides of the level are traps, which makes it harder and riskier for the players to take this path (another player might push then into the traps or into the water), which can push the player to choose for the centre of the level.

The map is made nearly symmetrical, in order to obtain a certain level of fairness for all players participating in the game. Only the top and bottom sides are different from the left and right side, in order to keep the level design interesting.

# 5. Section V - Interface

## 5.1. Visual System

### 5.1.1. HUD

The HUD displays the players' class, cooldowns, and current kills and deaths. The HUD is placed on the top of the screen, which seemed like the best position to not get in front of the players. The HUD seemed to go unnoticed by most players during test sessions, however. The players' identities and health bars are located above their corresponding characters for identification and not having to look away from the characters to find out how they are doing.. The rangers have also received an aiming gizmo, to indicate where they are shooting, and how far.


Figure 1 - In-Game HUD located at the top of the screen and above the characters' heads.

### 5.1.2. Menus

The game starts off with a start screen, which offers the players to start playing, or exit the game. If the players choose to exit, they will be prompted with a confirmation screen, but if they choose to play the game, they will be shown the control and character selection screen. The character selection screen offers every player that enters their own menu of control types to choose and characters to select. Finally, there is the post-game menu that displays when a player has reached the game its objective of obtaining five kills. The only option here is to continue to the start screen.

The mobile controller has its own menus. The starting menu allows players to enter the IP address shown on the Pc, and the second menu is created for character selection.

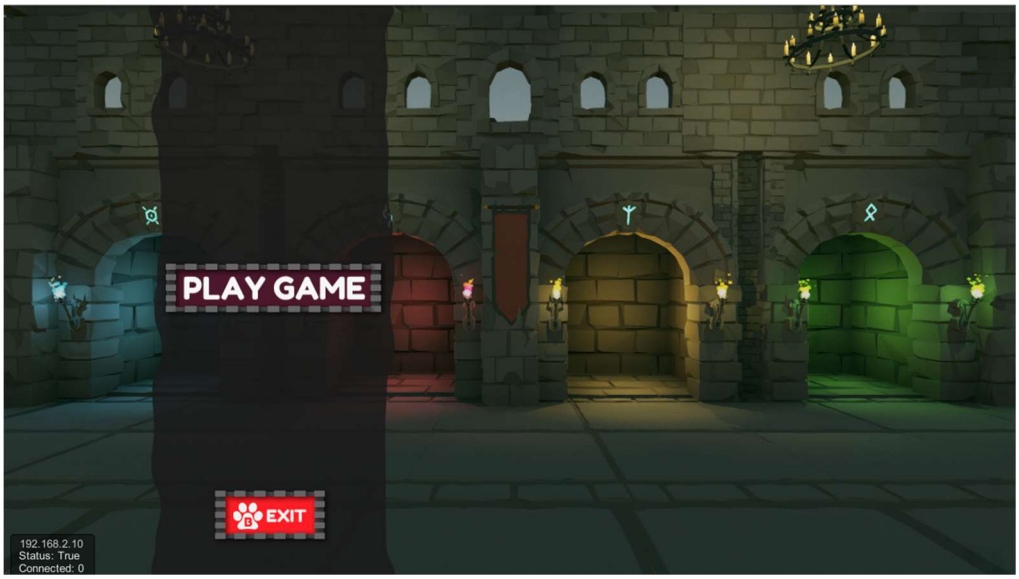



Figure 2 - Starting screen.

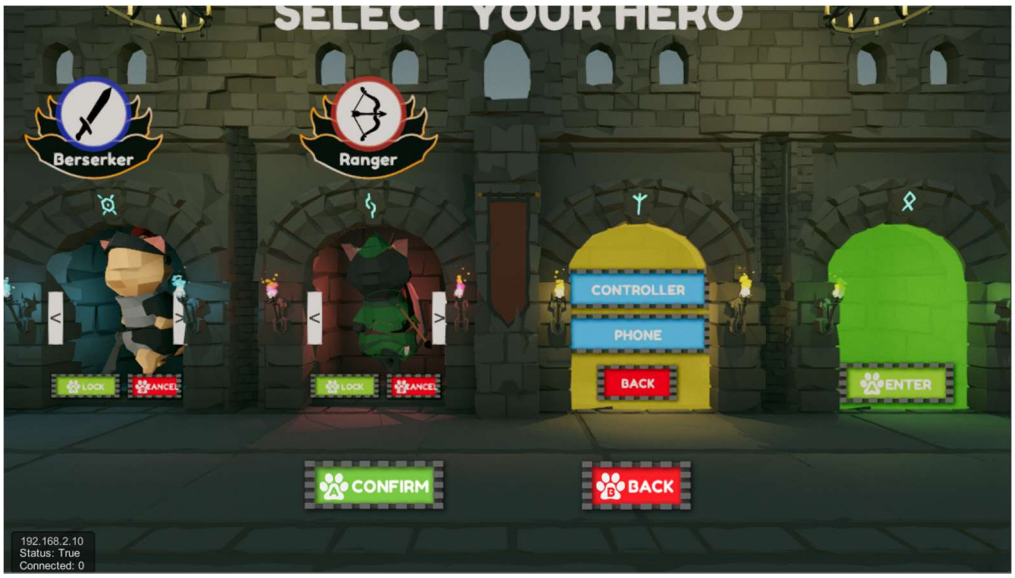



Figure 3 - Character selection screen.

Figure 4 - Post-game recap screen.


Figure 5 - Quit confirmation screen.

### 5.1.3. Rendering System

Unity its standard rendering system is used in the game. This rendering system is efficient, and light to run. This allows for great compatibility, and easy porting to multiple platforms.

### 5.1.4. Camera

For the camera, a dynamic camera system was applied to the game, which allows the camera to move and keep the players in the camera FOV at all times. when the players get closer together, the camera moves closer, whereas if the players move away from each other, the camera moves further away to fit every character.

### 5.1.5. Lighting Models

The standard lighting model created by Unity was used in this project.

## 5.2. Control System

The controls differ per input mode used, as the phone has its own custom control scheme with specifically created buttons. Regularly used controllers have a universal layout and, logically, the keyboard only has one.

| | Dualshock 4 | XBOX | Keyboard |
|---|---|---|---|
| Move | Left Analog | Left Analog | WASD / Arrow Keys |
| Roll | X / R1 | A / RB | Space |
| Attack | □ / L1 | X / LB | E |
| Ability | O / R2 | B / RT | R |
| Ultimate | △ / L2 | Y / LT | Q |

## 5.3. Music

One musical track is implemented in the game, which plays on a loop from the initialisation of the game, up until the game closes. The track is a royalty free song downloaded from an online forum.

## 5.4. Sound Effects

The game has a multitude of sound effects implemented for the in-game actions the players can take. All implemented sounds come from a source that distributes royalty free game sound effects.

# 6 Section VII – Technical

## 6.1 Target Hardware

### 6.1.1 Pc

The main game is designed to be a pc game, because most individuals from the target audience will own a pc. However, the target audience will not always have the best of the best pcs. The game must be able to run on a pc with medium graphics.

### 6.1.2 Phones

The phone application has to be designed in such a way that any smart phone is able to use it without problem. Therefor, it must be noticed that all phones have different graphics or a different screen size.

## 6.3 Game Engine

The game will be created in the Unity engine. Unity gives users the ability to create games and interactive experiences in both 2D and 3D, and the engine offers a primary scripting API in C#, for both the Unity editor in the form of plugins, and games themselves, as well as drag and drop functionality.

## 6.4 Network

The game needs a network to work between the main game and the application controllers. The network will work on a local basis, which means that the game and phone must run on the same Wi-Fi network to be able to connect.

## 6.5 Scripting Language

The scripting language that the game will be created in is C#. C# is intended to be a simple, modern, general-purpose, object-oriented programming language. The language is the current primary programming language used for Unity.

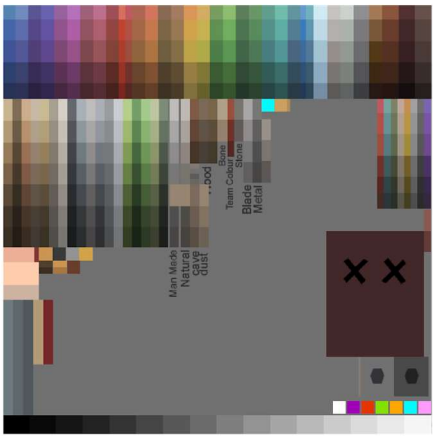
# 7. Section VIII – Game Art

## 7.1. Style Guides

### COLOUR SCHEME

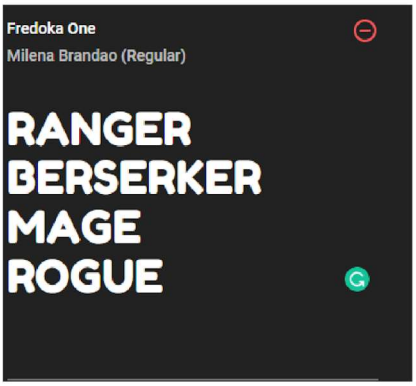
UI COLOURS

TEXTURE ATLAS




### UI ELEMENTS

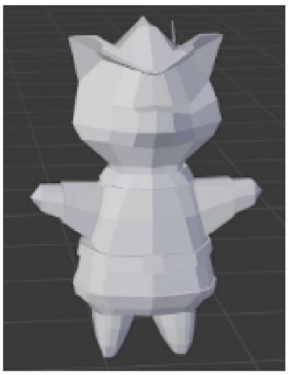
FONT




BUTTONS PLAYERS


### 3D MODELS

BASE MODEL



*Please note that this is the closest resemblance to the base model, that we have a picture of.

## 7.2. The Environment

Firstly, a mood board was created in order to figure out the best style and mood for the arena. At the beginning of this project, the focus was on a cyberpunk-like theme for the environment, for which the following mood boards were created.

Figure 1 - Previous mood board for the environment

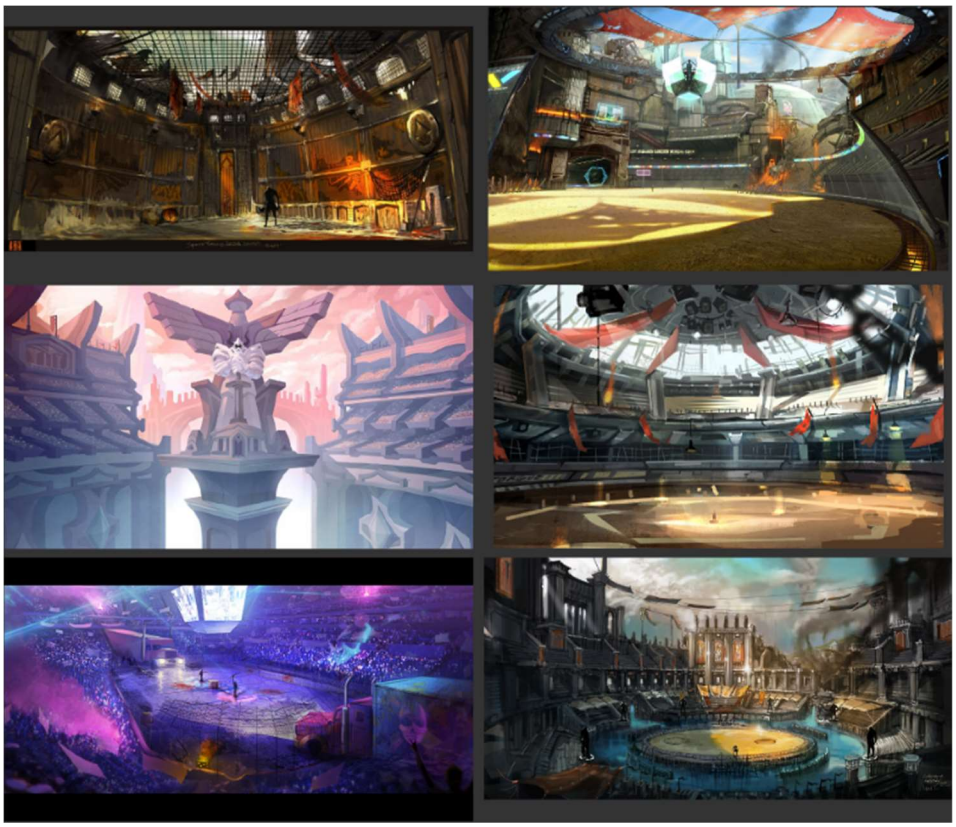


Figure 2 - Moodboard for futuristic arenas.

After a few meeting with the group, it became clear that the environment would be too different from the characters, who were going to be created in a stylized, ‘wacky’ or ‘cute’ style. According to this, the style of the environment was changed into a style with stone, ruins, maybe even castle-like. The mood board can be seen below. It is clear that, even though the mood board shows a lot of stone and bricks, there is also nature involved in the theme. This was also applied in the final level.

Figure 2 - Arena mood boards

After the mood board was created, the final level design was started. There have been a few iterations of the arena through the process. The first figure underneath shows an early 3D arena, in the traditional sense of an arena. The second one is a later version of it, created in a more garden-like environment, to explore all possibilities and to see how far we could go into the ‘wacky’ and ‘cute’ styles. After feedback about how the contract between the arena environment and the cat-like characters actually complemented the game and made it more interesting, the environment was changed back to the arena-themed environment.

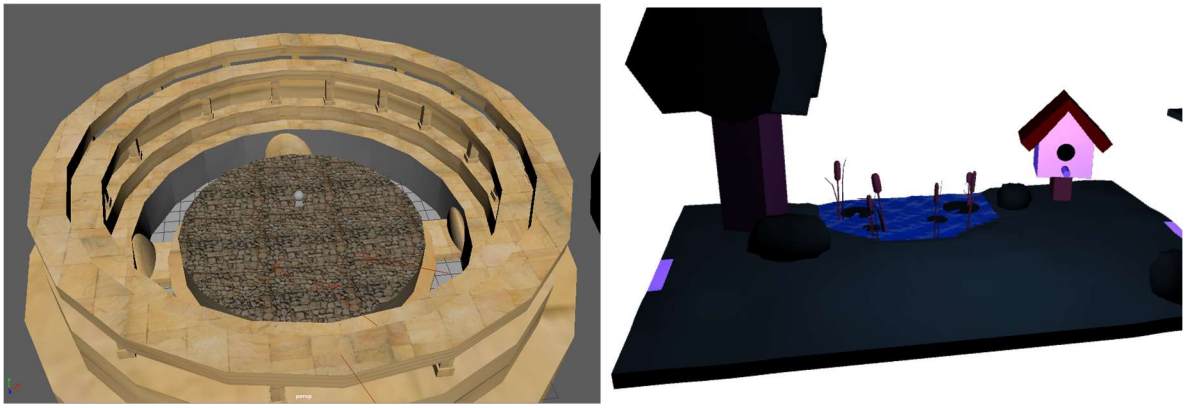


Figure 3 - Early versions of the arena

Figure 4 - Sketch of the arena

In order to be able to fully focus on the design of the game and to create a well-thought-out design, the decision was made to use an Asset Pack from the Unity Store, for the environment of the game.

*POLYGON – Dungeons Pack*, by Synty, was used in order to create the dungeon environment and feel in the game. The pack has a great range of assets that fit with our style and create an aesthetically pleasing level. The assets can also easily be used to create more than one map. (Synty Studios)

At first, an low-detailed arena was created to form a base for the layout, and give a sense of look-and feel to create the final design from. The map contains obstacles for the players to move around and take cover behind, traps for the players to utilize and evade, water to be cautious of, and basic lighting and fire particles. The floor is coloured to look like sand, and the walls are meant to look strong and impressive. There is a big gate added to show there might be a gigantic creature coming, as well as four smaller gates.

Figure 3 - Early version of the arena created with the POLYGON DUNGEON asset pack, in editor with post processing.

The set-up of the final level without rendering, can be seen below. The materials that were used are mainly brick walls and floors, brick pillars, some small accessories like vines, grass and crystals, and water. Traps were also used, like the round grinder on all four sides of the level and the square brick statues in between, which are also “traps” and sprinkle water while rotating 360 degrees.

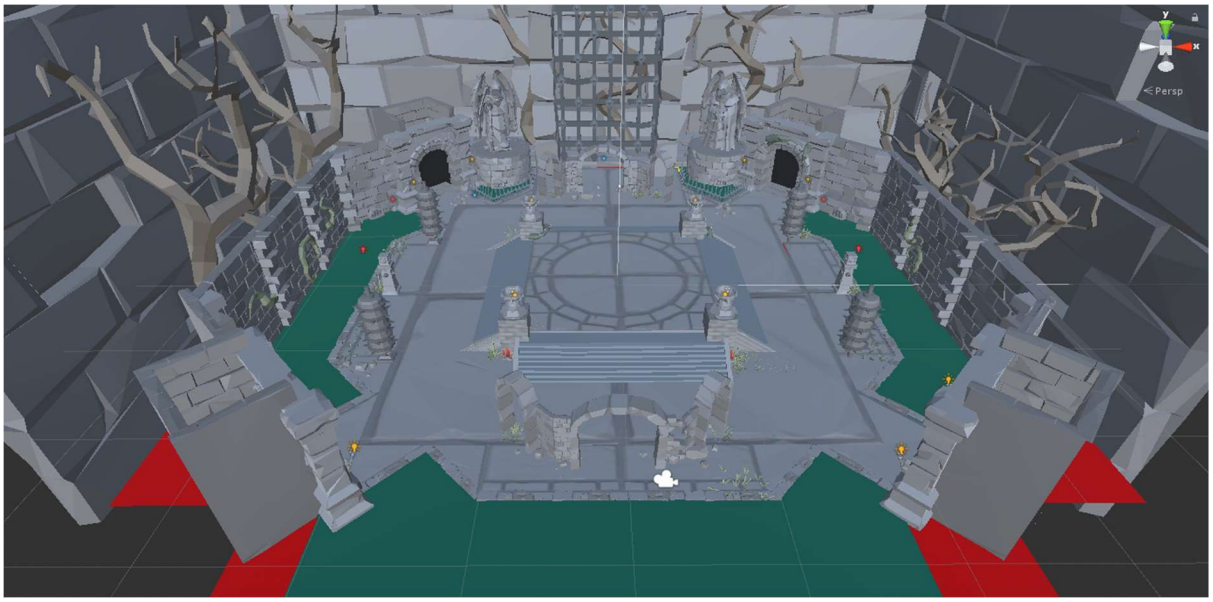



Figure 5 - Arena setup in Edit-mode, without rendering.

In the figure below, the final and rendered arena can be seen. These are also particle systems added to the chalices in the middle of the level and on the water on the outskirts of the level. A character model is added in the corner of the level, for scale.

The general purpose of the aesthetics, with a lot of stone and nature (grass, vines, trees, crystals) details, was to communicate to the player that this level is in an old ruin, with possibly

magical forces going on (because of the crystals and chalices). These details and also the light in the level (which is kept low) are used to strengthen the feeling of mystery, but also possibly danger (because of the dead trees in the background and the vines).

Figure 6 -In-game level, fully rendered

The particle systems in the game were added with purpose. The particle systems and light systems in the chalices in the middle of the game, are added to attract the attention of the player to the middle of the game. This centre is where the players will most likely meet and this is also where pickup able items might spawn more often. These items will also spawn outside of the centre, but the risk will be higher to pick them up there (due to traps and the water around the level). Although the centre also poses a certain risk, since it includes a choke point, where players will be forced closer together.

The particle system in the water, which features splashes with a green colour, is put there to sign to the player that the water is a dangerous area, even though it has a normal blue colour. The colour green was chosen for the splashes, since this normally makes players think of acid or another dangerous, green fluid, and will thus sign to the player that touching the water will be bad for their progress.

Other details that have been used in order to show importance to the player, is the circle of the floor in the middle of the level. This makes the centre stand out more and draw the player's attention to this point. The round traps were also chosen with this purpose, because they stand out from the level more and make it easier to evade them. The spikes emphasise the danger of touching them.

A last aspect of the level design is the level of detail that is used. In the middle of the map, extra details are added around the higher square, like grass and a few crystals. The rest of the map has less of these small details.

## 7.3. The Characters

For the characters, cat-like creatures that walk on 2 legs were chosen, because of the research done on the target audience. The mood board for the character art can be seen below. The style is quite stylized and the characters are chosen with the goal for the players to be able to sympathize with them. Because of this, especially characters with “cute”, but still rather stylized-realistic features were chosen.

The cats in the game are used in order to make the game feel more casual, with a “cute” and stylized style and very cat-like accessories (with paws and bells on it), to make the arena feel more different and interesting and opposite from the characters of the game, which makes both the arena and the characters stand out more.

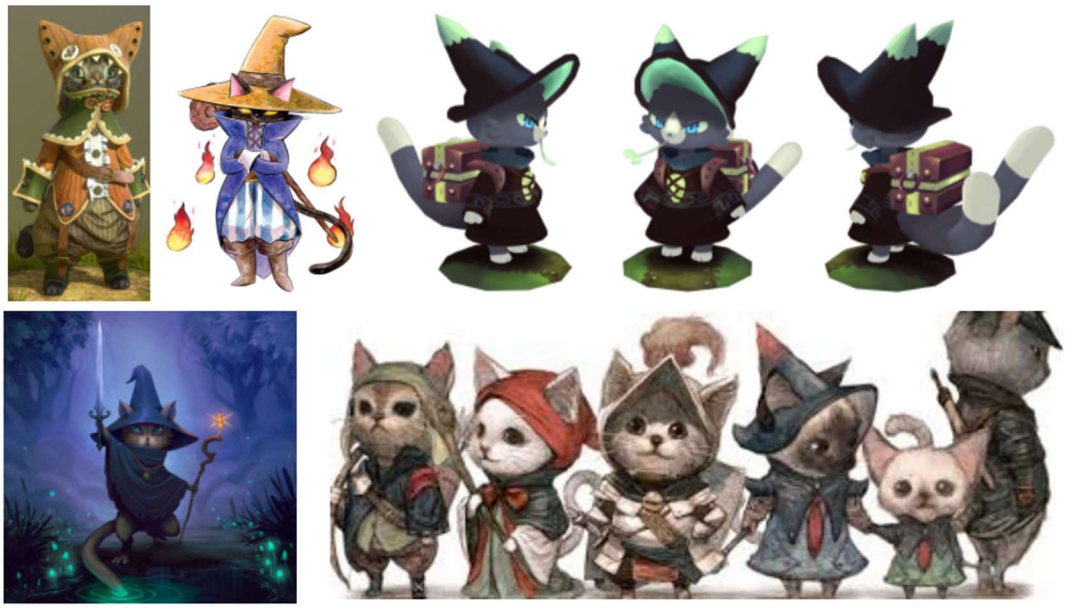


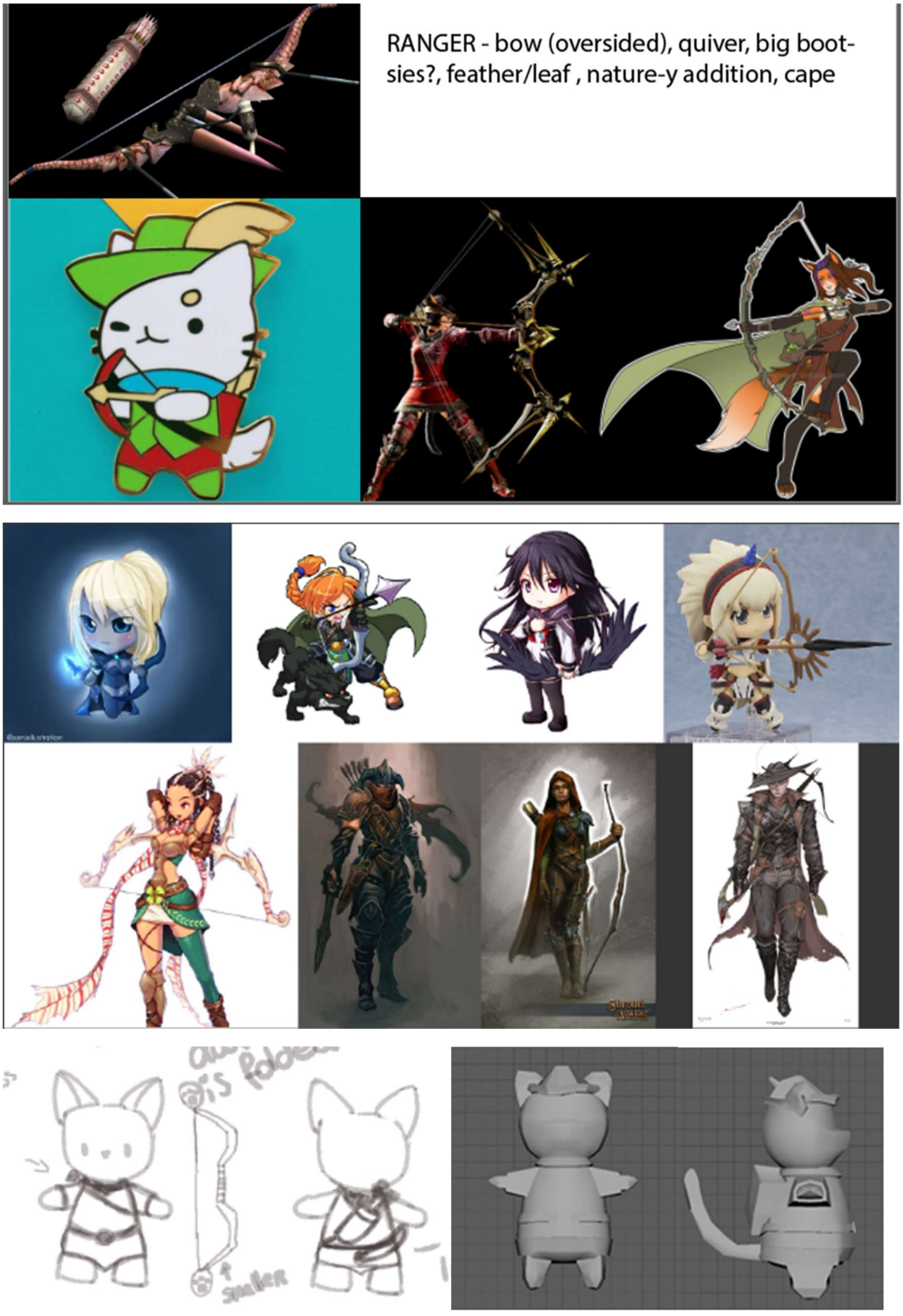
Figure 7 - Character mood board

Like mentioned before, the style we are striving for, is 3D and stylized. This style was applied by making sure defining aspects were enlarged and made more obvious. For example, the hat of the mage.

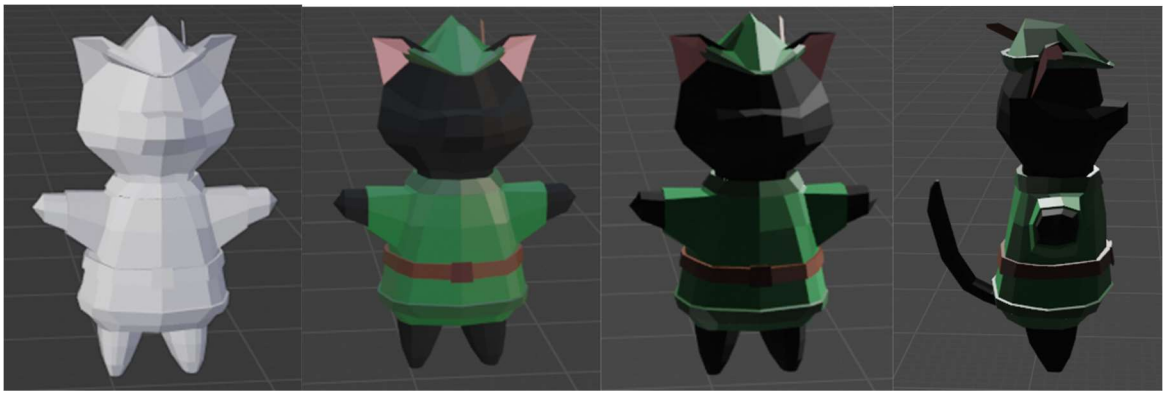
The first step for the concept art was to iterate on the right shape of the head and fur patterns. These patterns have not been implemented yet, since the character’s textures have not yet been applied in the game.


Figure 8 - Details and fur patterns

## Ranger

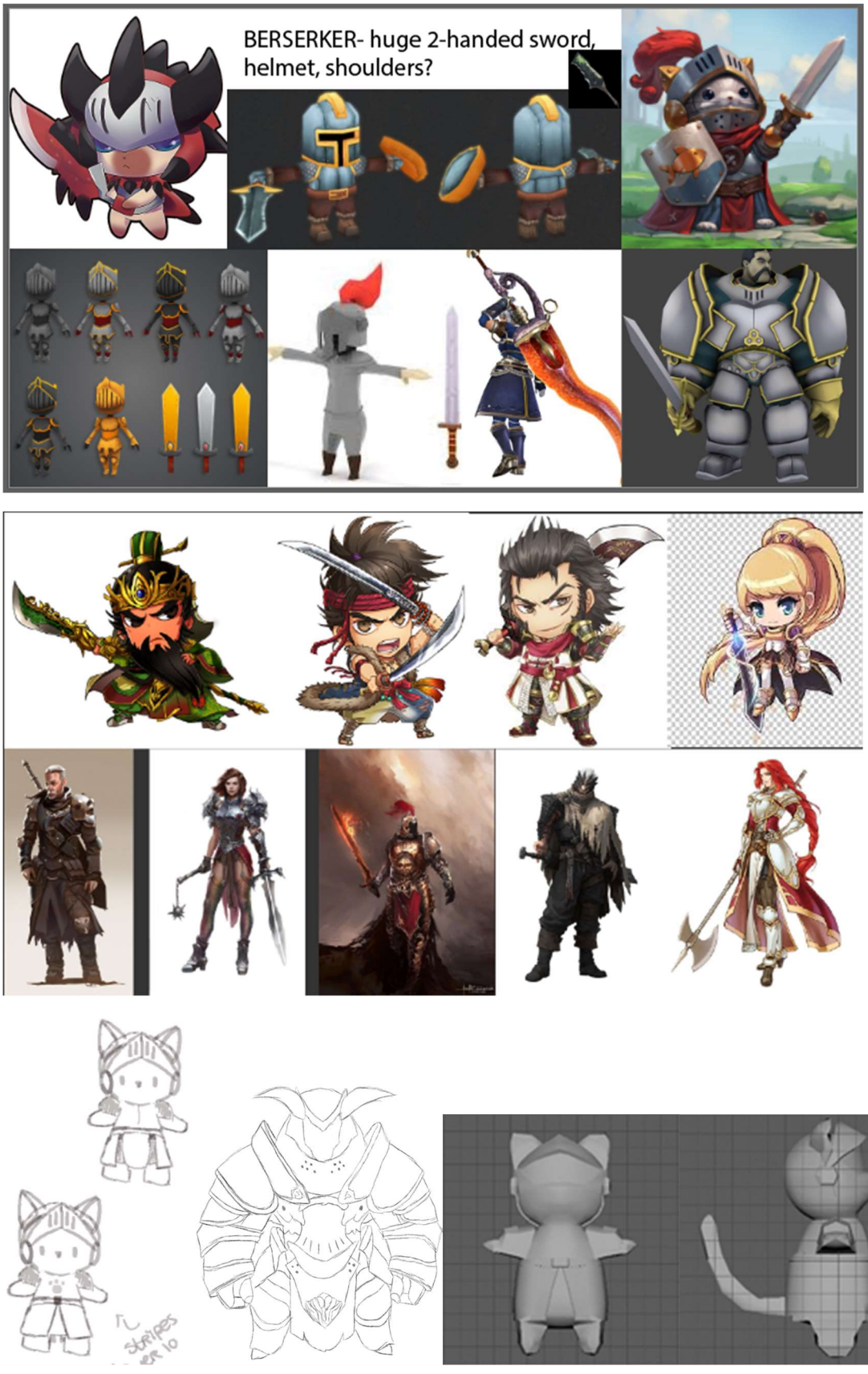



Figures 9-12 - Ranger mood board and early concept art (2D and 3D).



Figures 13-16 - Final Ranger model, with and without colours and rendering.

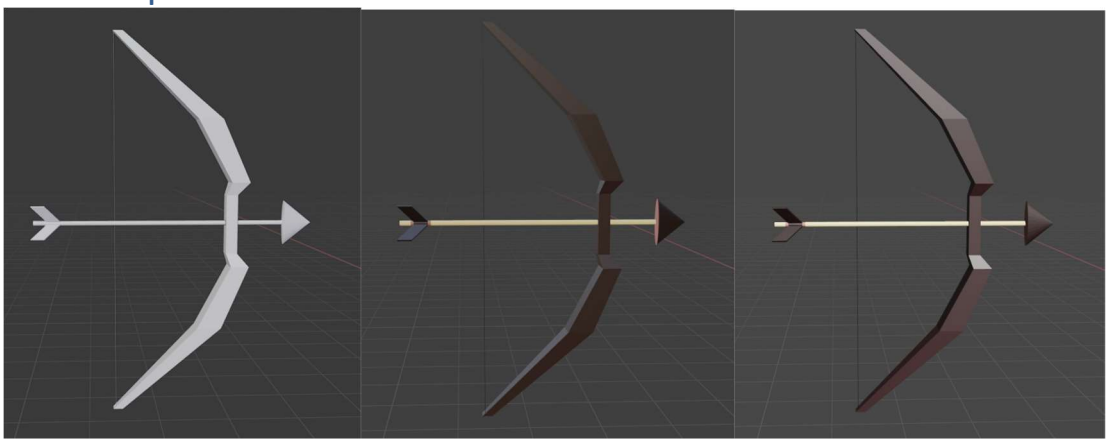
## Berserker




Figures 17-23 - Berserker mood board and early concept art (2D and 3D).

Figures 24-28 - Final Berserker model, with and without colours and rendering.

## 7.4 Weapons and accessories



Figures 29-31 - Bow for the Ranger.

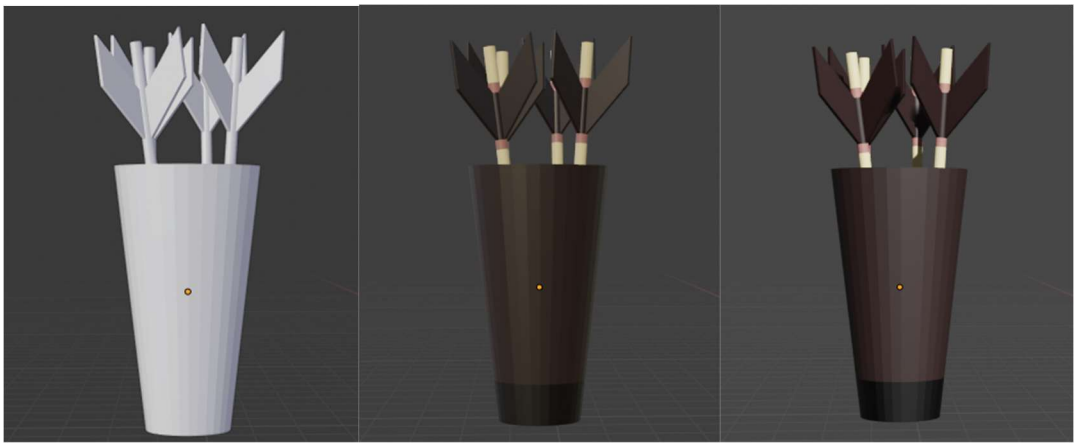


Figure 32-34 - Quiver for the Ranger.3d

Figure 35 - Berserker's blade.

## 7.5. Additional 3D models

### 7.5.1. Pick Up Objects

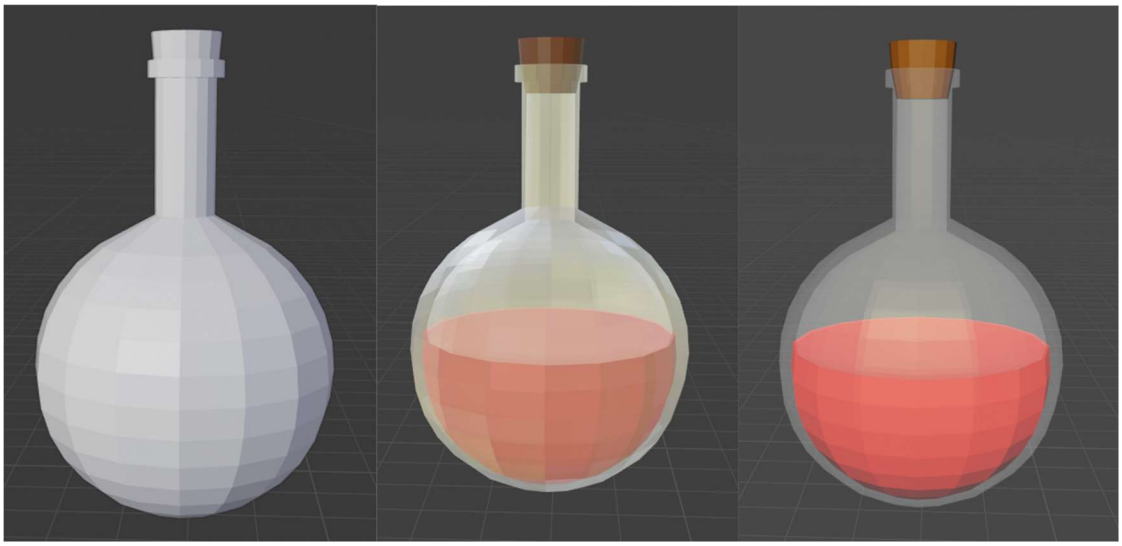


Figure …..- Earlier version of the healing-object, which was eventually changed to a heart, for visual clarity.

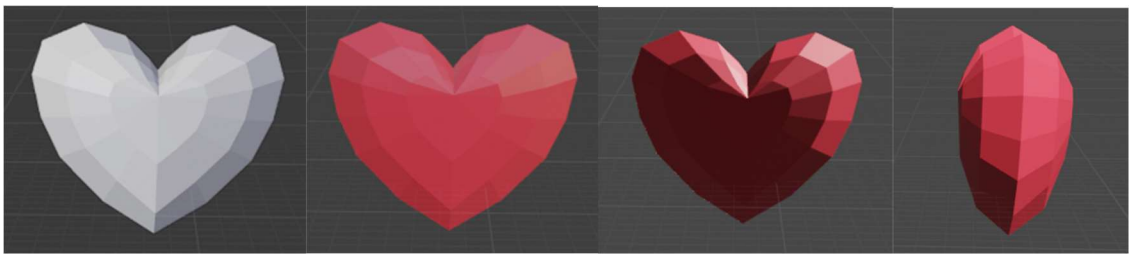


Figure …..- Heart-like 3D model, which can be picked up in-game, to replenish health.

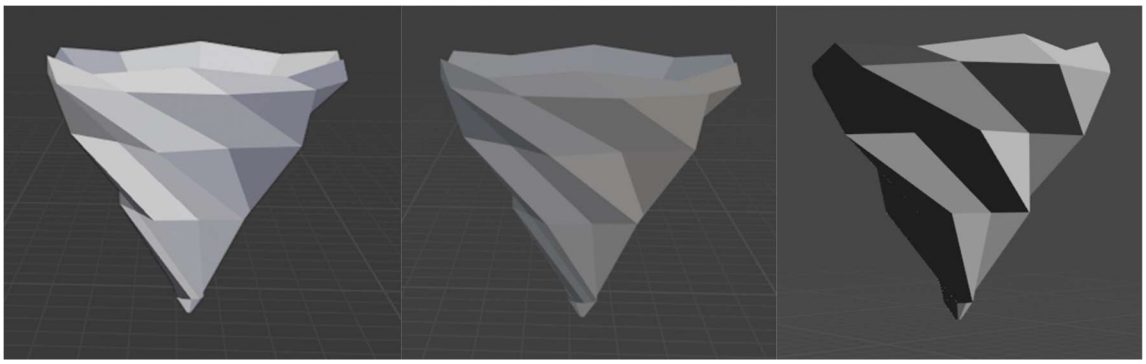


Figure …..- Tornado-like 3D model,, which can be picked up in-game, to push away enemies

### 7.5.2 Extras

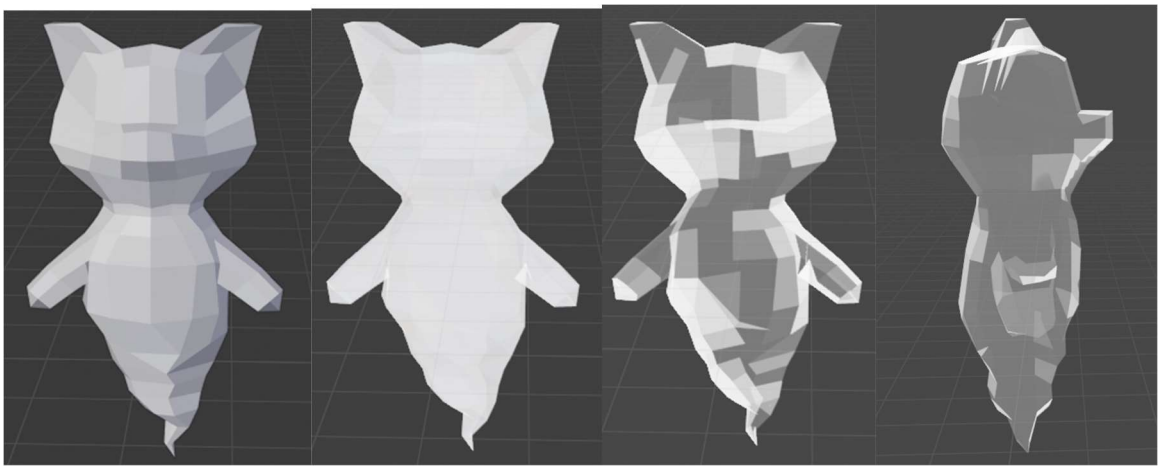


Figure …..- Ghostly cat, which appears and floats away once a player has died.

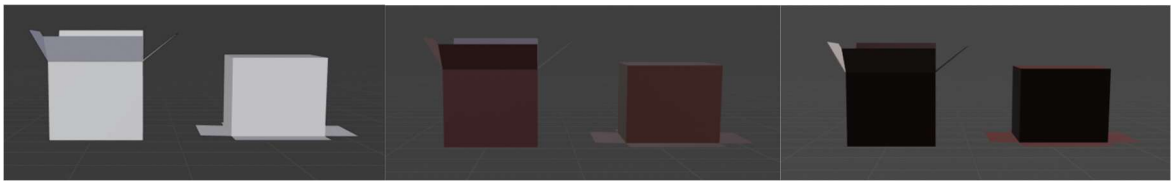


Figure …..- Box, used for the CC-attack of the Ranger. First model is thrown, second model is used if a player is captured.

## 7.6. Extra’s

Figure …..- Loading screen in-game.

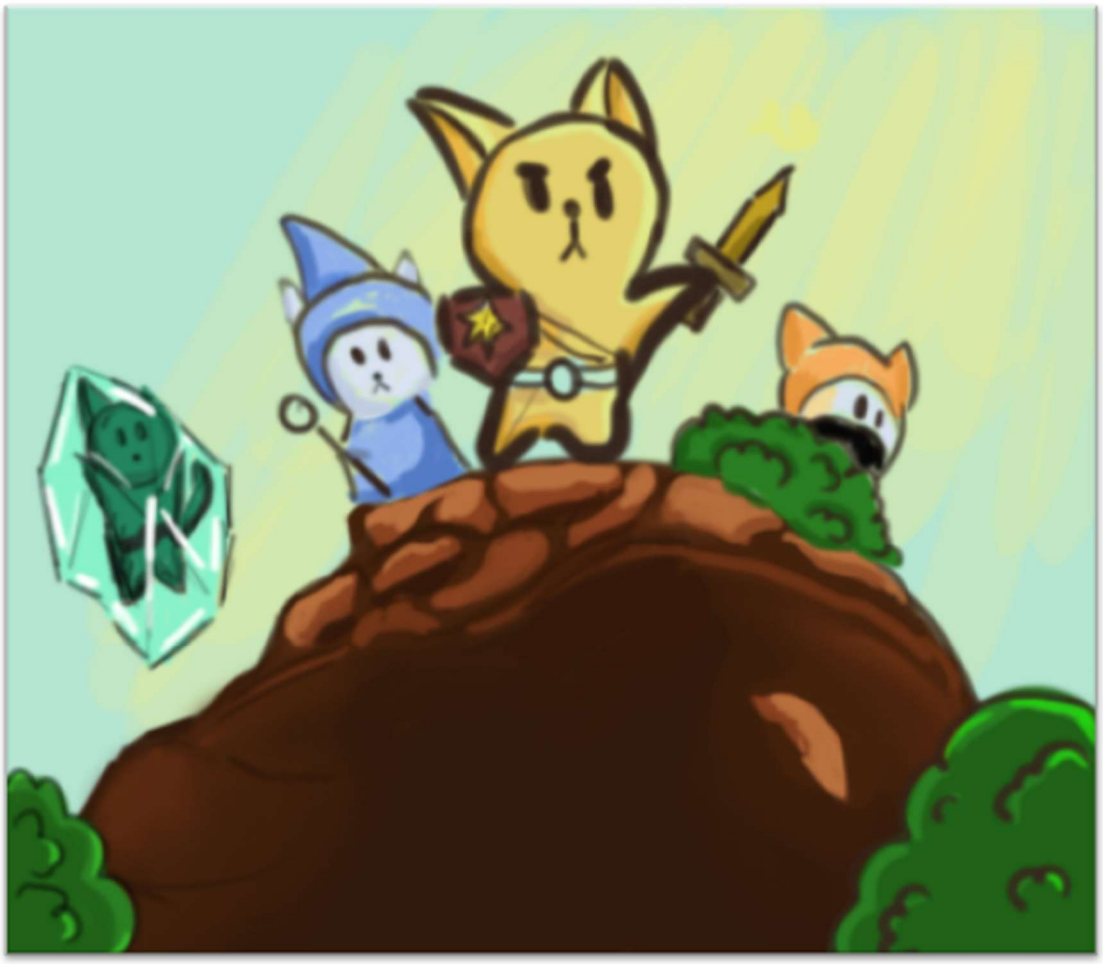


Figure ….. - Early concept art

# 9. Section X - Planning and management

## 9.1. Scrum

As a planning method Agile Scrum is being used for maximum productivity and control over the project. Current Scrum roles are:

- Chiril Ojoga - Product Owner.
- Machteld Dalhuisen - Scum Master
- Imme van der Made and Robin Potze - Development team.

### 9.1.1 Planning

As a way to control tasks needed for the project Product Backlog is used. All the tasks in Product Backlog are located based on level of importance from top - down. All the tasks in product backlog have an estimated time value and category tag attached to it.

### 9.1.2 Sprints

Every week a scrum meeting is gathered where tasks from product backlog were taken split into tickets, assigned to each team member and pinned to current's week sprint board.

### 9.1.3 Daily standups

To track development process and make sure that every member of the team is up to date, daily standups were used where every member of the team said what he/she did since previous stand-up, what went well and wrong, what he/she is going to do next.

## 9.2. Budget (Value proposition)

### 9.2.1 Costs

First we wanted to calculate the costs that we have made and will be making.

*Instant*

| | | |
|---|---|---|
| Assets Unity Store | € | 125,00 |
| KVK Register | € | 50,00 |
| TOTAL | € | 175,00 |

*Constant per month*

| | | |
|---|---|---|
| Salaries Development Team | € | 10.000,00 |
| Unity Seat | € | 16,00 |
| Promotion Costs | € | 2.500,00 |
| Communication Channels | € | 2.000,00 |
| Publishing rights (steam) | € | 86,00 |
| TOTAL | € | 14.602,00 |
| Unforeseen costs | € | 1.460,20 |
| Needed | € | 16.062,20 |

From this we know that we will need at least €16.062,20 per month to break even with the costs. We wanted to calculate the revenue per quarter, which means that the costs are going to be (€16.062,20 * 3 =) €48.186,20.

### 9.2.2 Price per unit

We decided that we were going to ask €20,- for the license. This means that we need to sell at least (€48.186,20/ €20 =) 802 units per month, and (€48.186,20 / €20,- = ) 2409 units per quarter.

### 9.2.3 Unit goals for the first year

With this in mind we decide to set the goals per quarter in the first year;

| | | Y1 | | | |
|---|---|---|---|---|---|
| | *Q1* | *Q2* | *Q3* | *Q4* |
| Sale Units | 750 | 2150 | 3000 | 3750 |
| *Price* | € *20,00* | € *20,00* | € *20,00* | € *20,00* |
| **Sale Profits** | **€ 15.000** | **€ 43.000** | **€ 60.000** | **€ 75.000** |
| | | | | |
| **Revenue for units** | **€ -33.186,60** | **€ -5.186,60** | **€ 11.813,40** | **€ 26.813,40** |

### 9.2.4 Skins

We also thought of selling character skins in three tiers. For which we have set the following prices;

| | Low | Mid | High |
|---|---|---|---|
| Skins | € 2,00 | € 4,50 | € 7,00 |

The goals for the skins in the quarters of the first year are;

| **Skins** | | | | |
|---|---|---|---|---|
| Low Units | 36 | 108 | 144 | 180 |
| Low Revenue | € 72,00 | € 216,00 | € 288,00 | € 360,00 |
| Mid Units | 30 | 90 | 120 | 150 |
| Mid Revenue | € 135,00 | € 405,00 | € 540,00 | € 675,00 |
| High Units | 15 | 45 | 60 | 75 |
| High Revenue | € 105,00 | € 315,00 | € 420,00 | € 525,00 |
| **Total Skin Profits** | **€ 312,00** | **€ 936,00** | **€ 1.248,00** | **€ 1.560,00** |

### 9.2.5 Total Revenue

Which means that the total profits and revenue are per quarter;

| | | | | |
|---|---|---|---|---|
| **Total Profits** | **€ 15.312** | **€ 43.936** | **€ 61.248** | **€ 76.560** |
| | | | | |
| **Revenue** | **€ -32.875** | **€ -4.251** | **€ 13.061** | **€ 28.373** |

## 9.3. Design research

### 9.3.1 Testing

On the demonstration day we conducted a test via a survey. We asked the questions;

- *How easy did you think the game was? Can you explain why?*
- *Do you like the phone controller?*
- *What did you like most about the game?*
- *What did you not like about the game?*
- *Was there any feature in the game that did not work as you expected?*
- *What did you think about the art style?*
- *What do you think about the mood in the game?*
- *Is there something that stands out, art-wise?*

The feedback that was returned was positive in general. People liked the local co-op PVP. They liked the art and the mood of the game as well. However, they did not like the UI, that the traps were dark and they said that the berserker was so much stronger than the ranger.

Some also complained that it was not clear what controls you had to use.

## 9.4 Future additions

### 9.4.1 Additional character classes

As one of future additions to the game, 2 new player classes are planned to be added. New rogue class is planned to be a melee fighter with fast movement and fast attacks. On the other hand, new Mage class will be a ranged fighter with heavy magical attacks. WIth four classes in total players will always be able to find what is an appropriate playstyle for them.

Figure ….. - Art for the upcoming mage class.

### 9.4.2 New game mode

Another addition will be a new game PVE mode for the game. Players will have to fight together against enemy waves. This mode would provide players with cooperative, but also competitive experience as player with highest score wins the game. New PVE mode is meant to have 4 enemy types: two casual enemies with different attack, one champion enemy for the end of each wave and one boss enemy for the last wave.

### 9.4.3 New aiming system

Player aiming system will also be changed. Based on data gathered during testing it was concluded that it's hard to control character's attack direction, because it's bound to character movement direction. To solve this issue it was decided to use second stick to control attack direction of the character.

### 9.4.4 New special attack

It is planned to add an Ultimate attack for every character. This would provide players with a way to deal sufficient damage and make the game visually epic.

### 9.4.4 Added UI features and screens

Planned additional screens are a hero information screen, control schemes options, pause menu, and tutorial screen. Additional features are attack and status effect indicators, improved identification, and improved HUD.

# Literature

Synty Studios. (n.d.). POLYGON - Dungeons Pack. *Asset packs.*