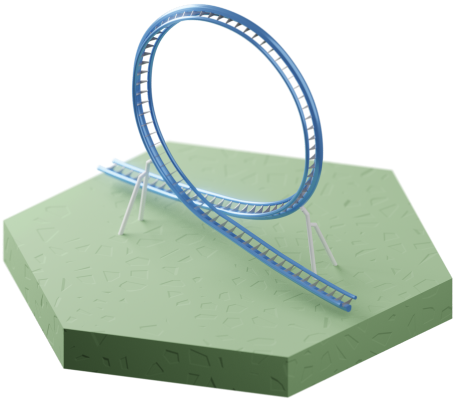
THAT COASTER GAME

- Game Proposal -



Game Idea

THAT Coaster Game is a roller coaster construction game with a specific set of coaster parts to choose from. The game world consists of a generated environment built upon hexagonal tiles. However, only the pieces and their surroundings next to the starter tile or next to already built roller coaster pieces are visible. To build the roller coaster, the player has an infinite stack of coaster pieces, where only the top five or so can be seen. The player can freely choose the next tile they want to place from these visible ones.

Players face 3 major difficulties: Time, how to place parts and environmental influences. Some amount of stress is added to the game as the coaster car continuously follows the tracks, starting to drive when the first roller coaster segment is placed. Thereby, players have to hurry with constructing the roller coaster as the game is lost when the coaster car reaches the end of the tracks and crashes. The game's difficulty is increased by restricting the number of roller coaster parts the player can pick from, which forces the player to strategically place and plan with the remaining coaster parts as well as environmental restrictions in mind. The environment contains buildings or terrain of specific height and wind, restricting the player from placing roller coaster parts of certain heights or forcing them to avoid this tile entirely. The game’s goal is to place as many roller coaster parts as possible, to finish track circuits and to gain points to achieve the highest score before the roller coaster car derails.

Gameplay

Following the course’s theme rather closely, we decided to create a game where the player builds a rollercoaster. Instead of going for a more creativity- or economics-based game such as *Rollercoaster Tycoon* or *Planet Coaster*, we are opting for strategy-based gameplay. We

largely took inspiration from *Dorfromatik*, where the player must build a world similarly made up of tiles, but in a purer strategy-based way. Instead of building a world itself, our game concept is focused on the player creating a roller coaster on top of an already existing world.

The game world consists of hexagonal tiles with different attributes and looks, each placed directly next to each other, creating a flat base plane. The most basic tile will represent a “grassland” tile with no additions that serves as an easy entry into the game and allows free building choices. All the other tiles introduce a height restriction that the player must adapt to: the *house tile* features a set of houses which occupy the ground level (≙ level 0), so any coaster pieces built onto this tile type have to be at least on height level 1. Taking this idea further, the *skyscraper tile* occupies 2 height levels and blocks the player from building on both height levels. The *river* or *lake tile* will only be traversable if the player places a bridge piece which allows them to overcome the body of water. Altering the general height layout of the world, the *mountain tile* changes the height level, but it acts together with other adjacent *mountain tiles* to create a more interesting landscape the player must adapt to, completely blocking the tiles to the player. Finally, a *forest/swampland tile* as part of a nature reserve might also fully restrict build access.

Apart from these environment tiles, the player faces the challenge of continuing the track with the next few available pieces in their stack of coaster parts. An infinite number of parts is available, but only the next top few can be seen and chosen from, so the player has to decide carefully which ones to use in which moment and which ones they wish to save for later. These coaster parts may include the aforementioned *bridge piece*, but also a regular *straight piece*, an *ascending/descending piece* (which changes the height level by 1), a *slight* and *steep curve piece* and lastly less practical ones such as a *looping* or *steep drop*, which might positively affect the players’ score.

The entire game is played from a top-down eagle eye perspective with a simple UI, prominently including the stack of coaster pieces. The player can move around the visible world, rotate their view and click to place coaster parts in places where visual indicators show that the selected part can be placed. At the start of the game, placement is only possible next to the starter piece, the coaster station, after which the player can add more to the existing track. Whenever a new piece is attached, close-by world tiles are revealed. The addition of a new piece depends on the compatibility to the most recent piece in the track and its height as well as the restrictions imposed by the environment. Since a coaster piece has two ends and multiple configurations are possible, the player can rotate it before attaching.

In addition to this strategy system, the player is under a time constraint, since, after some time, the roller coaster car will start moving along the tracks. If the player is too slow and the car falls off the end, the game round is lost. To create a little more randomness to be taken into account, a wind system will affect the track and coaster car: the wind simulation is going to have changing speed and direction and will also depend on the environment. The roller coaster car is going to move according to how much wind resistance it experiences or how much tailwind it gets. Strong wind speeds will restrict the player in building high roller coaster pieces.

Score determination

Apart from the binary of win or loss, the main measure of success in playing will be a score of some kind. Even though we have yet to decide on a specific way of calculating this score, there are three different approaches we are considering. We might implement more than one solution and test to see which fits best and is most enjoyable, ending up in the final game.

Basing the games' score on *Travel Distance* means that, as long as the fail state has not been reached and coasters are moving, the game keeps track of the distance all moving coasters traverse. The score is then calculated based on this distance. This encourages the player to attempt to build fast completed coasters before possibly playing it safe with a longer, slower and therefore easier to build coaster to gain time for the quicker, shorter ones to build up score. However, building a short and quick coaster might, in itself, pose a challenge, since the danger of messing up and losing the game would be greater, suggesting a sufficiently high skill ceiling and keeping the game interesting.

Instead, a possible metric is *Track Length*. Here, the score is simply based on the distance of track the player can place before losing the game. Contrary to travel distance, there is no inherent incentive to aim for faster coasters, since that just results in a higher risk of losing. Similar to coasters aimed to gain time for faster coasters to gather distance, the result would be slower and longer coasters. However, a more relaxed gameplay of avoiding coaster track collision with the environment and other tracks would be closer to the original inspiration, Dorfromantik.

Finally, a *Currency* could be added and act as a score. Starting with a certain amount, the goal would be to build a coaster, spending money either on each tile or to rarely replace a random tile with a preferable one. This could also restrict the player more by adding a new loss condition of running out of money to build track. Currency would be gained as a bonus when completing a track, ensuring the next one can be started properly, and when a completed coaster finishes a lap on its track. The amounts gained would be dependent on qualitative aspects of the completed track, like length or coaster speed. Depending on the implementation, various different playstyles could be encouraged, making it a very flexible approach, but also the most complicated to implement and succeed with.

Design

The design of our game aims to match the game idea and intensity of the gameplay. The rather slow paced gameplay with the main focus set on strategy emphasizes a design that is "uncomplicated and relaxing". The world should not depend on realism and detail, but take players into a playful and dreamy world. This we hope to achieve by taking inspiration from the game "Dorfromantik", which presents its playful environment on a hexagonal tile-based world just like our game world.

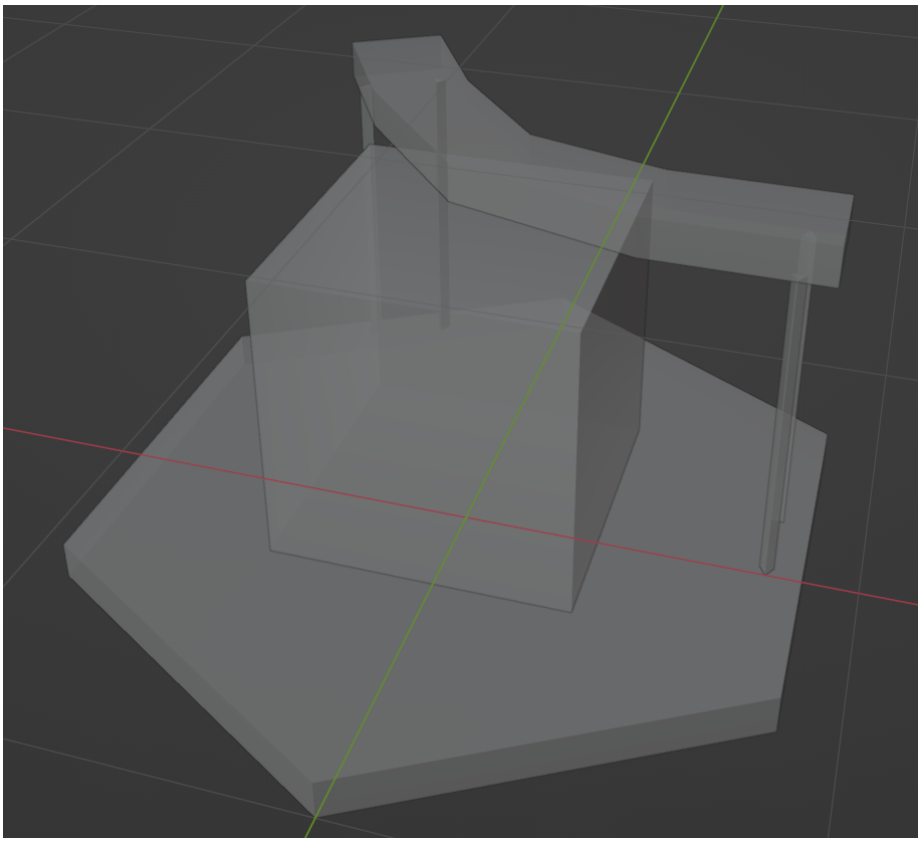
Image 1: Design inspiration
"Dorfromantik" game from https://toukana.com/dorfromantik/

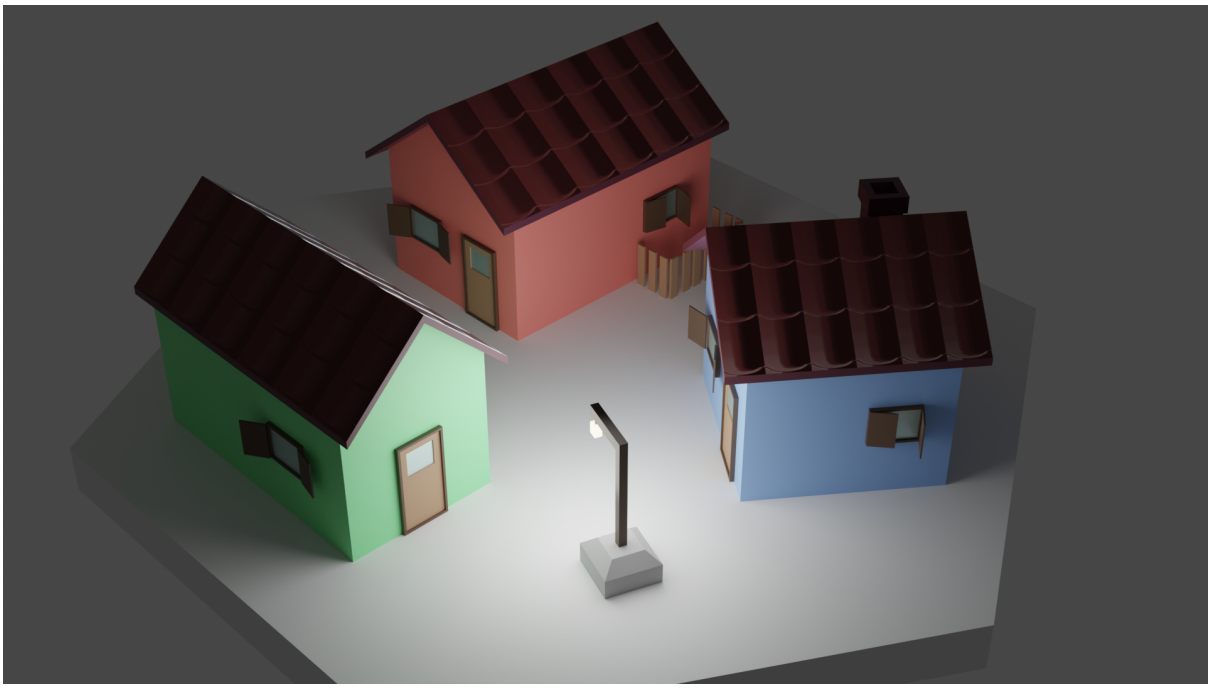
However, its sprite-based drawn looking environment does not match our vision. In THAT Coaster Game we try to fill the world, different to "Dorfromantik", with volume by using full 3D models, letting players perceive the world more immersively. Furthermore, the drawn look is replaced by a toon shader to achieve a cartoony look by use of outlines and adjustments in material behavior.

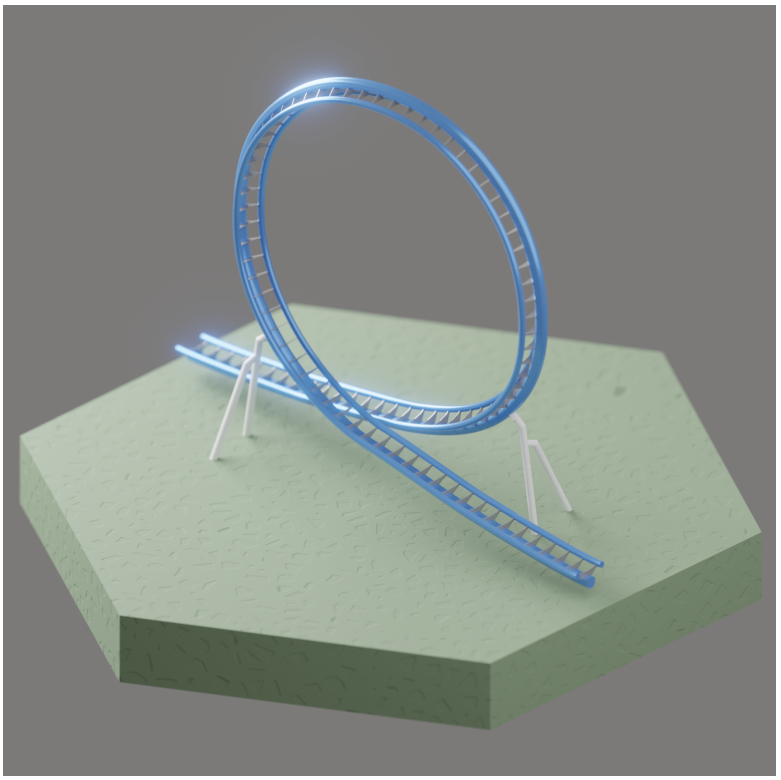
To emphasize the gameplay mechanic of restrictive environment tiles, it is necessary to add the corresponding visuals linking game behavior and design. This is done by creating tiles with things encountered in everyday life like houses, mountains, simple grassland, forest or even lakes. All these different tiles have differing height properties, forcing players to react, like houses restricting the players use of the lowest "ground" level, or mountains forcing them to build very high to get above even them.

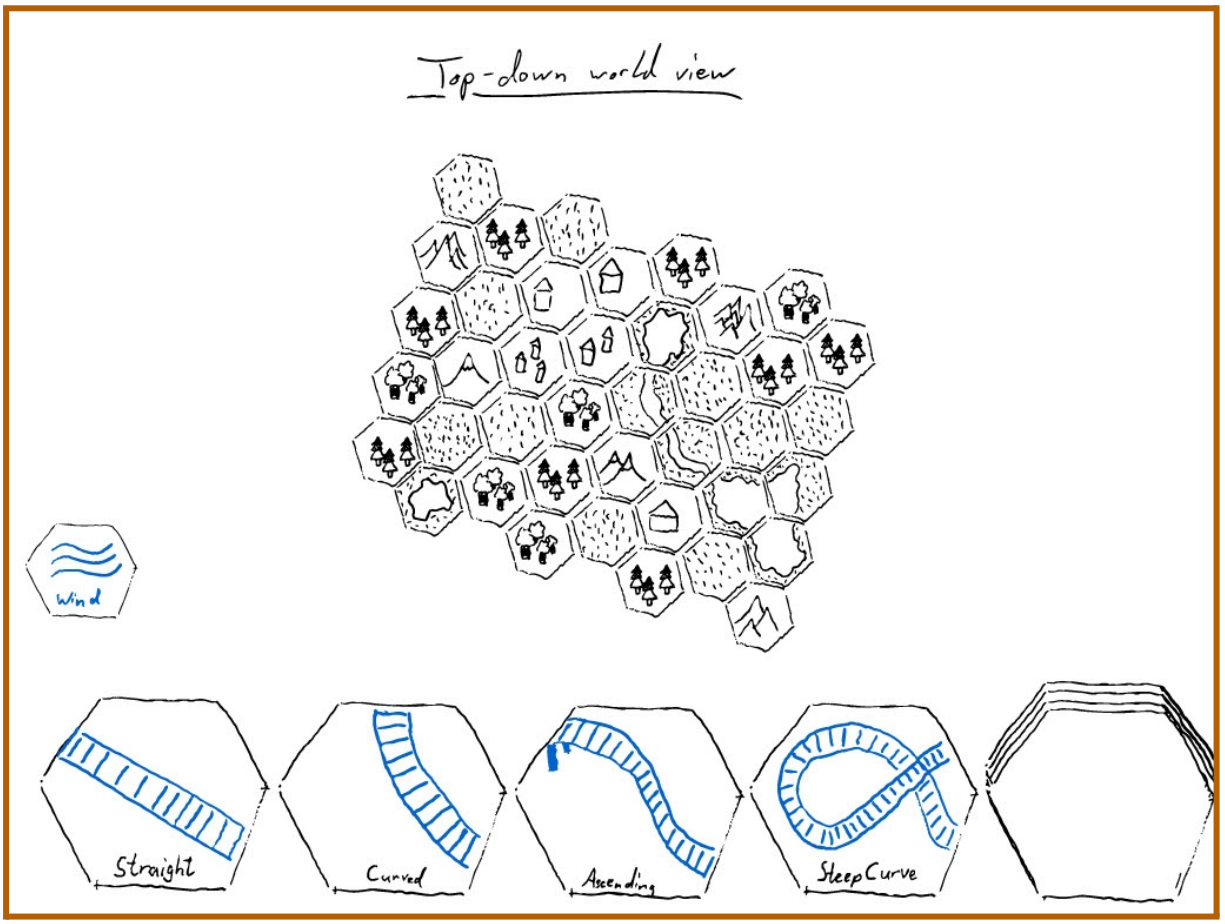
To achieve the same clear experience for the restrictions imposed by the wind system, there are two approaches we are planning to implement in order. First, to achieve the minimum of readability, there will be a UI overlay which can be enabled at any time. There, user interface elements on each tile clearly indicate which tiles can be built upon and which cannot. However, to show the direction and strength of the local wind even outside of this mode, we plan to later implement stream lines, faintly visible in white at any time, who's thickness or amount shows the strength of the wind. The previously mentioned UI overlay could still be utilized to increase readability, but it will be improved by adding color coding to the stream line wind strengths.

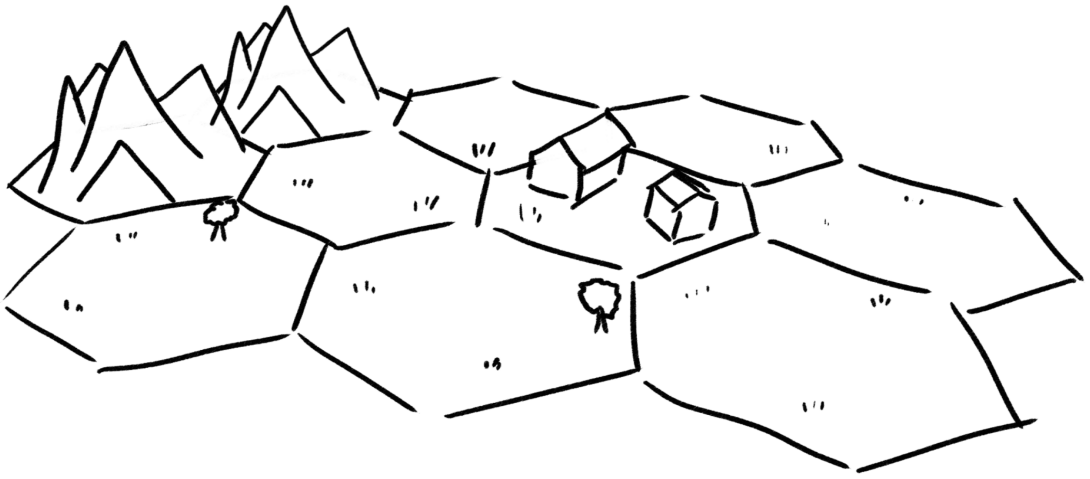
Through these two design categories, players should always be able to visually understand where specific parts can be built. Lastly, the UI is kept rather simple, only showing the roller coaster segments the player can decide to place. Those are represented by tiny floating versions and short descriptive names, gathered at the lower part of the screen.

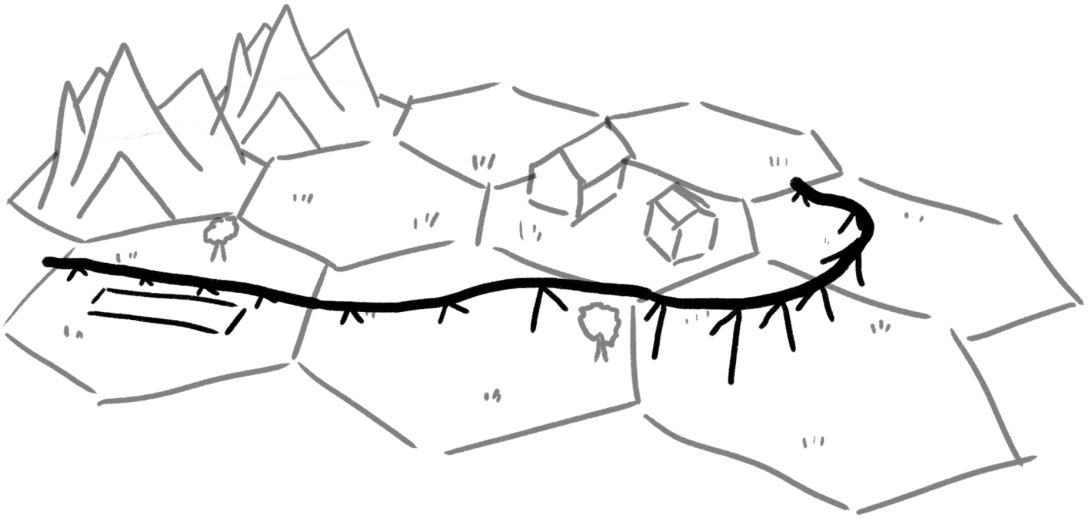


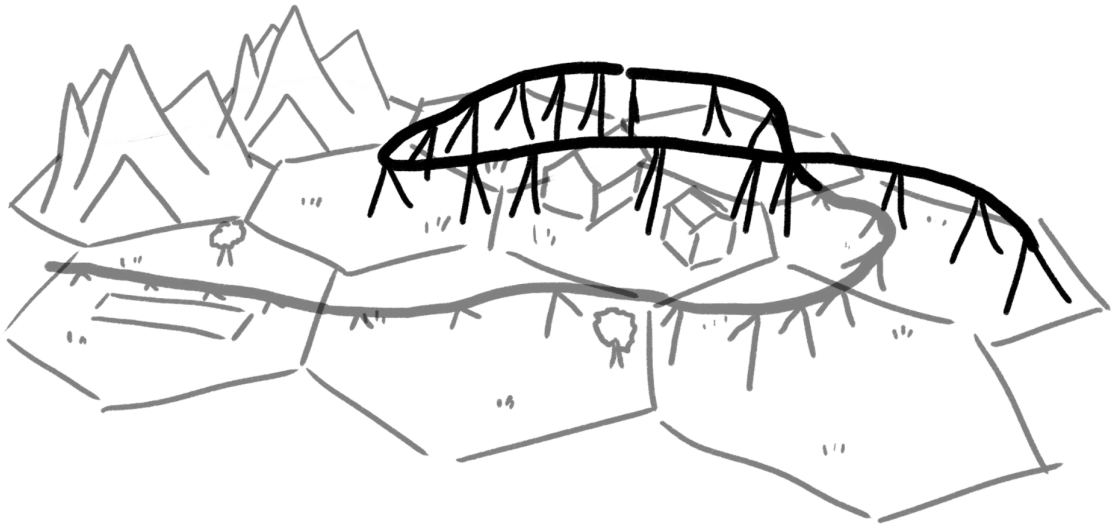





Top-down world view
Wind
Straight
Curved
Ascending
SteepCurve



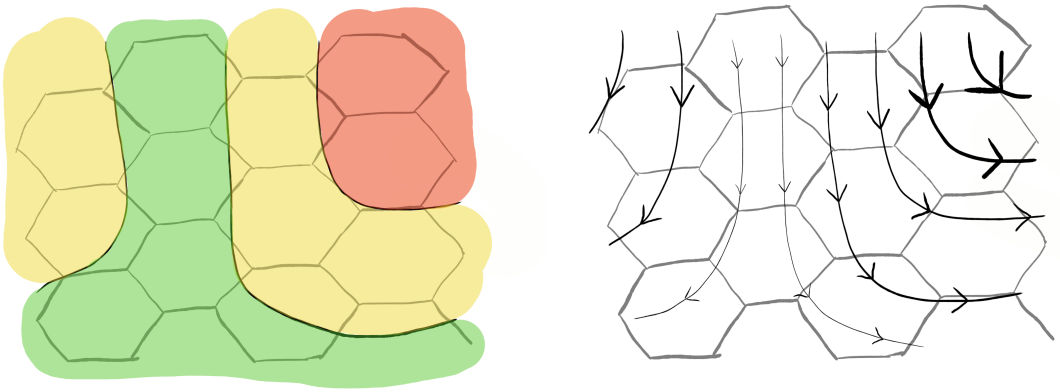




Technical Achievements

Two major technical achievements are found in our game. First, we aim to implement a randomized world generator, as every new start of the game should be a little different. The hexagonal, tile-based environment is generated automatically at the start of every game. The difficulty of this technology stems from the interplay of adjacent tiles, both as models but also during the world generation itself. For instance, mountain ranges, lakes or rivers can not be generated with only one tile, so they have to be placed in a certain way that enables generation of combined environments.

Secondly, a realistic wind simulation will be added, depending on the generated environment and possibly even the placed roller coaster parts. It then provides wind direction and strength per tile, which are attributes used graphically, like in the swaying of foliage or waves of water, as well as for the game logic, changing the roller coaster's speed or restricting the ability to build at high wind speeds.

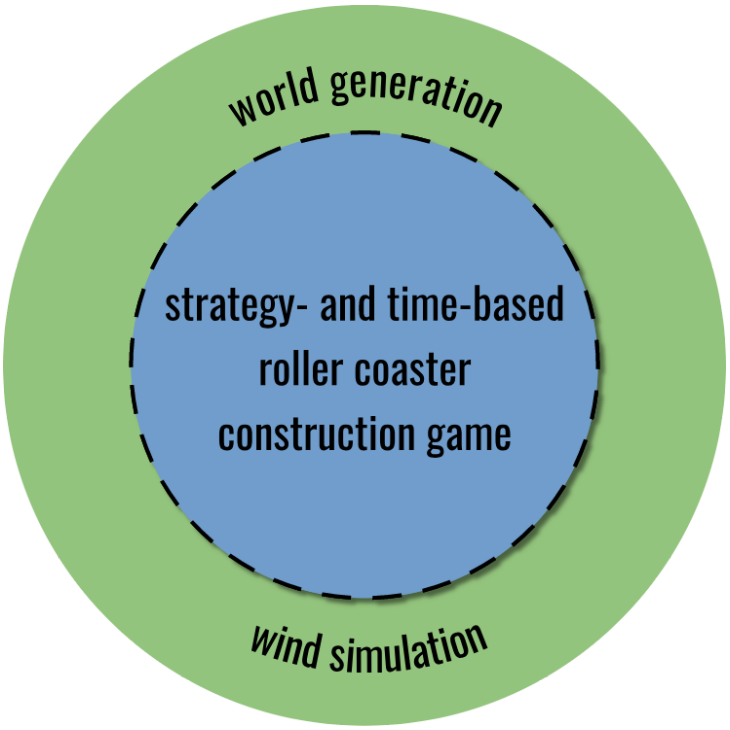


Visualization of the wind by isolines (left) and streamlines (right)

“Big Idea” Bullseye

The *core “bullseye” gameplay idea* is a construction game, where the player can place predefined roller coaster pieces. As the game progresses, it gets harder for the player to place this track, both due to increased time pressure of the coaster approaching the end of the currently placed track as well as spatial restrictions stopping the placement of track pieces on certain tiles.

We implement two *technical innovations* surrounding this core idea: procedural, tile-based world generation and a wind simulation. The world generation creates a novel world for each attempt, refreshing the game each round and restricting the player especially in lower coaster track height levels. The wind simulation provides continuous change in an otherwise largely static world, adds or relieves pressure on the player by affecting the roller coaster speed and restricts the player especially in higher coaster track height levels.



Development Schedule

Layered Development Description

General

This category contains tasks which are recurring throughout the entire development period. This holds true for three tasks: We plan to have about two meetings per week, which we will all attend. We will also all work on the Project Notebook. The third task is the project management done by Helga.

Functional Minimum

As a functional minimum we imagine a simple railway construction game. Tim will do everything concerning the creation of the environment, Helga will do a static user interface, Alex will make the roller coaster move on the rails and the game logic, and Tobias will model the railway pieces and make them puttable into the environment.

Low Target

Our low target is a roller coaster construction game with two height levels, procedural world generation, and a simple wind simulation. Tim will take care of the new environment tiles and the soundtrack. He will also help with the world generation. Helga will enhance the user interface to be dynamic and make a simple wind simulation. Alex will work on the world generation and the camera. Tobias will model some new rail pieces and make the rail pieces rotatable.

Desired Target

For our desired target Tim will model more environment tiles, which allow for structures across multiple tiles. Helga will enhance the wind simulation and visualization, Alex will add shaders, and Tobias will add more rail tiles for another height level.

High Target

If all goes extremely well, we will add a fourth height level. Alex will add a Fog of War and Helga will extend the wind simulation to a weather simulation.

Extras

As an extra we would make a Halloween DLC.

Timeline

Our setup phase in the beginning lasted from the Kick-off Meeting until the end of April. In the middle of April we also started with the development of our functional minimum. The switch from functional minimum to low target is planned for the second half of May. We then intend to pass over to the desired target phase during the first half of June. If we finish this before the end of June, we will then move on to our high target.

		Tasks	Who?	Estimated Hrs	Actual Hrs
2	General	Meetings	Everyone	20 per person	11 per person
3		Project Notebook	Everyone	20 per person	3 per person
4		QA & Debugging	Everyone	MAX	
5		Project Management	Helga	8	3
6	Functional Minimum	Project Setup	Tim	2	2
7		Environment (Houses, Grass)	Tim	5	8
8		Tilemap generation	Tim	10	
9		World Building	Tim	6	
10		Free rail piece selection (Simple UI)	Helga	5	
11		Coaster Movement along splines	Alex	6	3
12		Game logic (fail state, score tracking)	Alex	6	
13		Coaster collision func. min. Tile based	Tobias	8	
14		Coaster (straight, curve, steep curve, start) height 0	Tobias	10	
15		Roller coaster part placement (+ finish coaster)	Tobias	5	
16	Low Target	Environment (Forest, Mountain)	Tim	8	
17		Soundtrack	Tim	3	
18		Procedural World Generation	Tim, Alex	12	
19		Constrained rail piece selection (Dynamic UI)	Helga	5	
20		Static wind simulation	Helga	20	
21		Basic tile-based wind visualization (isolines)	Helga	5	
22		Camera	Alex	6	
23		Coaster tiles (ascending, descending) height 1	Tobias	5	
24		Roller coaster part rotation	Tobias	2	
25	Desired Target	Environment (lake, river, mountain range)	Tim	8	
26		Environment tiles Variants	Tim	12	
27		Sound effects	Tim	3	
28		World Generation (Combined tiles)	Tim	10	
29		Dynamic wind simulation	Helga	20	
30		Advanced wind visualization (streamlines)	Helga	5	
31		Multiple coaster tracks	Alex	4	
32		Shaders (Toon, foliage, water)	Alex	10	
33		Coaster (looping, steep drop) height 2	Tobias	4	
34	High Target	Environment (Skyscraper)	Tim	4	
35		Weather simulation	Helga	15	
36		Fog/Clouds of war	Alex	6	
37		Coaster height 3	Tobias	5	

Assessment

This game has two main strengths. First, the atmosphere, set by the toon look and melodic sound, focuses on calming and relieving players, creating a relaxing environment to be enjoyed after a stressful day. Further, the wind simulation, in combination with random environment generation at each game start, increases the game's replayability, providing a different experience each playthrough. To motivate the player to even start a new playthrough, a score system encourages experimentation. While there isn't one single aspect of the idea that could be considered the most important, players building their own roller coasters in a procedurally generated world with coaster cars following along on the already built track should lead to enjoyment already, simply by enjoying the world and watching coaster cars.

Having this in mind, the target group can be described as rather broad without age restrictions, neither specifically only for children nor seniors. However, it is a purely single player game, which means it would appeal less to fans of cooperative gaming experiences. Furthermore, the game is not intended to be mechanically challenging, broadening the potential target audience to groups not possessing such skills, but also limiting its potential appeal to ones that do and wish to employ it, like players of competitive first person games among others. Nevertheless, even to those groups a relaxing alternative experience might be desirable in between high adrenaline gaming rounds.

We consider our game a success if these strengths can be seen in players both in the short term and in the long term, so a calming effect should be visible and players should wish to replay the game multiple times, both to experience new playthroughs and to aim for new high scores.

Task \ Week	from	10.4.	17.4.	24.4.	1.5.		8.5.	15.5.	22.5.	29.5.	5.6.	12.6.		19.6.	26.6.		3.7.	10.7.		17.7.
	to	16.4.	23.4.	30.4.	7.5.		14.5.	21.5.	28.5.	4.6.	11.6.	18.6.		25.6.	2.7.		9.7.	16.7.		23.7.
Meetings		X	X	X	X		X	X	X	X	X	X		X	X		X	X		
Project Notebook			X	X	X				X	X	X	X		X	X		X	X		
QA & Debugging											X	X		X	X		X	X		
Project Management		X	X	X	X		X	X	X	X	X	X		X	X		X	X		
Project Setup		X																		
Environment (Houses, Grass)			X	X																
Tilemap generation				X	X															
World Building					X															
Free rail piece selection (Simple UI)							X													
Coaster Movement along splines				X	X		X													
Game logic (fail state, score tracking)							X	X	X											
Coaster collision func. min. Tile based									X	X										
Coaster (straight, curve, steep curve, start) height 0							X	X												
Roller coaster part placement (+ finish coaster)								X	X											
Environment (Forest, Mountain)							X	X												
Soundtrack									X											
Procedural World Generation								X												
Constrained rail piece selection (Dynamic UI)										X										
Static wind simulation								X	X											
Basic tile-based wind visualization (isolines)									X	X										
Camera								X	X	X										
Coaster tiles (ascending, descending) height 1											X	X								
Roller coaster part rotation										X										
Environment (lake, river, mountain range)									X	X										
Environment tiles Variants												X								
Sound effects												X								
World Generation (Combined tiles)										X	X	X								
Dynamic wind simulation											X	X								
Advanced wind visualization (streamlines)												X		X						
Multiple coaster tracks											X	X								
Shaders (Toon, foliage, water)							X		X		X	X		X						
Coaster (looping, steep drop) height 2														X	X					
Environment (Skyscraper)																	X			
Weather simulation															X					
Fog/Clouds of war														X	X		X			
Coaster height 3															X					
Halloween DLC																	X	X		

Game idea pitch

Alpha release

Playtesting

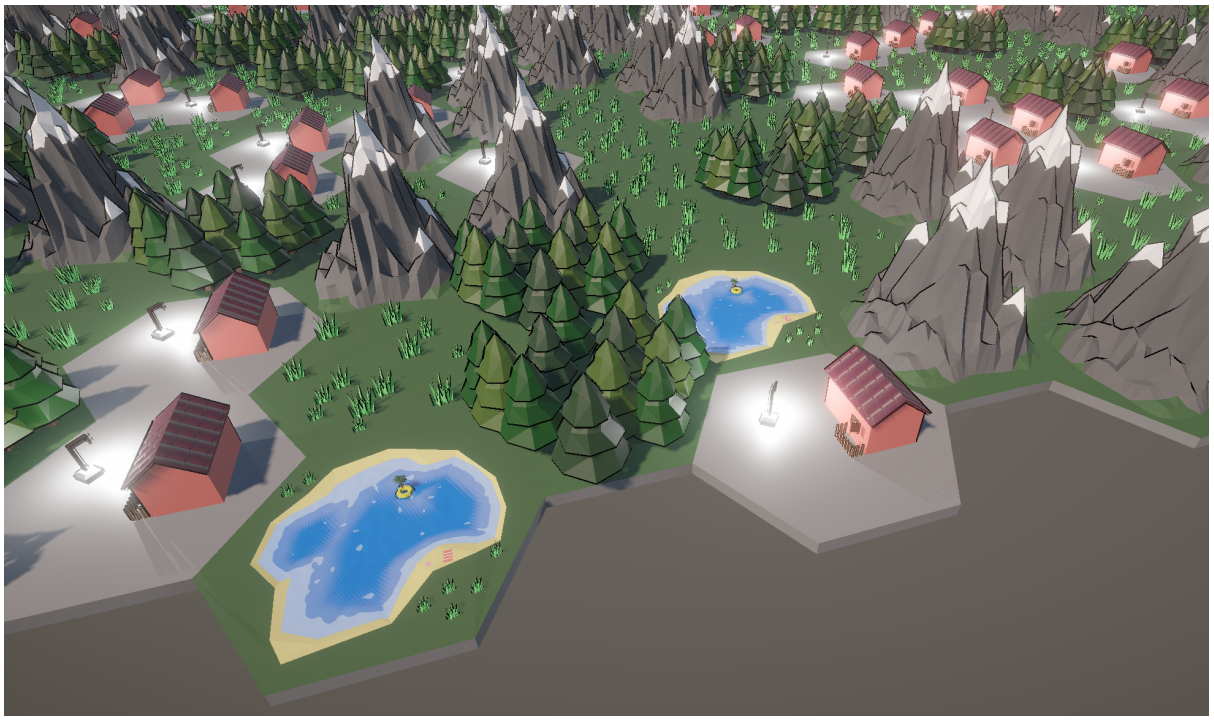
Final release

- Interim Report -

Progress

The progress of our game can easily be explained by looking at our layered development description in combination with our time table and completed tasks. The plan to complete the functional minimum of our game until the first demonstration worked great. We are even ahead of time in some areas. Because of this, we are now able to test the game continuously from a developers perspective until alpha release to find errors or unpleasant design choices before the actual playtests start. Even though we are ahead of time, there is still a lot of work to be done since only a functional minimum is completed.
Right now, the hexagonal world tilemap can be generated with some environments restricting rollercoaster construction. The rollercoaster can be constructed on ground level by free selection of the coaster parts on a simplified first UI. Additionally, the coaster cart starts driving on our spline-based roller coaster tracks and shaders are added to create the desired look. Furthermore, the static wind simulation, one of our technical achievements, is already calculated and visualized.
The actual tasks completed by each member are described in the following.

Tim



The current game world consists of five different environment tiles Mountain, Spruce forest, Grassland, Lake and a House which will be exchanged later. All those Models were modeled by ourselves using Blender. All of them follow our design choice like proposed in the game idea, being low poly to emphasize the playful and cartoony style. The environment tiles

currently have a maximum of 1.500 vertices each, leaving enough room to instantiate lots of them or add some details for future variants. Currently the environment tiles do not use any textures except the grass floor, instead only their base colors and specular/roughness values are adjusted to create the desired material look for different environments. Even when details were added, the tiles still keep the blocky look and by this an decreased vertex amount which is especially visible looking at the house consisting of a very simple cubicle base. In later phases, when performance tests were accomplished, this base might get a few more vertices to enable rounded edges. Such changes are considered to be made later in the project, when no performance issues are found.

The game world consisting of these described tiles is procedurally generated to emphasize replaying the game. It is achieved by using Unity's Tilemap system and generates the specific tiles by following a certain strategy. First, a naive approach was implemented where the environment tiles were placed completely random, but one of the critiques on this task was to probably add difficulty the further away you are from the center. This was taken to heart, in a way that we are now calculating weights for each environment tile at each tilemap index, representing their probability to be built. By using different weight calculation functions for different environments, we have also achieved a second approach, where the probability of Grassland decreases from the center, while probabilities for Forests, Mountains and Houses increase with the distance to the center. Only Lakes can appear anywhere on the map with a constant low probability, resulting in a map with mostly Grassland in the center and an increasingly more difficult terrain at the map boundaries.
After implementing this, we found out that the software architecture might be improved by using some pattern, therefore it was reengineered using MVVM and Visitor pattern, which was a substitute for the real desired type erasure that is unfortunately not possible to implement within C#. However, using MVVM and Visitor pattern expanding this with a strategy pattern for world generation strategies, our software architecture allowed great decoupling and therefore easily maintainable code.

Helga

My tasks so far were the implementation of a simple user interface (UI) for the functional minimum and a static wind simulation with a part of its visualization for the low target. This is round and about also what I achieved, although a few details here and there are still missing.

Creating the simple UI was easier than expected. This was mainly because both Tobi as well as I thought ourselves responsible for the mouse pointer raycasting. Luckily we clarified this to be his task early enough so the work was only done once. However the simple UI is not quite finished yet, because it still needs a button, which toggles the wind visualization on and off.

The static wind simulation was pretty straightforward to set up, but a lot of edge cases are still missing. Especially the behavior at the borders of the map is going to take a lot of time. I also want to refactor the source code so it will be easier to expand for later development. Further next steps are taking the different height levels of the terrain into account and expanding the simulation into three-dimensional space. The connection between the wind system and the game logic is yet to be implemented, too.

The wind visualization was more difficult than I anticipated, because I did not have as much prior knowledge as I thought. As of writing this report I managed to make a basic prototype. While doing that, I realized that the isolines should only be at the borders between tiles. That way it will be clear for the players, where they can place a rail piece and where not. This part of the visualization is still missing. I am also considering to introduce set color stages for the wind speed instead of interpolating between red and green.

In conclusion, I find it difficult to do one task after the other. For example it is easier to debug the wind simulation if there is a wind visualization. But at the same time I need a running wind simulation to have data for the visualization. Beside that I find development about as challenging as expected.

Alex

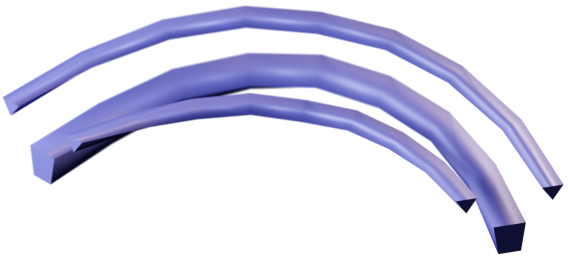
My three primary areas of focus were getting the coasters to move in the game world smoothly across their individual splines which the player places, developing a cartoon shader to give the game a more artistic style and to implement camera movement. The coaster movement proved somewhat complicated, because the spline package used does not offer exactly that functionality, and the preexisting component, which would be perfect to use otherwise, overwrites some values occasionally. To avoid this, the class itself was copied and adapted in a way to involve our required functionality natively. While doing so, two variants of our scoring were added as well: Track placement distance is calculated by adding the new track's distance to the prior accumulated distance when adding a new piece, and coaster travel distance is a complement out of the sum of prior fully traversed track pieces and the current track's length multiplied by how much of it was traversed already.

The toon shader, or its first somewhat naive implementation using shadergraph and multiple render passes defined in the render asset, does give the game a stylistic touch, but is, due to the game's models being flat-shaded, also not perfectly accurate. Corners specifically do not line up perfectly. For this demo, the shader will be used, as it does work reasonably well, especially considering the distance from these objects the camera typically has in our game. However, later it might be replaced by a more accurate alternative.

Furthermore, camera movement was also implemented well, utilizing the newer Unity Input System. Events call methods altering the given input, which is then used in the movement scripts update method. A zoom function was also added, using splines to pan the camera downwards and highlight the finer details in our beautiful models, as well as moving a lookAt target in the same way, ensuring the camera's downward rotation fits for all zoom levels. One issue with the combination of rotation and movement was also mitigated that way: When zoomed out entirely, rotating turns the camera around a virtual focus point on the ground at the position the camera looks at, since that is where the lookAt target is at those settings. When zooming in, however, the lookAt target moves outwards, while the camera moves towards the z coordinate of that rotation point, transitioning the rotation behavior into one around the camera instead.

Lastly, instead of helping Tim with the world generation elements as I had been assigned to, I ended up supporting Tobi with the track placement. We will continue to work together on that in the future too, since we now both know the detailed inner workings of our relevant scripts. However, for details on the specifics and what he achieved before I joined him I shall refer to Tobi's section.

Tobi



For this milestone, I was tasked with the rollercoaster itself. First, I modeled a variety of coaster track pieces, namely a *station*, *straight track*, *slight curve* and *steep curve*. Additionally, I created a rollercoaster cart model. All the track models were created in Blender using a spline, which gets its actual extruded shape from a profile consisting of two triangles and a trapezoid. This and the number of subdivisions keeps the amount of vertices needed for the models to a minimum. By enabling smooth surface rendering, even the triangular track rails can achieve the illusion of a round rail. The spline data was then manually copied into Unity's splines to ensure the cart will precisely follow the shape of the

track models. Apart from the *station track*, the track rails don't yet have support beams as well as connecting crossbeams in between them (mostly to see whether the performance is acceptable with the simpler models), but can simply be added since they can be generated along the spline, which will be done in the next development layer. The biggest challenge in this step was to precisely calculate the positions of the beginning and end of the spline within the hexagon layout, which turned out to be rather simple with some basic trigonometry.

Second, I implemented a GameManager, which takes care of and combines all the individual parts of this game. As of this moment, the player is directly transferred into a generated world with a station track piece spawned in the middle of the map. New track pieces can then be selected from a set of buttons. This behavior is modeled as a *state machine*, where each state – the base state and the placement state – takes care of certain tasks when it is started or stopped. This way, new states, such as a menu, can be easily added in the future, and we avoid a confusing and bloated manager class.
After clicking a track button, a "blueprint" of the track piece is spawned and snaps to the environment tile that is closest to the mouse position. The blueprint appears in two versions: green, if placement is allowed; red if it is not. A piece can only be placed if it is next to an existing roller coaster track, there is no other coaster track obstructing it and the height on the environment tile allows for building the specific selected track. In implementing this logic as well as the ability to rotate pieces, I was aided by Alex. For the rotation, we additionally check the orientation of the currently selected blueprint as well as whether there is a part of an existing roller coaster on a neighboring tile. If this is the case and, depending on the track model, the start- and end-positions of the respective track pieces match, the player can set the new part. This piece is then added to the existing coaster and the coaster carts can immediately ride along it as well.
Piecing together the different modules of our game was quite challenging, but together we managed to understand each other's implementations and how to properly connect them.

- Alpha Release -

Progress

As we reached the next milestone, the Alpha Release, so did our project. A lot of features were added and components put together since the interim report, resulting in there now being a very functional multi-layered game world with great wind simulation, expanded environment generation and various effect improvements like the outline shader used, a day/night cycle and much more. With this being said, we finalized our desired target with some changes made to our time planning. Most of those features were only postponed to later targets due to time issues, like adding further environment variations or automatic track rotation. However, we also completely removed some goals, like a third height level or larger city-like environments, to focus in the upcoming weeks on the knowledge we will gain from the playtests and make precise detailed changes to round up the experience. To ensure that nothing suggested by us or our playtesters is forgotten or lost, we started using git labs' issue-based boards, further increasing our project management abilities.
More details on the progress made in the project will be explained in the following sections.

Tim

With this milestone being reached, we accomplished our desired world generation. By using a wave collapse algorithm, larger forests, villages or mountain ranges are generated by themselves. Currently, the algorithm iterates over the tilemap and decides at each location which of the environment tiles that could be placed, is actually placed. Then the valid environments for the neighbors are updated accordingly. Therefore, all environment pieces now have restrictions in their neighborhood ensuring that larger biomes of the same kind are generated more likely, while also increasing the environment's plausibility. Additionally, the center area is always grassland ensuring a nicely playable start and a correctly placed Rollercoaster station onto a grassland tile. However, the size of the area is determined by a certain radius, which has to be investigated during the playtest as it is currently larger and therefore might make the surrounding environment irrelevant as players can stay in this starting area. This will have to be investigated in the playtesting and by taking our highest scores into account.

Furthermore, adding more landmarks and plausibility to the environment, rivers were added, which are combined tiles, meaning they consist of multiple contiguous tiles. There are River starts, straight river parts and curves to ensure variety in river generation. Currently 3 rivers are always generated having no limits in minimal or maximal size. By this, rivers can only consist of start and end pieces looking like larger rivers or might get very long increasing the plausibility and look of the world.

Lastly, in terms of world generation, having tile variations was postponed to the high target due to time issues and other focus points. Nonetheless, an additional model for the villages was added, containing 3 houses and the known first model was expanded by an apple tree

that adds further detail to the world. By this, especially villages look much more vivid and playful. Even though the effect of this addition also heavily relies on a new random factor added to the world generation. Random rotation of each individual environment piece. By this, even larger parts of the world, consisting of the same environment, have variety at least to a certain degree, but villages with these two types of models and the additional rotation enabled much more playful and beautiful world generations.

Apart from these World generation topics, I added a very simple sound system and a first ambient soundtrack emphasizing the calm atmospheric design of the game. However, changes to the soundtrack will heavily depend on our future playtests.
All audios of our game will be controlled by an audio mixer that was set up in a way that the soundtrack and sound effects can be controlled independently and are easily adjustable through the editor at development as well as by code.
In terms of sound effects, multiple effects were added, which of course might change after the playtest. Sound effects were mostly taken from royalty free sources, but even some of them have been made by ourselves. Sound clips were added for UI clicking, coaster explosion on the fail state, Coaster part placement and a sound for roller coaster tile transitions.

To conclude this part, I will briefly state the difficulties I faced. Most of the time I had to deal with the rivers as their placement in the world created a lot of issues with the plausibility of the world. Our goal was to use as few possible models and rotate them accordingly, for instance the curved river piece whose rotation complicated a lot of the placement. However, all these challenges could be overcome and a great world generation is now being used.

Helga

My tasks for up until now were a dynamic wind simulation with two visualizations. Originally I was also tasked with an advanced User Interface, but given the difficulties I encountered during development of the wind, we reassigned this task to Alex. The current state of my work is a fully functioning wind simulation with two visualizations, one of which was done in collaboration with Alex.
First I completed the basic wind visualization, which I had already started to work on before the Interims Report. I changed the color palette such that it interpolates between green and yellow for low wind speeds and between yellow and red for high speeds. Tiles which block the wind entirely are rendered black. This color scheme allows for a continuous display of the continuous range of wind speeds. That is why I backed away from the idea of using isolines as they would visualize the wind discreetly.
Next I continued my work on the simulation itself. But when I set out to refactor the code, I found setting up the references between the different classes incredibly hard. In the end I spent twice as much time on the refactoring than on writing the original simulation code. Nonetheless I am glad I put in the work, because the much improved code quality did indeed make further development much easier and comfortable. The simulation is now three-dimensional and takes wind-blocking terrain into account. There is also a class which regulates behavior of the system at the borders. Implementing this class would not have been possible before the refactoring. Instead it became much easier to implement than I

expected. This is also very useful for the implementation of the dynamic wind directions. (As of writing this report, they are still in development.)
Between all my tasks mentioned above, I took some time for a close collaboration with Alex: We made the gras bend according to the wind direction and strength at its position. My work was to provide the wind data for Alex's shader as well as fine-tuning some variables for it to look good.
My biggest takeaway from the previous few weeks is to never again program trash quality code to refactor later. I could have saved so much time and nerves if I had written clean code from the beginning.

Alex

While my areas of responsibility prior to the first milestone were varied, this time there were many more separate smaller tasks to be done. Starting with ones I had begun on already, I expanded on scorekeeping by adapting it for multiple coasters and saving the top 10 highest scores on the device disk. This went pretty smoothly as soon as I got xml to work properly. Similarly, the preparations enabling coasters to crash enabled that feature smoothly, including the ability to complete the circuit, preventing that particular coaster from ever crashing. Similarly, after experimenting a bit, the coaster speed properly changes depending on the coaster's current angle, or specifically the difference in y coordinate between the first and second car. However, this is based on guesstimated physical calculations, and as such is not proven to obey energy conservation laws. Sound effects could also easily be integrated into previous code, which however does result in them being very closely intertwined with that code and therefore not too architecturally pleasing.

The second area I worked on, and probably the largest one with the most varied content, are the shaders. Since the last milestone, I implemented an improved outline shader using a full screen render pass and edge detection on the depth buffer. Figuring this out took a bit, since it felt weird just throwing that many nodes out there at the problem in shader graph. Since then, I have rewritten some of it into a custom function, reducing the amount of those simple calculation nodes. We use the Roberts cross operator, resulting in a kernel size of 4, if I were to implement the Sobel operator I would definitely write a second custom function for the input part of the shader. Apart from that though, there is a second major family of shaders in our project, that being foliage shaders. There are two versions, one for grass and one for trees. First, they introduce a slight swaying motion, depending on the wind speed and direction at that position. To achieve this wind integration the wind system creates and updates a texture each frame containing the wind info of each tile in three channels, the last of which including the strength with the first two forming the direction vector, which is then read by the shader. Next, there is a minor fake ambient occlusion effect, darkening the lower parts of foliage, and finally, the trees have a dynamic color randomization picked from a preselected color space. Not only does that ensure variation, it can also trivially be adapted to different styles, like for example an autumn one for a Halloween DLC.

As a last major general category I worked on the user interface. First, I implemented a currently still somewhat functional two dimensional menu system, including various features like a display for that top 10 high score list and a settings menu with sound volume sliders. I

started off that section with a very sleek function handling practically all button calls. However, as I implemented more features, this function got more and more convoluted and is now a bit too complex for its own good. The UI defining the main user experience of placing blocks was also done by me, taking over some work previously assigned to Helga to give her time to focus on the wind system. Here, I simply used invisible buttons to handle the user interaction, while using 3d models in the background for visualization.

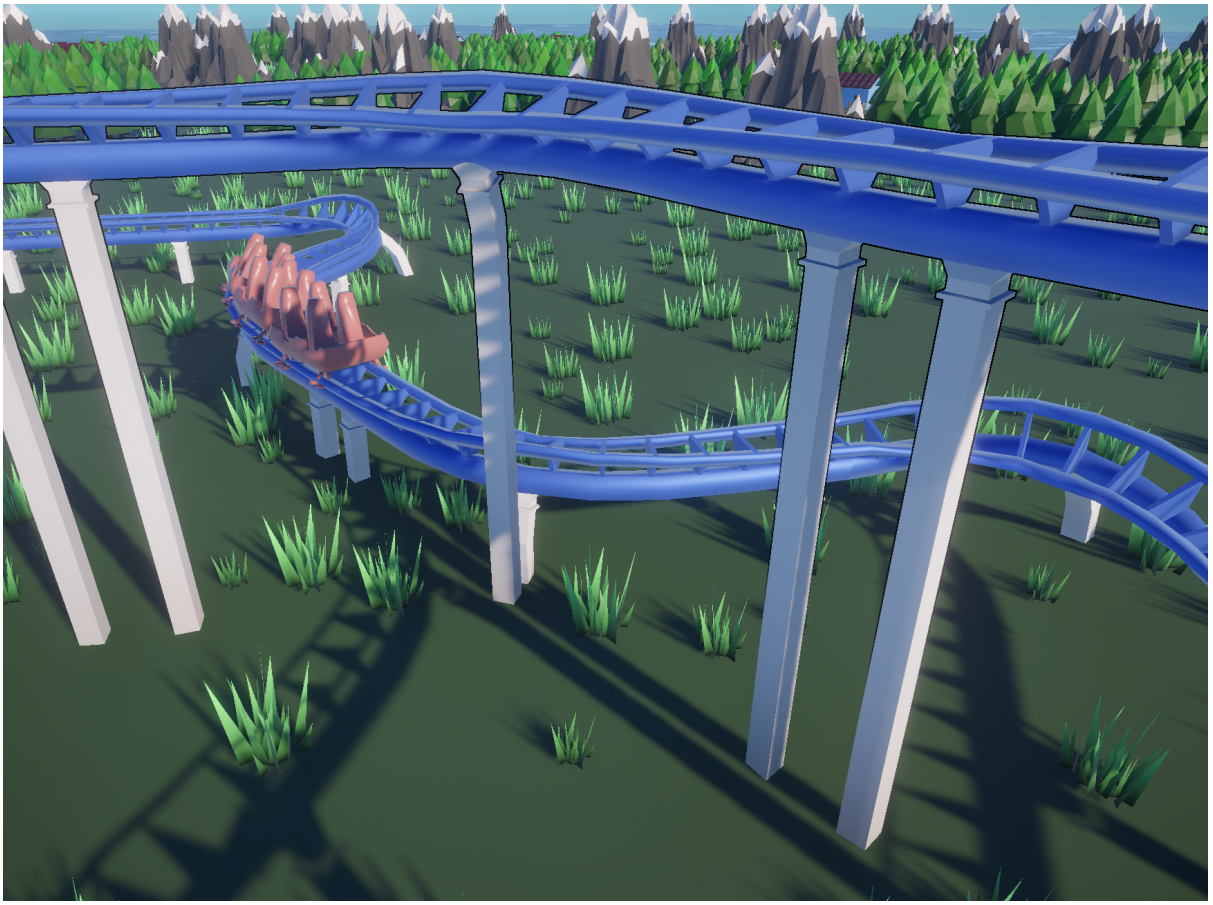
Other than that, there were also smaller features I implemented. This includes the creation of a Unity procedural skybox and using that to show a neat day-night cycle, which unfortunately gets quite green very easily and has very finicky color selection, as well as importing a text font.

Tobi

After completing the placement logic for the rollercoaster on ground level, my main task was to implement the game logic to now also support varying heights. First, I created a slope piece that actually enables the player to extend the coaster to another level. This piece in particular caused many changes in the previous code, since whereas before track pieces only had to check for neighbors in the same level, the slope piece now has to check for neighbors in its own level as well as the next higher level. Similarly, each piece has to check whether it is placed adjacent to a preexisting slope in the level below itself. The different heights in general are implemented by placing track pieces into the tilemap with a z-coordinate offset, while the actual models themselves also are placed into the scene with a height offset corresponding to the height difference of the slope. In order to change the height of the currently selected piece, the player can press the *F* key.
Building at different heights also entails that it is now possible to cross over other rollercoaster tracks as well as certain environment tiles such as the tree- and house-tiles.

The appearance of the coaster tracks was improved by adding cross-support beams between the track rails for each track piece. Furthermore, ground support beams were added for all the models. These also exist in varying heights to support the new height level feature.

In addition to the new height levels, we also added a button so a new station piece can be placed. This evidently also caused major changes in the game logic, and the game has to keep track of where the most recent piece of each rollercoaster was placed.



- Playtesting -

Methods

Our eleven testers were diverse in gender and their previous experiences with games. They also had various tastes in regards to which kind of games (board games and video games) they like to play. However most of them were rather young and all of them had a high level of education.
We conducted our playtesting sessions individually either in person or online with the tester sharing their screen. Afterwards we used a combination of a questionnaire and a structured interview to retrieve information on our game.
Since we wanted to connect the data from the questionnaire with those from the interview, we also assigned IDs. In order to ensure anonymity anyway, we split our group into two people who conducted the tests (Tim and Tobi) and two people who evaluated them (Alex and Helga).
All of the playtesting took place in German, because we only had German testers. Thus the questions in the following are only translations of the actual questions.

The questionnaire asked:

Demographics

- Age
- Gender
- Level of Education

Game Experience

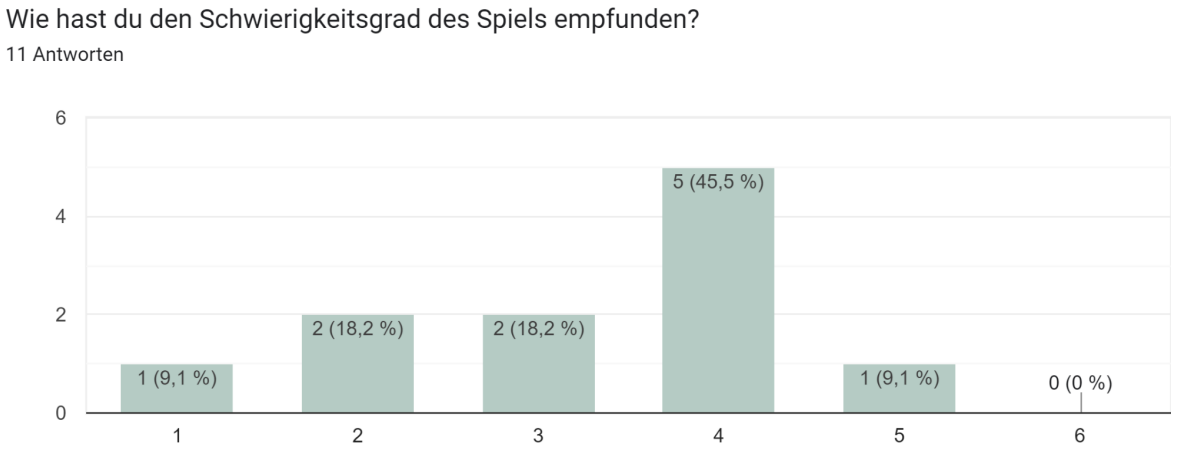
- How did you perceive the difficulty of the game?
- How would you like to change the difficulty level? (with center option: no change)
- How did you experience the controls?
- Could you perceive everything on the screen well?
- If you could not perceive everything well, what and why?
- Do you want to continue playing?
- Why (not)?
- Were you motivated by the highscores?
- Is the design consistent?
- If not, why?
- Did you feel anything lacking from the game?
- If you could change one thing, what would it be?
- Do you want to tell us anything else?

During the interview we asked:

- Do you have prior experience with games?
- What games, if any, did/do you play?
- Does our game fit your usual tastes in regard to games?
- How did you like the game and how did you feel playing it?
- What was your goal in the game?
- What would you change in the controls?
- What did you not understand from the game mechanics?

Results

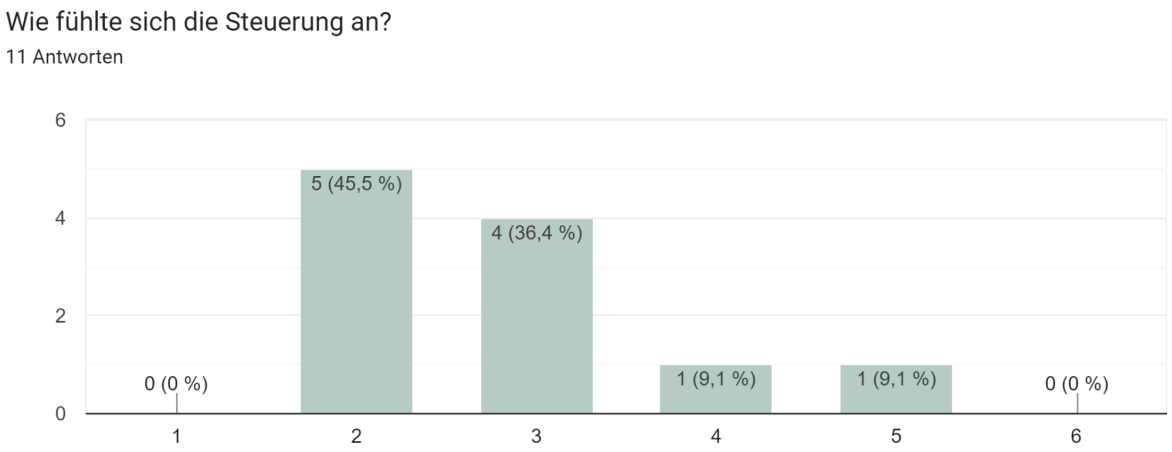
Difficulty



"How did you perceive the Difficulty of the game?", 1: Very Easy, 6: Very Hard

Most players perceived the difficulty to be higher than what was originally intended by us, as our original goal was to create a relaxing game with a perceived difficulty between 2 and 3 out of a 6 point scale. However, all players still enjoyed this level of difficulty, with 90% in favor of keeping it like it is and only one player hoping for only a slight adjustment to make it easier.
The main factor of difficulty in our game is the speed of coasters. The factors resulting in this speed, however, were unclear to 40% of players. Lack of understanding, and therefore possibly lack of control over it, might have contributed to this increase in difficulty.
Furthermore, the player is constrained at ground level depending on the environment tiles. These particular constraints are loosened one height level up, though, and the ability to build over these environment tiles was also not clear to 60% of players.
Both of these issues could be mitigated by informing the player about them better, which might reduce perceived difficulty and enable players to experience the full potential of the game. However, problems regarding the game's controls might also be influencing the perceived difficulty.

Controls



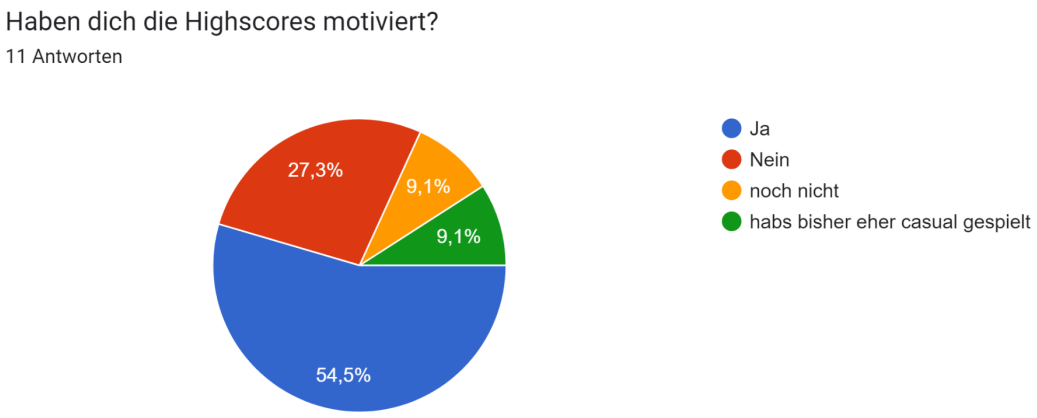
“How did you experience the controls?”, 1: Easy, Intuitive, 6: Bothersome, Unfitting

Even though a majority of testers rated the controls to be at least somewhat easy, not one person was perfectly happy with them. Specific wishes for a different control scheme are the most common responses we received. Some people wished for alternative methods of input for matters which work well enough on their own, like controller support or moving the camera with the mouse instead of the keyboard. Others mentioned problems with the current control scheme and ideas to improve it.

Accessibility

There were no obvious shortcomings with regard to visual accessibility in our tests. However, we did not specifically select testers to ensure diversity in this aspect, so these results might not be representative.

Highscore Effectiveness



“Did the highscores motivate you?”; blue: yes, red: no, yellow: not yet, green: played casually

The highscores were the most divisive topic of the playtest. Half of the testers were motivated by them. However, only 27% expected to not be motivated at all, with the other two responses specifying that they simply did not achieve enough proficiency at the game to figure out if the high score system would motivate them later. Some players criticized the scoring for being unclear or boring, though. This could be solved by either explaining it better or exchanging the method of scoring, e.g. calculating the score with a clear focus on speed.

Problems discovered in interview/through observation:

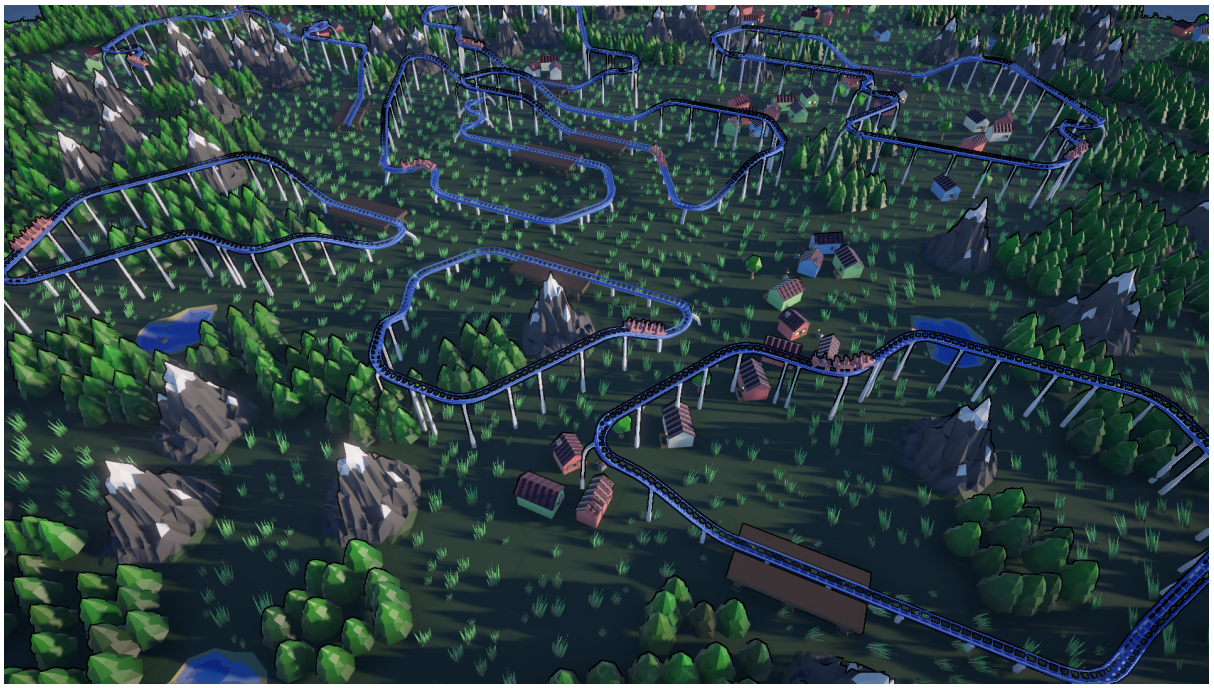
Some issues were not mentioned at all on the questionnaire and only talked about in interviews or observed during gameplay. For one, after completing a coaster it was not clear to multiple testers what they should do next. Adding this point to the previously mentioned explanations might support players in this regard.
Moreover, after finishing a game and having the coaster crash, the player is still able to continue building. While this is merely not yet implemented on our part, this did lead to confusion among testers, some of whom were unaware of the game being completed.
Restarting a finished run was also not immediately clear, since there is no restart button present in the game over UI and the subsequent score screen.
Finally the integration of the wind system into our game is still lacking, too. Even though many players were able to discover it, it has, apart from the visuals, little impact on the gameplay. As some players were hoping for a restriction on the player in higher height levels, integrating the wind system further should alleviate both problems.

Free-form answers

Two questionnaire questions remain to be analyzed. First, if the player felt anything lacking from the game. Here, a variety of responses were given, with no specific point mentioned multiple times. This includes things like a feature we still hope to implement: a game over death animation. Someone mentioned a feature we decided to forego for now: a preview of the next piece of track to be placed. One person also advocated for a feature we completely

missed and definitely have to add: a small jingle tune as a small celebration when completing a coaster. Destroying previously built rail pieces is mentioned as well, though that might be out of our scope for now due to a variety of edge cases that would have to be considered.

The question “Do you want to continue playing?” reveals a very positive general sentiment, as all players wanted to continue playing. The prompt to elaborate reveals a variety of motivations, including highscores and fun, as well as interest in the game’s mechanics and functions. Players also liked the game’s design, with every participant seeing it as consistent and no further elaboration being provided.



General positive sentiment: All players wanted to continue playing, this one did.

Finally, the last question of the questionnaire simply asked if the participants had anything else to tell us. Apart from general appreciation, which we are glad to hear as well, some players specifically praise the day-night cycle and our choice of music. One player added a request for a looping or a jump, the first of which we hoped to add and the second being an idea with high potential.

Conclusion

First, we want to reduce the factors of unwanted difficulty due to players not knowing what to do. To achieve this, we have already prepared the framework for a full tutorial menu. Here, next to a short piece of text, a repeating video plays, showing textual information. With three tips per page and multiple pages, information shortcomings from the level of difficulty section should be mitigated, making the game more readable and less stressful. Apart from basic building tips, the content will be catered towards clearing up the two main unclear aspects of the game: piece placement and coaster speed.

We also plan on implementing some suggestions regarding the user input. Some features require a bit more code. This includes remembering the last height level the player built at, and defaulting selected pieces to this height level to make building at higher levels more streamlined. Secondly, we plan on having rail pieces automatically snap to a valid rotation, if one is present. Switching between multiple of these valid rotations, like when selecting between a right and a left turn, can then be done with only one button more easily. One tester mentioned a “reroll” button to randomize the player’s entire hand of three cards, which we will also implement and evaluate its impact on the rest of the game to see whether we want to keep it.

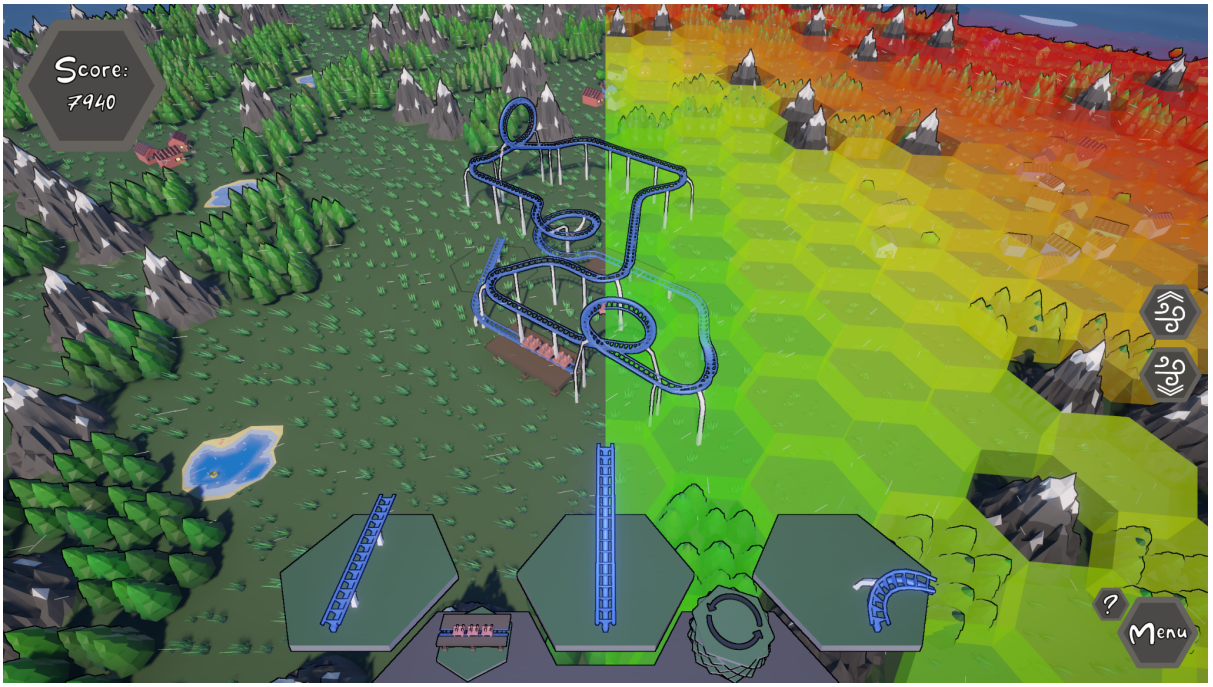
Less code intensive are changes which simply alter the input scheme. We plan to place height level switching additionally on the space bar and discarding pieces on the right mouse button. However, we are going to experiment with this and might change it in the future.

Finally, we might also see about approaching the question of player motivation. We might try and improve motivation through highscores, either by adding information on them to the aforementioned tutorial, or by changing up the scoring method altogether, for example adding coaster average or top speed to the calculation, or as a score on its own. Furthermore, we might also explore other avenues of motivation, like making our developer-intended photo mode (hiding the UI) more easily available, even changing the controls of it, to entice the players to share images of their creations on social media.

- Conclusion -

Summary

We developed a roller coaster construction game exactly as planned. Our game world is generated on a hexagonal tile map inducing the look of board games like “Catan". For the world generation, a wave function collapse algorithm is used at game startup to decide which tiles contain which environment. Furthermore, to increase variety, all environments are randomly rotated and rivers are randomly generated. Additionally, a dynamic wind simulation was developed, simulating the wind strength (flow rate) and direction of the wind at two height levels for each tile updating at each frame and always considering neighboring tiles from all height levels. The wind is visualized through colored boxes on the tiles, with the wind strength encoded in the coloring, as well as, less obtrusive but also less clearly, by particles floating through the world at all times. The more distinguishable presentation can always be enabled and disabled through the UI buttons on the right side of the screen.

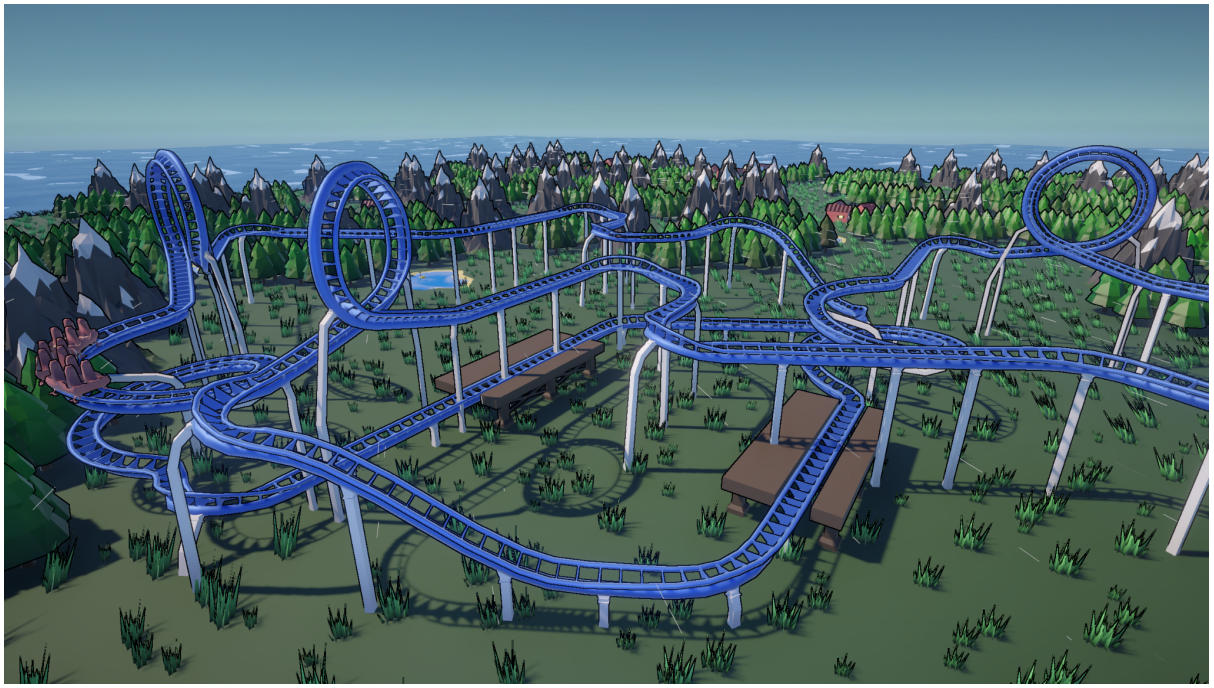


The wind is visualized as particles (left) and as colored boxes (right).

The environment consists of Mountains, Forests, various Houses, Lakes and the previously mentioned rivers. Visual polish of the environment is achieved by using various handmade shaders, like the cartoon outline shader, shaders coloring and animating the foliage and ones toggling lights in the houses.

The environment of the world

After the world generation, the gameplay starts, consisting of placing roller coaster track pieces onto the environment to complete the coaster and gain score. 3D assets for Coaster track pieces (straight, curved, narrowly curved, slope, looping and helix pieces) and the previously mentioned environment tiles have been completely handmade with blender.

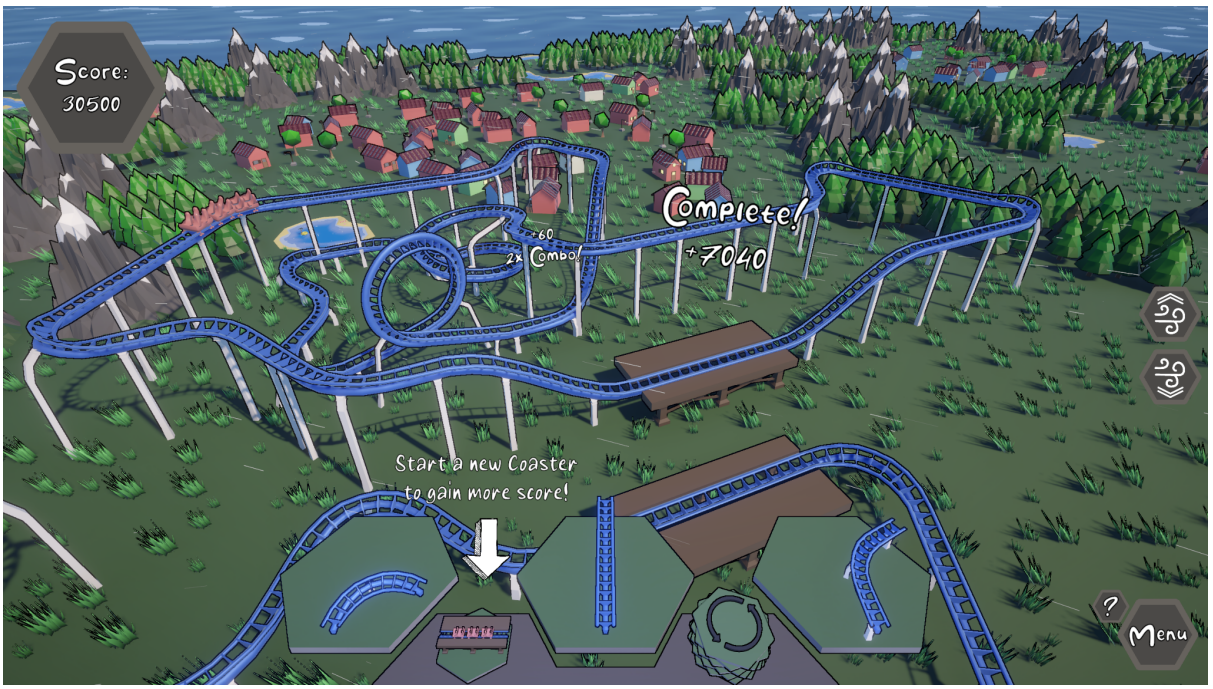


The variety of coaster track pieces

Furthermore, many checks are added to ensure that track pieces can only be placed at valid positions on multiple heights, which is also necessary, since environments can block placement. Mountains block both height levels and Forest/House tiles block all tiles on ground level, with grassland and water tiles enabling building on both height levels. The players can always decide to pick one of three track pieces, however they are not forced to place one of them, as they are able to discard single track pieces or reroll all three available options. Nevertheless, they are still rushed to build, since the coaster is steadily advancing towards the end of track and might crash if the player fails to build more in time.

One page of the tutorial with a winter landscape in the background

The game is rounded off with various audio clips and menu screens, including a tutorial, and other small features, like the ability to select a world and coaster theme.
Since the alpha release and playtesting, we were able to make various changes. Directly addressing playtest results, we changed some controls and improved their feel in general by remembering the last height level used and defaulting new pieces to that. Furthermore, the UI was improved both stylistically as well as functionally, with an added tutorial including video tiles to support players better in the first few minutes of the experience.
Another aspect that was often unclear was the scoring system, so we implemented a full rework thereof. Instead of the complicated concept of tracking two different distance measures, one derived from the distance of placed tiles and one from the distance the coaster has traversed, the new system gives the player score by placing pieces and encourages interesting gameplay by rewarding placement of different pieces each time through a combo system.



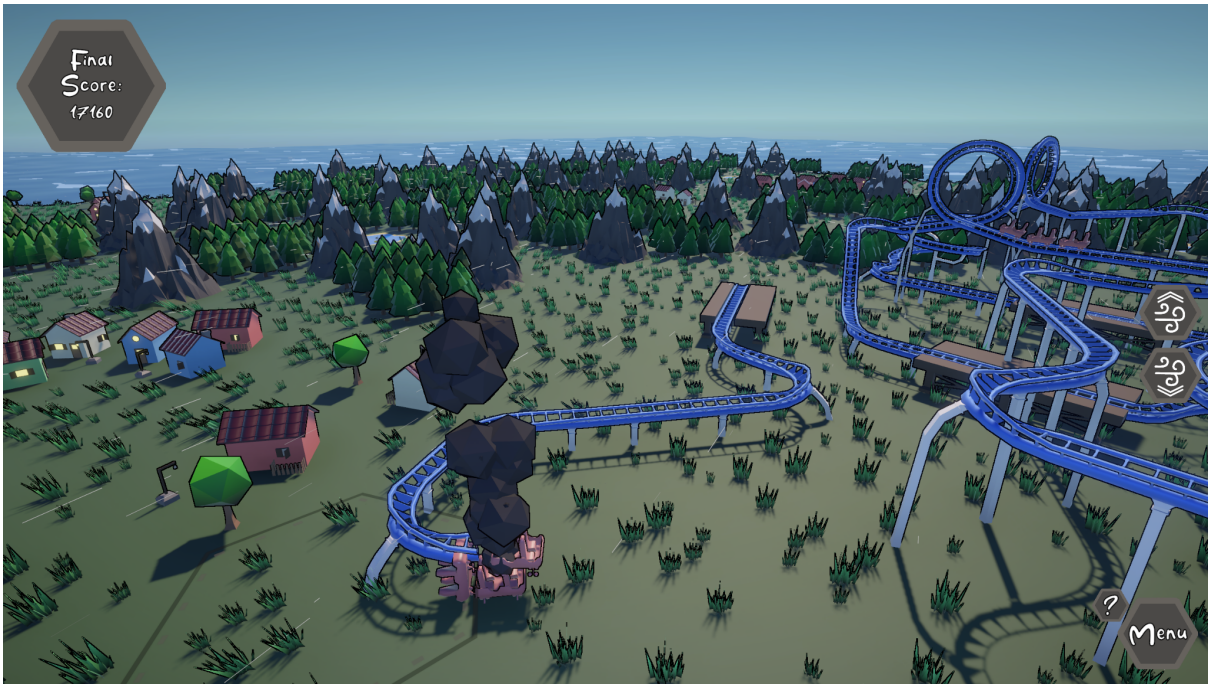
An excerpt of the score system

The coasters themselves are also made more interesting by the two new track pieces that were added, the looping and the helix. The looping is functionally a straight piece, but has a different effect on coaster speeds, since the coaster first needs to crest the loop, slowing down in the process, but then finally overcomes it and moves downward again, speeding up. The helix offers an alternative to the straightforward slope, with a reduced slope angle and longer track length that the coaster must traverse. To ensure these special new pieces don't dominate the coaster, their probability of being offered to the player is comparatively reduced.

The wind system was integrated more with the game first through a particle system showing the wind direction and strength even with the more concrete wind visualization disabled. This further visual clarity helps the player when confronting the game's new challenge, an inability to build in tiles with strong winds.

If the player fails due to that, or simply by running out of track due to indecisiveness or for other reasons, the coaster now drives off the edge of the track and falls with all of its nonexistent passengers to a tragic demise with a pillar of smoke ascending from the site of the accident.

If the player manages to avoid this, we also added one more feature to increase variety. Players can now select a color scheme for their coaster, and even a season or theme for the world. One day, for an even more final release, we plan to add score unlock requirements, so only the most proficient of THAT Coaster Game players are able to experience the world of Candyland!



The game ends when a roller coaster crashes.

Tim

Developing a game over the whole semester with a small team seems always challenging but in this case my experience was a bit different. Mostly, we really took the time to keep going forward, even though we began rather slowly, we improved a lot and put a lot of effort and time into the project. Due to this great teamwork we got a lot done and always pushed each other to get things done as we shared the same vision of the game. Everyone wanted to see the final result. Sure, everyone encountered problems or hurdles, but everything could be accomplished together. Luckily, everyone of us wanted to do this project as structured as possible, so we decided that Helga should be our project manager who also did a great job keeping us on track especially in our 1-2 meetings per week.
Overall my experience was great but we wanted almost a bit too much as a great part of my weekly time had to be taken for the project. However, I learned a lot of new skills considering development and game releases.

Our initial design idea is nicely represented by our current game state. Almost everything we wanted to accomplish was added to the game. So in our case the initial idea matches the current state/idea of the game. Only fewer changes had to be done but, those only reduced working time while not changing the game concept, like the number of height levels or skyscrapers for the environment. Only one part did not match my initial idea. The game is not really as relaxing as I thought it would be, even though it does not make the game worse in any kind. The playtesting by us and our testers showed that it was great to have a bit of competition/stress induced by the coaster, driving around.

For our time schedule, there is not much to say. We were mainly able to follow it, but especially before the interim demo we got a lot into stress as some core elements were

missing that were required to show a working game. Afterwards everything worked according to the plan at least from a timing aspect. I personally immediately started developing without having a week where nothing has been done. So I never got into time issues and always managed to meet my project plan. The project structure definitely helped to keep staying on track especially as it was clearly defined when the game should be in which state. I would love to have more time in total, but the distribution of the milestones and time in between was great. Although I am happy that the prototyping phase was not done this time as it only hindered the development process in my last project without helping in later development.

My personal impression of the course is great! It was very nice to work on a game for such a long time as this gets a little bit missing during the bachelors program. Therefore it is great to have this chance in this course. It even led to our great game result of which I am very proud and I even catch myself playing the game from time to time just because it is so nice to see what we achieved in such a short time period. So, yes the time was short but it is restricted due to the semester time. Therefore, the schedule was totally fine. Hopefully we will continue working on the game!

The biggest technical difficulty I encountered was clearly the procedural world generation as I never did this before and was not aware of the wave function collapse. This world generation difficulty was again a little more complicated when I added the river generation which took quite a long time to be as expected.

Working with the theme was totally fine. It was very helpful, as the whole team restricted their minds to the theme and it was much easier to imagine the ideas of the others as you always have “roller coaster” in the back of your mind. Even though brainstorming about the game idea took some time, the theme definitely led us into this direction. However, after finalizing the game idea and conceptualizing everything, we never worried about the theme anymore. Total freedom is always nice, but especially when no one has the role of a product owner you can very easily run around in circles on your way of deciding on a game idea. I would only suggest offering more than one theme and selecting one of them together with all participants. However, I would not say that the theme improved or hindered our game. It only led our thinking into a specific direction.

In my next project I would like to keep having a project manager as I think this heavily improved the workflow. Furthermore, meeting 1-2 times each week also helped a lot to keep in touch and find possible bottlenecks or just finding time to help each other. The only thing I would like to change/add is the usage of issues/jobs that have to be written down probably into some kind of backlog and then at the meetings it is decided who has to deal with them and when they are done.

My greatest success in the project was definitely finalizing the world generation with its own handmade environment assets and combined rivers. It is just so nice to see the world changing on every game startup and knowing that it was your own work. Additionally, I have to mention our UI design as I really love it and took a huge part in redesigning it to match the hexagonal style of the game and making it look more modern with the darker coloring.

From my perspective, the game is great! It looks good and feels good and is absolutely fun to play. So yes I am happy with the final result of the game and I absolutely consider the game a success.

My only recommendation for the course organization is just as mentioned previously, offering multiple themes and collectively deciding which theme will be used for every team in that semester.

Helga

The computer games laboratory was overall a uniquely challenging but also rewarding experience. Both stem from the scope of the project which is larger than that of every other project I have ever worked on. This is why I found planning and coordinating to be more difficult and the outcome to be much cooler than every other game.
In the beginning I thought I knew how complex the project would be. I specified my ideas regarding the wind system in as much detail as I could. But while implementing the wind system, there were so many things I had not considered yet. The more time passed the more I realized, I totally underestimated the complexity of the project. Especially the points of connection between the wind and the other parts of the game needed much more attention than what they initially had gotten.
This also reflects in the development schedule. We planned the development of the singular modules of the game, but not the connections between them. Granted, we did not pay particularly much attention to the schedule, as long as we were making progress and were not in the last week before a milestone, but still. Early on I used the task distribution to have an overview over everything which needed to be done. But I found a lot of the actual tasks were not considered for, so I switched to taking notes during meetings. Unfortunately this led to a lack of a complete overview, so there frequently arose important little tasks shortly before milestones. I would have preferred to either be able to plan those tasks ahead of time or to have some buffer time before each milestone for them.
In that sense I am quite glad about the project structure we were given. While I did not enjoy having to write the chapters for our game design document, the milestones themselves helped us to stay focused on what needed to be done next. I would like to mention one idea for improvement though: The playtesting milestone chapter mentioned the importance of testing all throughout development. This also perfectly matches with what I learned in other classes. But we never thought to let other people play the game during the earlier stages of development. Perhaps next time the instructors could mention testing with one or two external people during earlier milestone presentation meetings.
In conclusion I am very satisfied with the course, not because it fulfilled my expectations, but because it did not. I thought I would mostly apply the knowledge I gained during the past years of studies, but as it turned out I also learned incredibly much. In addition, the lessons I learned are lessons I hardly could have learned in any other way. Hearing “Plan not only the development of the part modules but also the connections between them.” in a lecture never could have had the kind of impact as realizing my own shortcomings did.
Since my experience with this course is so heavily defined by my work as a project manager, it is hard to say what my feelings toward the game are. I am certainly happy it turned out as good as it did, but mainly, I do not have any feelings at all. My thoughts and feelings

regarding this project have much more to do with our team and the cooperation within, both of which I am very happy about.

Alex

The roller coaster theme is closely interlocked with our game. Without it, we would not have had the exact same game concept, the theme is always and immediately present, and even if it weren't there but we had thought of the same game concept, the game would be the same. As such it can't really be called working with the theme per se, and for our team at least the theme simply provided just the right amount of restriction to arrive at our game concept.

Our game turned out to be very similar to the original design concepts. Only few concrete details were changed, like the four planned height levels for the high target being reduced to two as a design choice, despite us possibly having had the time to add at least a third one. The largest difference between the game now and its original design are the minor features bringing the game together, many of which were either not planned at all or at least not planned in the scope in which they are present now. This simply means we had more time than expected for general all-around polish.

Especially in the beginning I was mostly able to follow my schedule well, excluding some slight delays due to the technical challenges mentioned later, not amounting to much more than a week at any time. However, later I transitioned to completing various small new unplanned tasks, while one task of mine, adding a fog of war, turned out to be unneeded. Those small tasks didn't take away time from planned tasks though, and apart from the one task that was discarded, I was able to complete all my tasks first and then work on the newly added extras or whims.

The existence of frequent milestones definitely ensured we integrated our progress with the rest of the team often and focused on key features at the right times. Playtesting specifically also brought up some helpful minor details we had overlooked, like the lack of a Coaster Completion sound effect or a change in controls to default to the last-used height level when building, but also revealed some issues in the control layout in general that we were able to address afterwards.

Having successfully developed a video game in a more structured environment with regular milestones and guidelines for different aspects of non-coding game development like playtesting, the course definitely fulfilled my expectations well, and I have no complaints nor improvement proposals regarding the course. The resulting game is the one I am most proud of among the various game jam and university projects I have completed, so I am definitely happy and proud of the team's work on it, as well as my part in that work. I'd do it all again too, there's nothing I would do differently, and with how well things went there's also nothing major I see that I would want to do differently next time. Various lessons were mainly learned in working with Unity's shader graph, towards the end now also the vfx graph, and various other small coding improvements.

My own biggest technical hurdle was working with Unity's spline package. The preexisting framework was simply not up to the task of animating objects smoothly across multiple splines assembled at runtime, and as such had to be extended directly, which I achieved by simply copying over the relevant class and directly making changes to it. Apart from that, learning to work with shaders and properly implementing an outline shader also proved challenging, with an earlier version requiring a separate shader and render pass for each type of animated foliage.

My own greatest success during the project would be the graphical impact the various shaders I developed in the shader graph had, with the smooth coaster movement in use on every coaster being a close second. Seeing the outline shader affecting the foliage created by Tim animated by the shader receiving data from Helga's wind system using random colors picked from a configurable color space simply brings everything together well, narrowly beating out the smooth movement of the coaster cars across Tobi's tracks that players can place down.

Tobi

Initially, it was somewhat difficult to find a good and tangible idea according to the course's theme, especially since the concepts were either very close to a literal rollercoaster game, which felt like having a small degree of freedom, and more metaphorical ideas, like an emotional rollercoaster, which felt a bit too far fetched. Having some theme is certainly a good idea, since creativity often arises from working within restrictions. After a lot of brainstorming and coming up and then discarding ideas, we settled on a concept that seemed fun to play as well as being realistic to execute within the timeframe of a semester. In the end, we stuck surprisingly close with our original plans. Concepts we got rid of or changed are for instance an economy system, the layout of the track piece stack and a fog of war that turned out not to be necessary.
The biggest challenge of my part was certainly programming the game logic of how to place the different coaster tracks. It involved a lot of edge cases and special behavior like tracks which change the height level etc. which were prone to bugs. Fortunately I had some help from Alex with looking over the game logic which helped a lot in finding mistakes and keeping me sane.
The biggest success on the other hand was certainly seeing the coaster carts in action and following the tracks I had modeled for the first time and similarly seeing how each new track piece I modeled fit so nicely into the game. Having the opportunity to do some modeling in the context of a university course was very welcome and enjoyable for me.
In future projects, I'd like to try and tackle different areas of work I haven't touched on like the shader graph.
Concerning the schedule, we mostly managed to finish our tasks on time, although our goals were somewhat set with a margin of interpretation. For example, I only added the looping track piece after the playtesting although it was originally planned to be done by then, but apart from that, we reached our goals as we planned them out in our time schedule. Having a fixed deadline from the course also helped in keeping up with development and consistently making progress, not having such a schedule most likely leads to delayed development into the exam session. Especially having the Alpha Release somewhat early

makes the team focus on delivering a playable game and not work too much on details and polishing, which can easily and more safely be done after the Playtesting phase.
Our team’s internal organization was always on point, with frequent meetings of about 1-2 times per week, which definitely helped to stay on track and get things done in time to, for example, also write reports and prepare for the milestone meetings. In particular, we decided on a team leader, who would have a list of topics to be discussed as well as keeping the meeting on track, which turned out to be very useful. In general, working with this team was a breeze and I’m proud of everybody's contribution.
In conclusion, I feel like our game turned out fun, polished and the best game project so far I took part in during university. We set ourselves realistic expectations and tasks and managed to create a well-rounded game this way. The time to work on the project is of course always tightened by other university courses as well as a few months simply being not very much time for developing any game, but within the university-context of mostly having projects/exercises which merely accompany a course, I enjoy the rather big scope of the Games Lab.