Codename: Jungle Farmers (working title)

Producer: Alex

1. Game Description

Genre: Top-Down 1 Screen Hoarding Multiplayer (2-4 Players)
Platform: PC/Mac Standalone and XBox
Control: XBox-Controller
Play Time: approximately 30 secs per round, 5 rounds in total per game
Key Features: Indirect Controls, funny Artificial Intelligence

1.1 Background story
The Jungle is a difficult place for doing agriculture. The natural jungle makes it difficult to set borders. And of course none of the farmers want to stay with its cattle all the day. That's why it's common sense, that at the end of the day all the farmers are fighting for every cattle which can be found in the jungle. The fastest and smartest farmer will have the most cattle in the end.

1.2 Core mechanics
The players have to catch and drive the cattle distributed in the jungle into their own stable. The game is set on a single screen. The players are spawned in the corners of the map, where their stables are located. All the stables, jungle clearings and points of interest are connected through jungle paths. But the paths and location of the cattles are always changing and each round is different.

1.3 Micro mechanics

- **Limited resources provoke the fight** – There is only limited cattle in the jungle and players have to fight for the ownership. Therefore a player can push other players and try to push the cattle in another favored direction.
- **Procedurally generated map** – Players have to adjust their play style to the different maps.
- **The cattle has its own will –** Sometimes the cattle isn't reacting as the player wants it to.
- **Blocking ways** – To guide the way of the cattle and to interrupt other players, the player can temporarily block ways.

1.4 Visual Style
The visual Style needs to be clear, so every player can keep the overview. Because of the tile-based map generation, the field will have a pattern-like look. To underline the fun and party-chaotic gameplay the game needs a colorful and striking design. Everything will happen on a single screen.

1.5 Controls
Left analog stick to move around. A for direct interaction and B for using an item.

2. Technical Achievement

Different AI behaviour:
To make the game more interesting, each animal should behave differently. Every different type of animal should behave significantly different. Additionally, each animal should behave in a slightly unique way, but not too different to the stereotypical animal in its group. This way, the players are able to learn the general behaviour of a group of animals and can use that knowledge to better manipulate them.

Procedural generation of levels:
A big challenge here will be that the levels don't always feel the same, while staying fair all the time. Things that can be procedurally generated include the position of the bases, the placement of animals along with their type, items and obstacles.

3. "Big Idea" Bullseye

Fun interaction with dumb but cute animals

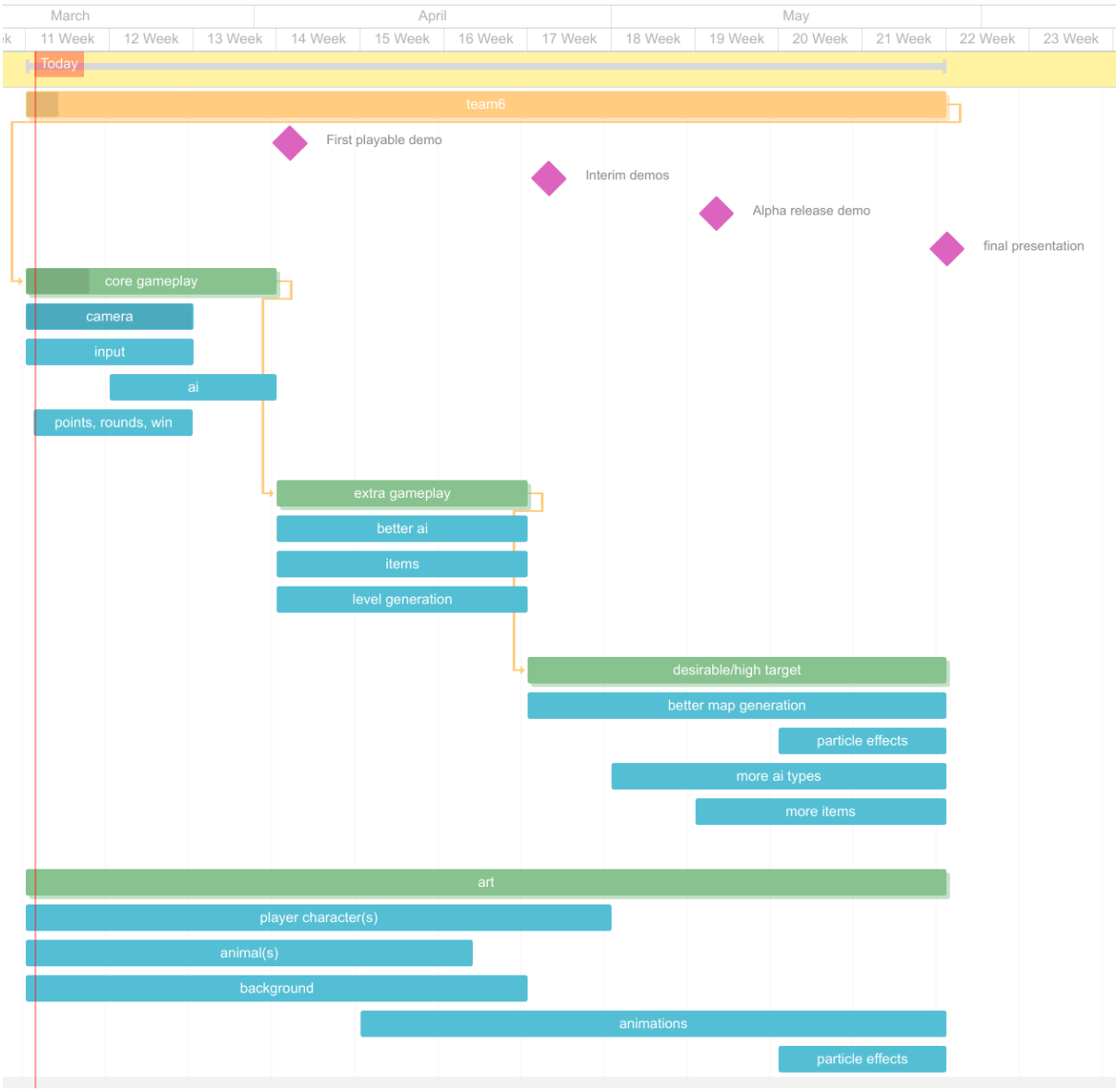
4. Development Schedule

4.1 Layers

1. **Functional minimum**
   This would be the very basics of the gameplay: A very basic map, maybe just a field with some boxes as obstacles. The core gameplay should work well. There will be one type of animal, represented by dots. Players can drive animals into their bases and the player with the most points should win the game.
2. **Low target**
   Make it fun, e.g. add more types of animals (one other type to reach target), make the maps more interesting.
   Replace most of the placeholder graphics with art assets.
3. **Desirable target**
   Polish level generation, such that each level feels unique. Add items to enhance player interaction. Add sound + music to the game.
4. **High target**
   Make it juicy: Add little details to make the game more engaging, such as nice animations, particle effects and fitting sound effects.
5. **Extras (out of scope)**
   Multiple game modes, different theme for maps, online multiplayer, …

4.2 Schedule

Responsible for the programming tasks will be mainly Moritz, Alex and Florian. We will not distribute tasks upfront, but decide individually based on priority during development.
The main responsibility for the art assets lies with Sonja and Stefan. They too will work on what's most important during each stage of development.
Things like playtesting and design related decisions is of course everybody's task.



5. Assessment

Imagine four friends sitting on a couch on a friday evening. Every player gets a corner of the screen assigned as his base. From there he starts to collect animals walking around the jungle. To stop others from succeeding one can interact with other players. The animals will walk away from players. The animals give the players a cute feeling because of their art-style and also the way they move on the map.
The most cool thing about the game will be, to build up a strategy to bring as many animals to the base without being too much distracted by the other players.
The main criteria to judge the game will be how much fun the four friends have while playing the game and if they would play it again a week after.

Codename: Jungle Farmers (working title)

Producer: Alex

1. Game Description

Genre: Top-Down 1 Screen Hoarding Multiplayer (2-4 Players)
Platform: PC/Mac Standalone and XBox
Control: XBox-Controller
Play Time: approximately 30 secs per round, 5 rounds in total per game
Key Features: Indirect Controls, funny Artificial Intelligence

1.1 Background story

The Jungle is a difficult place for doing agriculture. The natural jungle makes it difficult to set borders. And of course none of the farmers want to stay with its cattle all the day. That's why it's common sense, that at the end of the day all the farmers are fighting for every cattle which can be found in the jungle. The fastest and smartest farmer will have the most cattle in the end.

1.2 Core mechanics

The players have to catch and drive the cattle distributed in the jungle into their own stable. The game is set on a single screen. The players are spawned in the corners of the map, where their stables are located. All the stables, jungle clearings and points of interest are connected through jungle paths. But the paths and location of the cattles are always changing and each round is different.

The cattle can be guided through the level via two main mechanisms. One way of guiding them is by pushing them away from the player. This could for instance happen via shouting to scare the cattle. This can be used as a way of interfering with your opponents and at the same time to herd animals to your stable. The other way of interacting with animals is via pulling them towards the player. This could for instance happen by attracting them via food. This is a more precise way to guide animals, but you can control fewer animals that way.

The game will consist of multiple (short) rounds and each round is limited by a certain amount of available time [before sunset]. The exact amount of time will be determined during the course.

1.3 Micro mechanics

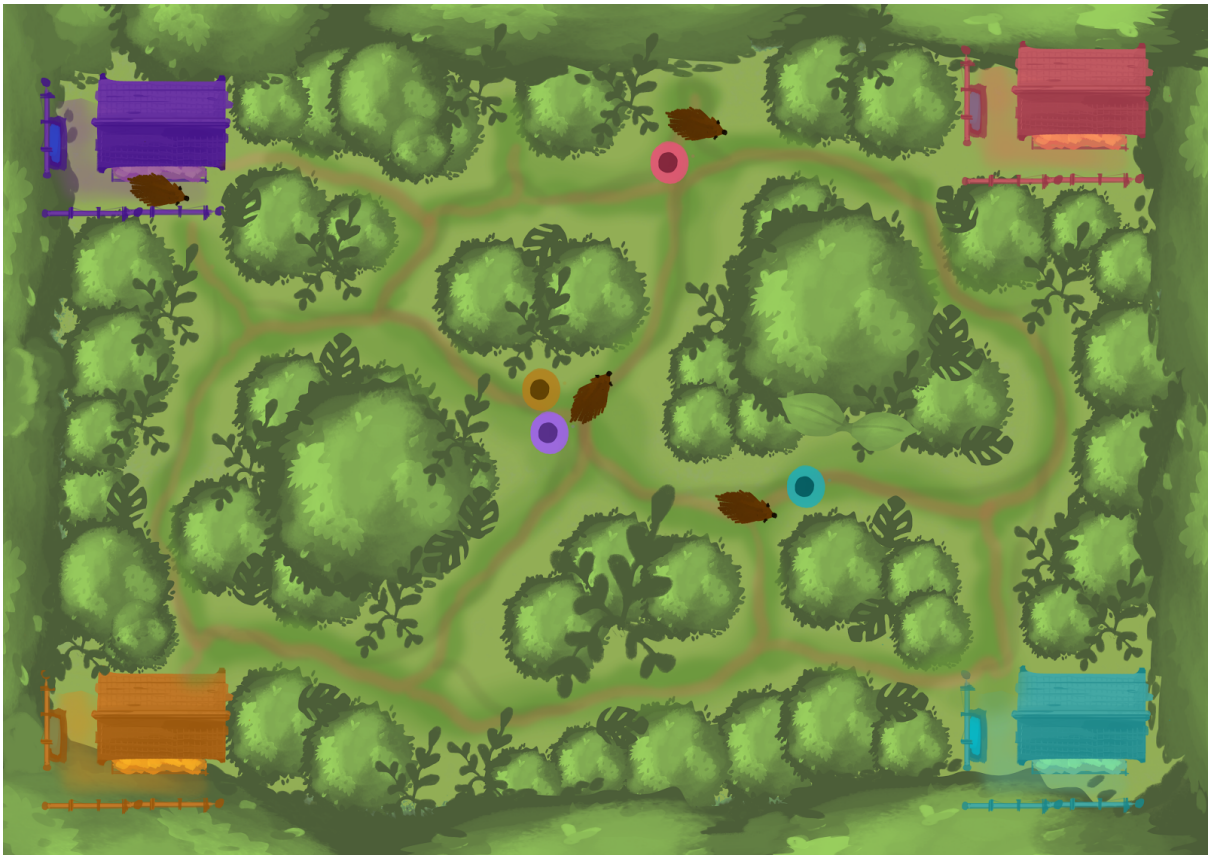
- **Limited resources provoke the fight** – There is only limited cattle in the jungle and players have to fight for the ownership. Therefore a player can push other players and try to push the cattle in another favored direction.
- **(Procedurally generated map)** – Players have to adjust their play style to the different maps.

- **The cattle has its own will –** Sometimes the cattle isn't reacting as the player wants it to.
- **Blocking ways** – To guide the way of the cattle and to interrupt other players, the player can temporarily block ways.

1.4 Visual Style

The visual Style needs to be clear, so every player can keep the overview. Because of the tile-based map generation, the field will have a pattern-like look. To underline the fun and party-chaotic gameplay the game needs a colorful and striking design. Everything will happen on a single screen.

Early mockup:



Circles represent players, each player has its matching barn at a corner.

1.5 Controls

Target platform is the Xbox One, hence, an appropriate scheme for using a XBox One Controller will be used. This will (most likely) consist of using the left analog stick to move around and A, B and/or the Bumpers for the interaction with cattle.

2. Technical Achievement

Different AI behaviour:
To make the game more interesting, each animal should behave differently. Every different type of animal should behave [significantly] different. Additionally, each animal should behave in a slightly unique way, but not too different to the stereotypical animal in its group. This way, the players are able to learn the general behaviour of a group of animals and can use that knowledge to better manipulate them.

Different types of animals might flock or roam for example. Some might be easily scared others are more trustful. They might be also dumb: for example they might need explicit guiding around an obstacles.

3. "Big Idea" Bullseye

Fun interaction with dumb but cute animals

4. Development Schedule

4.1 Layers

1. **Functional minimum**
    - Basic map
    - Controllable players
    - One type of animals
    - Core gameplay works
        i. “Pushing” animals away
        ii. “Pulling” animals
    - graphics can still be debug/placeholder graphics

2. **Low target**
    - Scoring and rounds: player with most animals in his barn wins
    - A second type of animal
    - Replace placeholders with real art assets
    - Make map fun - i.e. fine-tune first gameplay
3. **Desirable target**
    - Make AI of each animal type feel unique
    - One animal type with a flocking AI
      Ideas: https://www.youtube.com/watch?v=Bdu5_Q5QI2Q, https://www.youtube.com/watch?v=QbUPfMXXQIY
    - Add sound effects
    - Add background music (a single track is enough)
4. **High target**
    - Add a second map
    - Animated characters and animals
    - Animated scores and matching sound effects

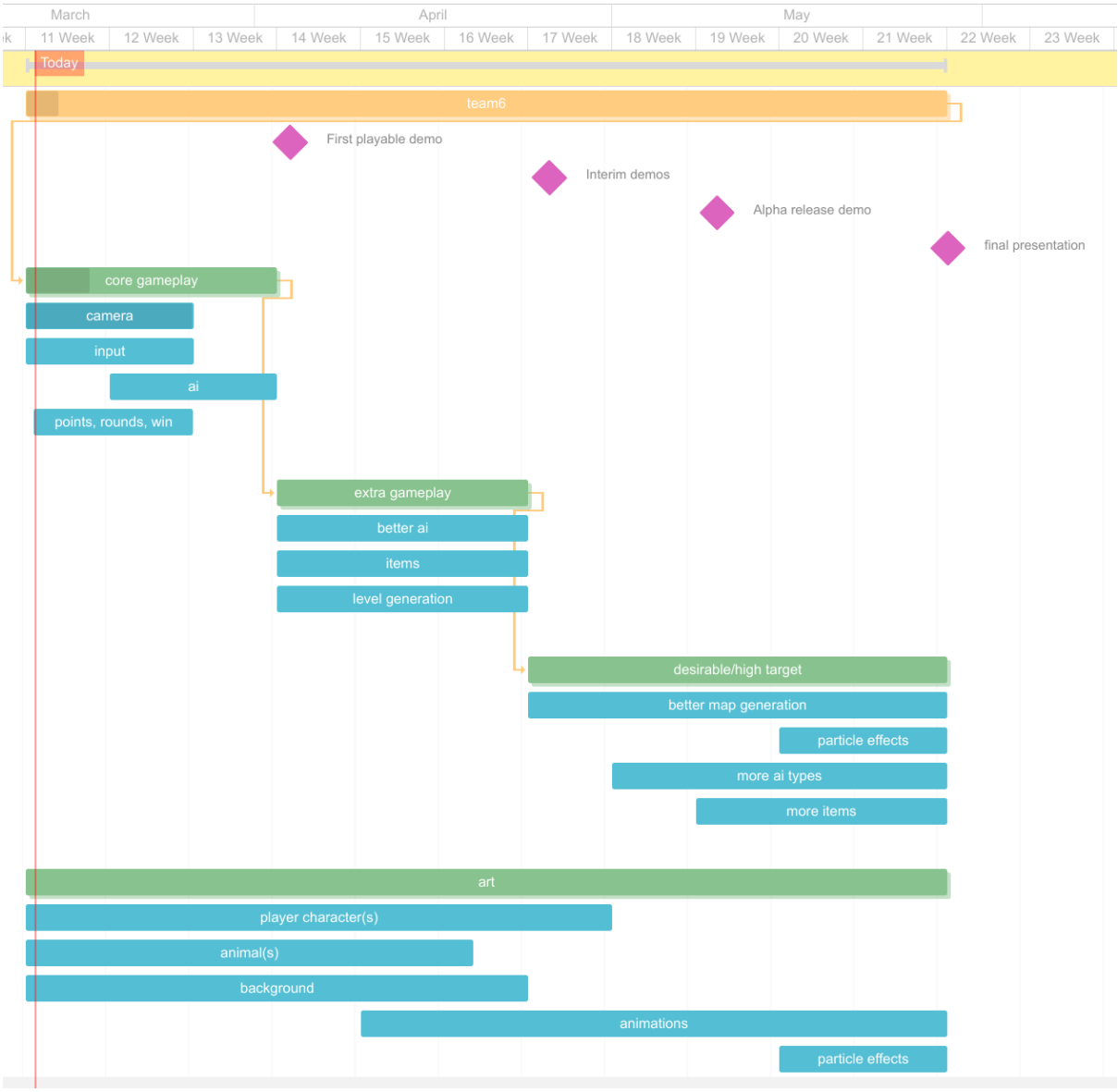
5. **Extras (out of scope)**
    - Different Lighting
    - Themes for maps
    - Different game modes
    - Fighting between players
    - Online multiplayer
    - ...

4.2 Schedule

We will work in an (informal) agile fashion: small sprints (probably one/week) to meet each individual deadline over the duration of the course. We will distribute tasks [at the start of each sprint] based on priority during development.

While Alex, Florian and Moritz will be mainly doing the programming tasks, Sonja and Stefan will mostly work on assets production. Playtesting and designing/making decisions are in the responsibility of everyone.

The following chart gives a rough overview over the planned tasks:


March
April
May
11 Week
12 Week
13 Week
14 Week
15 Week
16 Week
17 Week
18 Week
19 Week
20 Week
21 Week
22 Week
23 Week
Today
team6
First playable demo
Interim demos
Alpha release demo
final presentation
core gameplay
camera
input
ai
points, rounds, win
extra gameplay
better ai
items
level generation
desirable/high target
better map generation
particle effects
more ai types
more items
art
player character(s)
animal(s)
background
animations
particle effects

5. Assessment

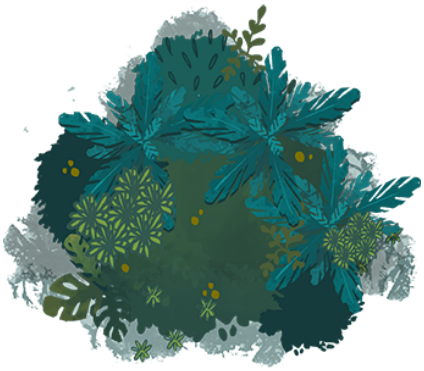
Imagine four friends sitting on a couch on a friday evening. Every player gets a corner of the screen assigned as his base. From there he starts to collect animals walking around the jungle. To stop others from succeeding one can interact with other players. The animals will walk away from players. The animals give the players a cute feeling because of their art-style and also the way they move on the map.
The most cool thing about the game will be, to build up a strategy to bring as many animals to the base without being too much distracted by the other players.
The main criteria to judge the game will be how much fun the four friends have while playing the game and if they would play it again a week after.

Prototype Report Team 6

Prototype

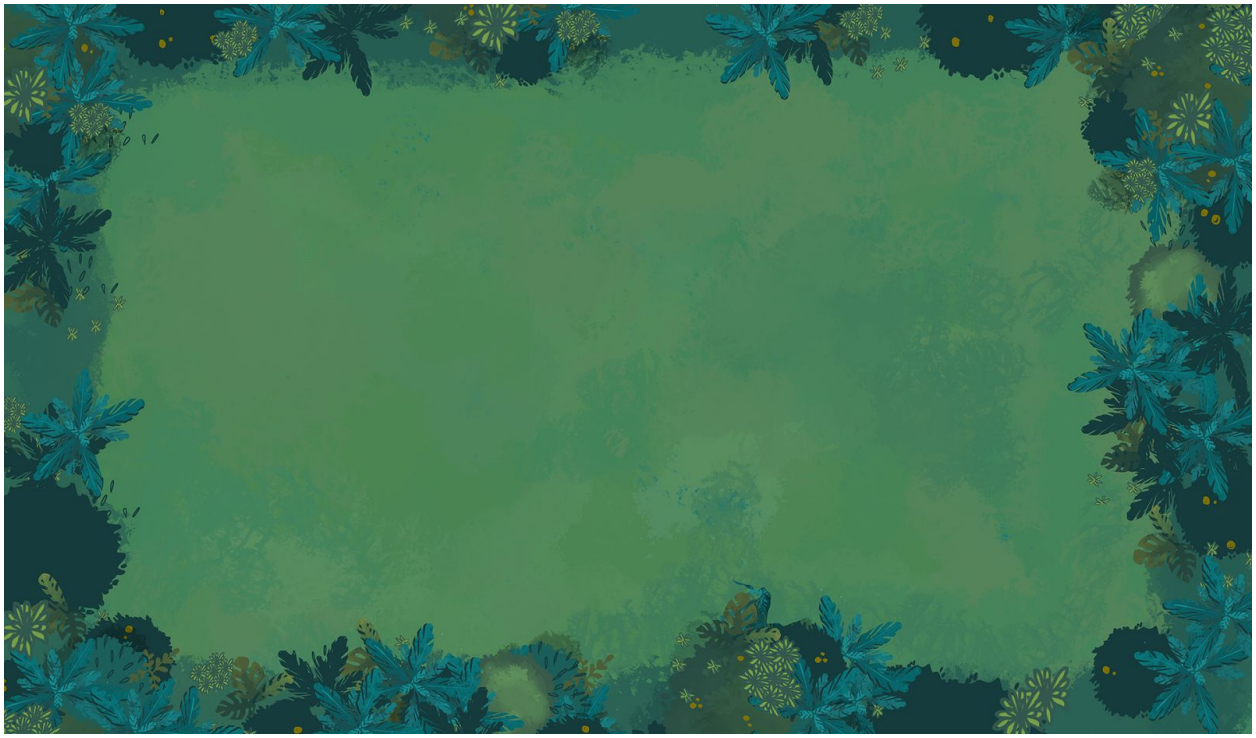
The prototype consists of the different entities in the game, each of the entities is printed and cut out and can be placed over the background.
Trees have a fixed position on the playfield and farmers and cattle move around. The Farmers are controlled by a player and the cattle is controlled by the Computer. The following graphics show the entities:

Farmer



Trees

Cattle



Setup

One person is the computer and has to handle the behaviour of the animals respectively is the referee. The others each can steer the persons. We used kind of a real-time approach with breaks to settle/resolve interesting situation and tested 2 different scenarios. The first scenario was with 4 barns where each player has its own as seen on the picture. The second scenario was with a single barn in the middle of the playground, where all players “deliver” the cattle.

Experience

The overall gameplay was fun, we found that especially the idea of “stealing” somebody’s cattle while he is attracting it is sparking competition between us. However, we were not able to really simulate the dynamic behaviour of the cattle and behaviours like flocking.

Scenario 1: With the different barns were as expected, however, we felt that it will be probably a good idea to have some kind of direct player to player interaction - for example - the possibility to hit someone and stun to allow to gain influence over the cattle.

Scenario 2: In scenario 2, we quickly noticed that a viable strategy will camping around the single barn and only try to steal animals from the other players there - without really doing anything else. This is not really what we wanted.

Lessons Learned

- Mark animals that react to the player to give feedback on who is influencing them the most.
- Single barn will inevitably lead to camping in front of the barn
    - This means that we will probably stick with 4 barns for now.
- Maybe, some powerups/skills (such as a carrot that is used to “pull” animals) have to be collected first to make things more interesting
    - Idea: Carrot may be eaten if animals get too close
    - Probably, we need to have player interaction aka hit to stun somebody
- As the dynamics really matter, the “static” prototype did not really help us in judging the final result. However, it still was quite helpful, as it forced us to discuss, reevaluate things and to come up with new ideas.

Interim Report – Team 6

The current working title is “Flockamole!”.

Progress

The team made steady progress over the last few weeks and groundwork for all features for all layers are essentially there. We have completely implemented an “engine” using an entity-components approach which in the end, turned out quite like what Unity uses. Gameplay-wise, we have started almost all features of all levels – we have completed layer 1 and most of layer 2. As the “engine-part” of the game is ready to support the remaining features, we should be able to advance quickly in the next week(s).

The following list shows the current status as of 23.04.2017:

Functional Minimum

- Basic Map — Done
- Controllable players — Done
- One type of animal — Done
- Core gameplay
  - Pushing — Done (Needs AI polishing for higher targets)
  - Pulling — Done (Needs AI polishing for higher targets)

Low target

- Scoring and rounds: player with most animals in his barn wins — Done
- *A second type of animal* — *In Progress: Only Graphics and supporting modular code*
- *Replace placeholders with real art assets* — *Mostly Done (a few art assets are still considered draft)*
- Make map fun – fine-tune first gamplay — Done (Needs more tuning for higher targets)

Desirable target

- *Make AI of each animal type feel unique* — *In Progress: modular code supporting it is done*
- *One animal type with a flocking AI* — *In Progress: experimenting with flocking behaviors*
- *Add sound effects:* — *In Progress: fully implemented but not yet used*
- Add background music — Done

High target

- *Add a second map* — *In Progress: different levels available for tuning*
- *Animated characters and animals everywhere* — *Sprite-Animations are supported but not used*
- *Animated scores and matching sound effects* — *Not started*

Screenshots

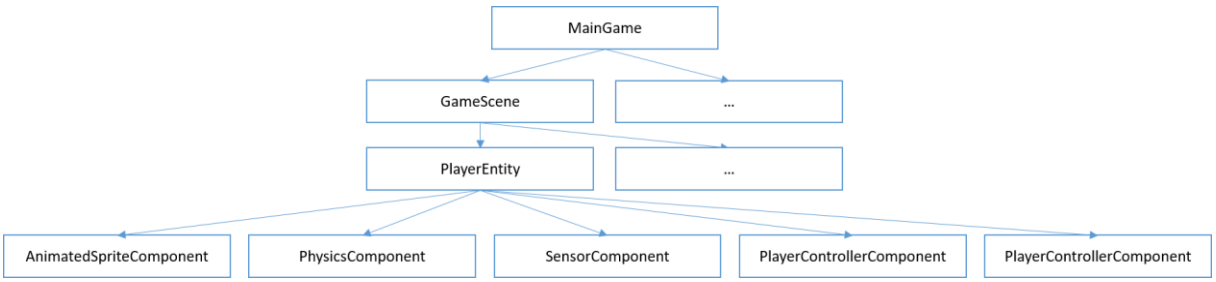
A few screenshots to show the current state of the game.

Title/Join Screen is just text based at the moment:

Debug view in game: Showing collision shapes (orange), sensor shapes(white) and a layout grid. This screenshot also shows an experimental game mode.

Software Design/Architecture

Our game is divided into an engine and game part. A game is composed out of multiple scenes, all of which inherit from the engine's base class engine that provides basic functionality such as entity management. Each entity consists of multiple components. Some components are readily available from the engine - such as a physics component for interaction and collision with the world – and can be composed into complete entities. The following diagram shows an example of this architecture:



This structure is quite like unity. We did not use unity as a model but quickly converged to that architecture. As unity is widely used, this is an indicator that our design is ok.

Component Lookup

We have a lookup container that allows for components to implement interfaces and for consumers to easily query them. For examples, components, that need to be updated regularly, can implement IUpdateableComponent and will be then updated automatically in the game loop by the engine. For example, the PlayerControllerComponent implements IUpdateableComponent and IDrawableComponent:

```
public class PlayerControllerComponent : InputComponent, IUpdateableComponent,
IDrawableComponent
```

It then gets automatically updated by this easy query:

```
foreach (var component in components.GetAll<IUpdateableComponent>())
     component.Update(elapsedSeconds, totalSeconds);
```

This is allows for very easy extensibility of our engine and/or game by writing new components or easy introduction of new interfaces such as `IDebugDrawable`, which we introduced to allow any component to draw in the debug pass.

State Based AI

Our technical achievement is the AI. We implemented a composable, modular architecture for state based AI. A state based AI consists of an enumeration of different states and multiple behaviors. Each behavior is associated to a single state but can transition to multiple other states. This allows us to compose an AI from different behaviours: While for example, the behavior for grazing might be the same for different animals, we can plug in a different fleeing or following behavior. For example, multiple chicken might follow a single player while flocking around him while an aggressive boar might decide to not get too close to another boar that is already following a player. Note that the AI is still very much subject to tinkering and not yet finalized in any way.

Alpha Release Report – Team 6

The current working title is “Flockamole!”.

Progress

Since the interim release, we have made steady progress towards a more finished game. For the alpha release, we added the following:

- Dash & stun other players
- Second type of animal
- New join screen
- Unique behavior of each animal type
- Collision avoidance behavior of animals
- One animal type with a flocking AI
- More sounds
- More animations, animated animals
- Transition animation between scenes
- Right to left map is more polished now
- Pause screen

This completes the low target, almost finishes the desirable target and the high target. The following list shows the current status as of 08.05.2017:

Functional Minimum

- Basic Map — Done
- Controllable players — Done
- One type of animal — Done
- Core gameplay
  - Pushing — Done (Needs AI polishing for higher targets)
  - Pulling — Done (Needs AI polishing for higher targets)

Low target

- Scoring and rounds: player with most animals in his barn wins — Done
- A second type of animal — Done
- Replace placeholders with real art assets — Done
- Make map fun – fine-tune first gamplay — Done (Needs more tuning for higher targets)

Desirable target

- Make AI of each animal type feel unique — Done
- One animal type with a flocking AI — Done
- *Add sound effects:* — *In Progress: fully implemented but not yet widely used*
- Add background music — Done

High target

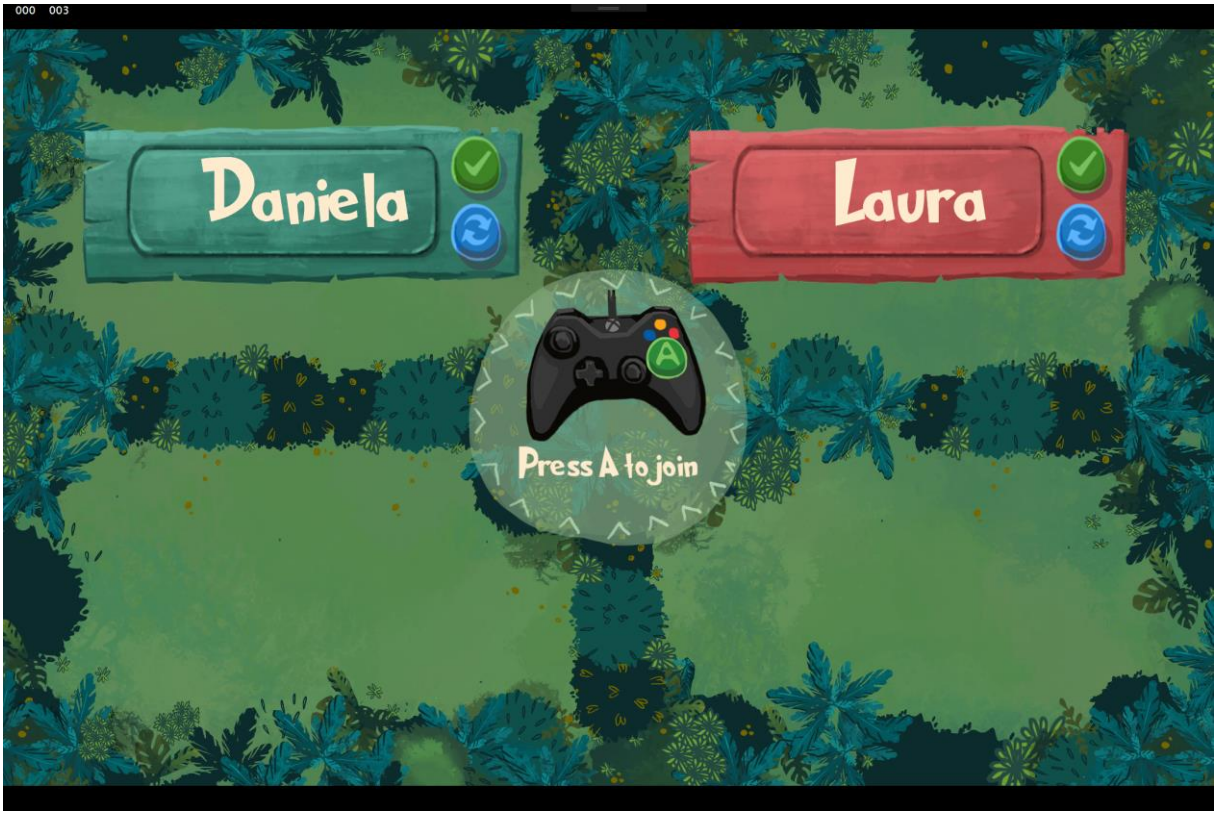
- Add a second map — Done

- Animated characters and animals Done
- *Animated scores and matching sound effects* *Not started*

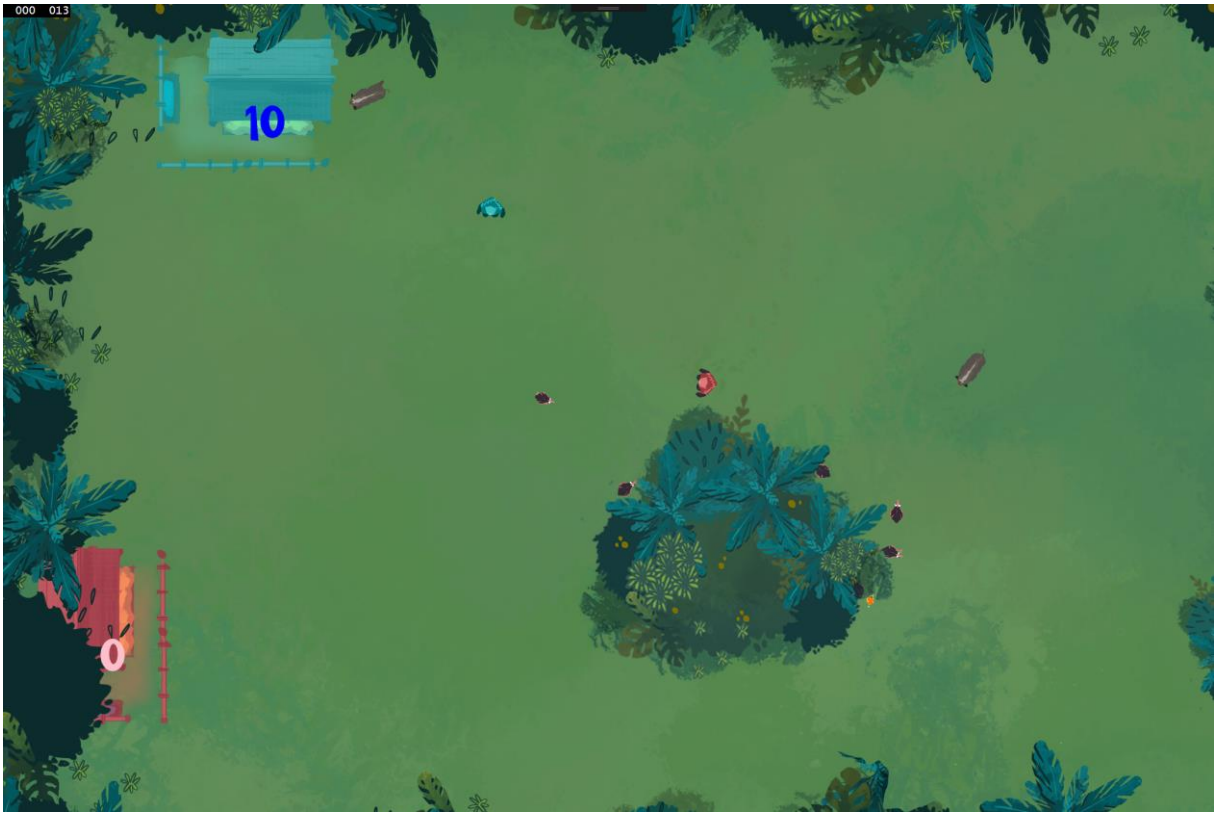
Screenshots

A few screenshots to show some of the new features and improvements.

New join screen:



Second type of animal and more polished right to left mode:



Pause menu:

Transition:

Playtest Report – Team 6

Early-on playtesting

During the last weeks, we handed the game to several people to playtest. This way we could directly adapt their critique into the design and development of the game. When implementing and designing any software, one already knows the tricks of using it. With testing in parallel of development we could find problems with the game early on.

These are some of the critique we got early on and could mostly incorporate into the game:

- Shouting should start the fleeing process of the animal. The animal should continue running for a while, after the shouting stops.
- The Animals should have a unique behavior, more boar- and chicken-like.
- The game could need a higher pace, maybe with a smaller map
- Missing a more competitive aspect (maybe dashing and stunning)
- An enemy in the game could produce more pressure on the players
- More obstacles like a river

With these remarks, we decided to add an AI component system based on a state machine to modularly combine behaviors to get different animal type. With the state machine, we can let an animal run for some time before it returns in the neutral wandering behavior.
We also reduced the size of the game field a little bit to increase the pace. With the dash action, we introduced more competitiveness between the players. Also, having a game mode where the animals spawn on the right and all the barns are on the left adds to this.

Playtesting week

In the last week, we then did several playtesting sessions, because now the game has all its functionalities and needs to be polished for the final release.

Visuals, Sound and other Feedback of the game to the player

Some of the feedback we got from our test subjects, was that the visuals need some adaption. We should add maybe some more to indicate stuff happening in the game and tune others.

- The animals seem to slide and not walk
- The dizzy stars could be more visible
- Let boars fall on the back when dashed (instead of spinning)
- Maybe mark the animals when influenced by lure

- Indicate if an animal only follows player and cannot be scared
- The radius of the influence of screaming and luring is not clear

Other criticism addressed the level design:

- The players should be placed in front of the barn, so that they are visible at the start of the game
- The positions of the barn are not the same as in the join screen, this can be confusing
- The level of the zoom is good
- Collision shapes of trees are too small, entities can walk under the trees
- Some sounds are missed too:
- Sounds for shout and lure and for animal arriving in barn

Behavior of interaction with the cattle

- Dashing in a boar is strange feeling
- If scare and lure would be designed as an action, they could become more tactical and interesting
- The lure should have more influence on the chickens
- The boar runs to long when scared

User Interface

- At end of round, scores not visible, because always somebody presses A
- In the game mode menu, entry that is being edited could be enlarged

Balancing

- Luring and shouting needs to be more balanced, shouting is too strong
- Is it possible to perma stun a player?
- Balancing of distance of dash to map size

Game-play

The most important feedback about the gameplay we got, was that playing our game is fun! Using the dash is a large improvement in the gameplay, it gives it more interaction. Having two actions (lure and shout) requires some tactics.
Nevertheless, we got some interesting ideas how the game could be even better and how we could add more depth to the gameplay:

- Place an item on the right of the field to give a reason to walk on the other side
  This item could be a power-up that could:
  - increase the size of the barn. This would make it easier for animals to enter
  - be an item that can be placed to lure animals
  - be something to sabotage the fence, letting animals of other players escape

- Introduce a strategy phase, to place bait and draw paths that then can be used in the second phase of the game to bring animals to the barn
- Add an «expert»-mode with holes in the map. Animals falling in the holes let them vanish and players falling in will respawn them in the barn. This way the player has to lure the animals around the holes.

Pictures

Vacamole

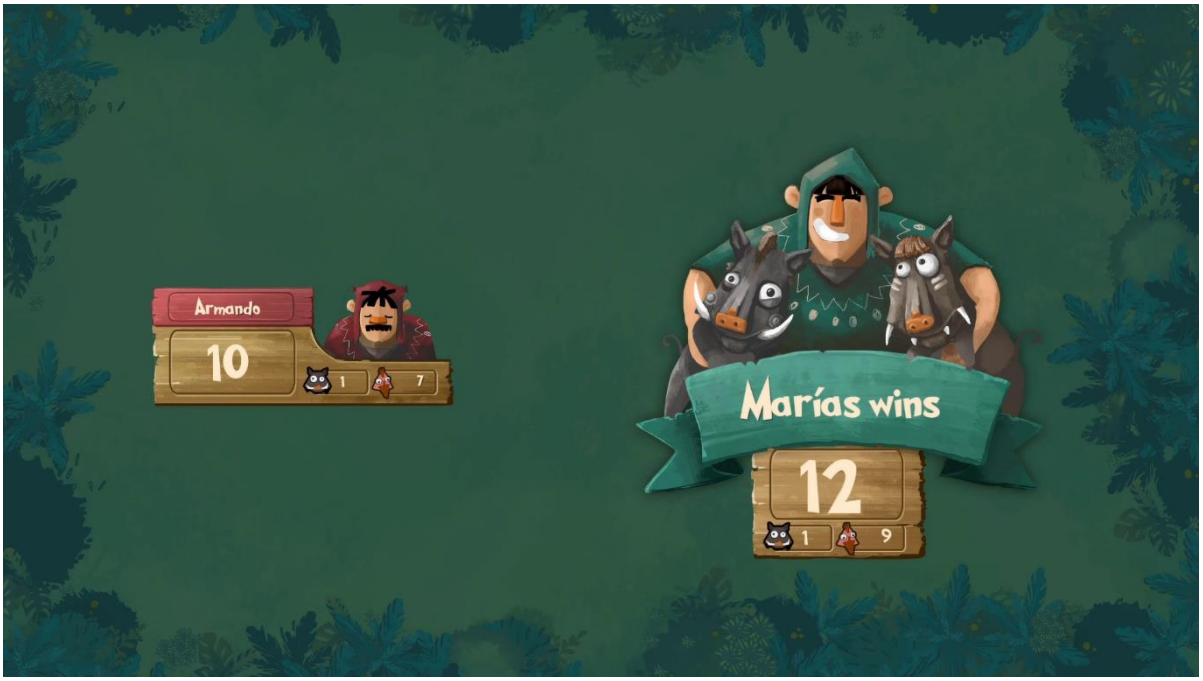
Conclusion – Team 6

Targets – Final Result

We reached all our targets and implemented even more features. The additional features include a second game mode (2vs2), player “combat” with the ability to dash and stun and some more. In contrast to the alpha release, we included all the sound effects of the animals, improved the User Interface and fine-tuned the gameplay completing the remaining targets.

The presentation went reasonably well and we received the jury choice award. Almost everyone that came to our booth had fun – hence, we can say that we are proud about what we achieved during this semester.

The following screenshots show how the game looked at the presentation. It shows some of the new features since the last version: The chickens now flap, a complete winning screen, countdowns for both starting and finishing the game and more.




104


120
0
GO!
+1
2


120
0
2
0

Development Process

We did not follow the development schedule that we drew out at the beginning: As we said, we followed an agile development process with (at least) weekly meeting where we discussed the tasks we wanted to do until next week i.e. the sprint. Each task got a priority: red for essential in the current "sprint", orange for "we should also do that" and yellow for "still somehow important". For managing that, we used Trello where we also tracked all the bugs we found using the same colored task based approach.

The first few game ideas we had were all discarded. But once we settled for our game idea, we followed it relatively closely. The gameplay testing, which we did continuously, did only fine-tune it, the biggest gameplay change was to include the dash and with it, the ability to stun other players.

Personal Impressions

First, we can repeat that we created a fun, local multiplayer – as this was our main goal that we set out to achieve, we can say that we are content about the game. The jury award is a nice topping of course.

- What was the biggest technical difficulty during the project?
    - We had a few issues with the physics engine that we used and had to lookup some source code and fix it. Also, the object pooling and our "garbage-free" enumerators have been a nice challenge. However, as we have been a quite experienced team, we could set realistic goals, both technically and on an deisgn/art perspective. Hence, we did not really encounter any serious issues.
- What was your impression of working with the theme?
    - It did not change much, as "jungle" can be interpreted in a lot of different ways.
- Do you think the theme enhanced your game, or would you have been happier with total freedom?
    - For our game it was fine having the theme.
- What would you do differently in your next game project?

    - We sometimes implemented something in a cheap, quick way and then left it like that, which almost every time backfired and we had to do it properly anyway. But as always, this depends on the time (and money) constraints.

- What was your greatest success during the project?
    - The final result: Other players playing it had fun.
- Are you happy with the final result of your project?
    - Yes
- Do you consider the project a success?
    - Yes
- To what extend did you meet your project plan and milestones (not at all, partly, mostly, always)?
    - We did in an agile fashion, most of the time we achieved all set goals.
- What improvements would you suggest for the course organization?
    - We felt like the presentation where a little bit too much and continues playtesting with the other groups would be both more fun and
- Did you like using MonoGame?
    - Yes and no, we felt that in contrast to using an Engine like Unity, we couldn't spend all the time we want to work on the game. Instead, we spent quite some time to write our own 2d game engine, which turned out to be at least to some degree quite like unity (without the intention to do that).
      However, we feel that it is a good compromise for a game design course at the ETH as MonoGame is only a high-level library abstracting OpenGL and DirectX and hence, still exposes us to technical details.

Outlook

We are motivated to bring the game to completion and iron out the remaining glitches to finally publish the game.