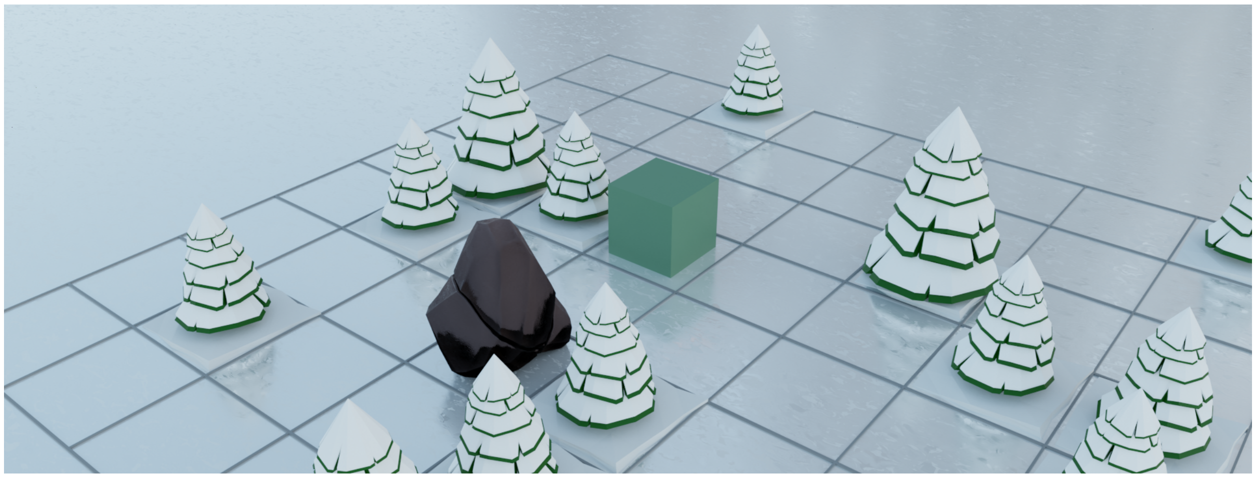
# Game Idea Proposal: qubi

Team *FünfKopf*:

Felix Brendel
Jonas Helms
Van Minh Pham

November 2020



## Contents

## 1 Game Description

We as the team FünfKopf believe that great games build on simple concepts. A perfect example for such a game is Portal (Valve, 2007). The game magazine GameStar even ranked it number one on the list of the 250 best pc games of all time, which they published as the special edition issue 01/2020.

### 1.1 Game design of Portal

Portal builds on the concept of portals that connect two positions in the game world. When the players enter one portal, they instantly appear at the location of the other portal. Using this, the game is divided into levels, each with a specific starting point and target the players have to reach.

Just this basic concept – teleporting to another position – alone however, is not enough to create an interesting game. Starting from the simple concept one can begin to shape the game mechanics by exploring different ideas. In the case of Portal they might have looked like this:

1. What if the players could place the portals themselves?
    - Give the players a portal gun and let them shoot the walls to place portals on them
2. What if there are some surfaces, the players cannot place portals on?
    - If players can place portals everywhere, it might be hard to create challenging levels, so also use surfaces, where no portals can be placed on
3. What if some levels require the players to transport objects in the levels to solve them?
    - Let players pick up and carry one object at a time. This also harmonizes with the Portal mechanic, as players can carry objects through the portal, adding depth to the game design.
4. What if there are some barriers, that when moved through, destroy the carried object?
    - Letting the players carry the objects freely through the level might make the levels too easy, but barriers which limit the movment of the objects adds depth to the level design

It is interesting to note, that idea 2 and 4 seem to be direct responses to the ideas 1 and 3.

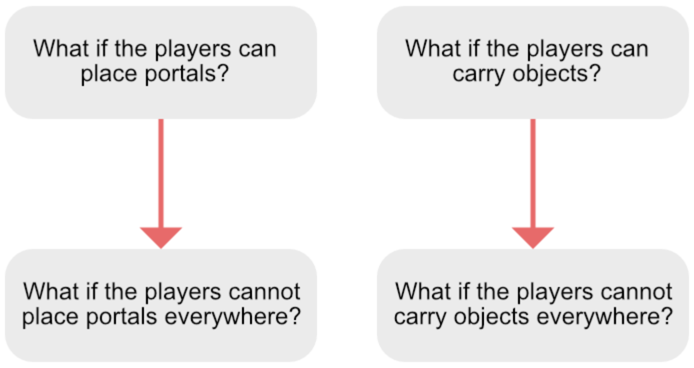

What if the players can place portals?
What if the players can carry objects?
What if the players cannot place portals everywhere?
What if the players cannot carry objects everywhere?

Figure 1: The explorative design decisions of the game Portal

## 1.2 Game design of qubi

Seeing this we also wanted to design our game in a similar fashion. Our game concept follows the given theme *wet and slippery*: **The players should slide around on a slippery floor**[1]. This is our main mechanic.

We came to the conclusion that the most fitting gametype for this mechanic would be a puzzle game where the player controls a simple object. After agreeing upon this as our starting point we will go over our decision making process for the rest of the game design in the following section.

### 1.2.1 2D vs 3D

It was clear, that a 3D game would fit better to the setting of this practical course. However we also really liked the idea of 3D puzzle games as it has the potential to emerse the player more into the game.

### 1.2.2 What kind of object should the player control?

**A sphere** There are some games that successfully use spheres as the player characters, however it seemed that a sphere would not feel like slipping on the floor, it would rather be a rolling motion.

**A cube** Could be a good option

**A tetrahedron** Could be a good option

Of course, as with Portal, we had to think about explorative questions to experiment with additional mechanics.

### 1.2.3 What if only part of the floor is slippery?

If there are some sections on the ground that are not slippery, we could make the cube flip on it's side when reaching a dry section of the map. We can make this a secondary game mechanic, where the sides of the cubes are different in some way – maybe they are colored differently, or have different shapes on them. An additional condition for finishing a level could then be to match a specific side of the cube with the target tile. This means that both the route and the orienation of the cube have to be considered by the player resulting in another layer of gameplay depth.

### 1.2.4 What if the object could also unfold?

We wanted to add only one more mechanic to our game – one more action that the player has, that will let us design more challenging and interesting puzzles. One of the first ideas we thought about was the objects ability to fold open. By unfolding the players can bridge slippery tiles or unfold ontu slippery tiles. By introducing the unfolding, we can also add multiple finish tiles to the levels, which all have to be satisfied simultaniously by unfolding, for the level to count as solved.

The two shapes we already considered beforehand, cubes and tetrahedrons, can unfold onto a two dimensional plane. For a cube the unfolded form would be located on a regular square grid, while a tetrahedron unfolds to a regular triangle tiling. For this game we first decided on using the cube as a base shape, as it allows for a simpler map structure and therefore shallower learning curve for the player.

[1]Of course the word slippery is a bit vague. We thought about what it means to us if something is "slippery" or "wet". We came up with these simple definitions: Slippery is an attribut of a surface which implies that the surface friction is low and the attribute slippery can only be observed when another object is touching and sliding on it. "Wet" on the other hand is to us the sensation you feel when you touched a liquid. Usually it is connected to a reduced surface friction of wet objects.

A sample unfolding can be seen in Figure 2.

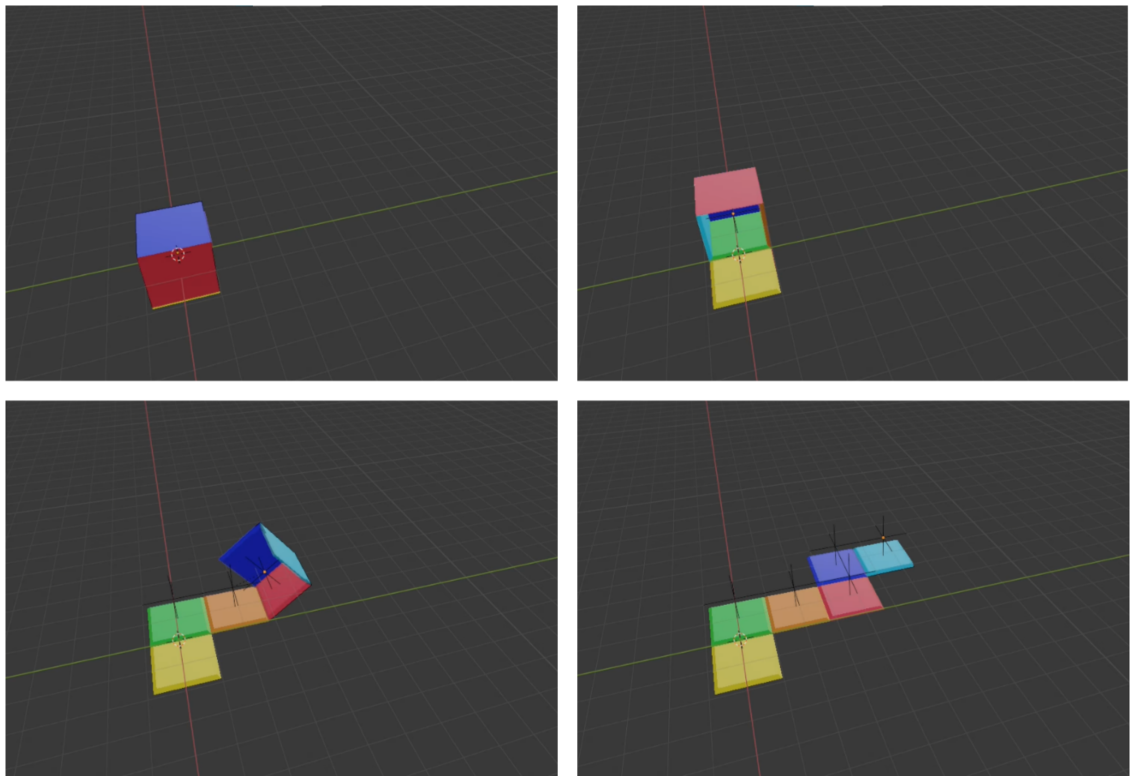


Figure 2: The unfolding of the cube, following the player's inputs of `up`, `right`, `right`, `up`, `right`

Note also, that there is inherently a limited amount of valid unfoldings due to fundamental geometric principles. An illustration of all possible unfoldings can be seen in Figure 3. This is the kind of complexity in a puzzle game we really want to highlight. Everybody can imagine a cube flipping over and think about which side will be facing upwards or downwards, but using this complexity – which is really accessible to everybody – we can then design levels that require the player to plan their movement, flips and unfolds ahead to finish the level.

### 1.2.5 Bringing it all together

qubi is a tile-based puzzle game, where the players control a cube and try to make their way to the goal. There are two different surface categories: slippery and dry. On slippery sections, the player can give the cube an impulse in one direction but then has to wait as the cube slides. The cube slides until it hits an obstacle or reaches a dry spot and comes to a hold. While sliding the cube does not change its orientation. If the cube moves from or to a dry tile, it flips on its side. This is important because now the cube will have a different orientation and each level requires the cube to land with the right side up on the finish tile. As an additional mechanic, the players can unfold the cube as a method of movement and overcoming otherwise impossible gaps in the levels.

The first levels will only feature the sliding mechanic as we want to let the player get used to the fundamentals of the game. And only then we will introduce dry surfaces and force the player to make smart use of both the sliding and flipping mechanic. Lastly for the last few levels we will let the player unfold the cube and open the possibilities for the most strategic use of all three mechanics. With this approach we hope to archieve a steady learning curve, that keeps the players motivated.

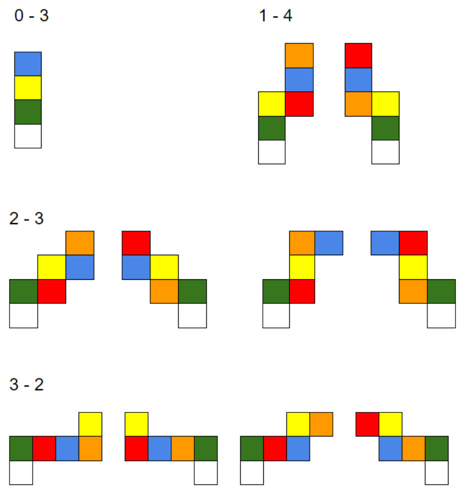


Figure 3: All possible unfoldings of the cube starting on the white side and first unfolding onto the green tile; all other possible onfoldings are permutations of the colors and rotationary symmetries

Compared to the desing decisions of Portal from Figure 1, the tree main design questions for qubi can be seen in Figure 4.

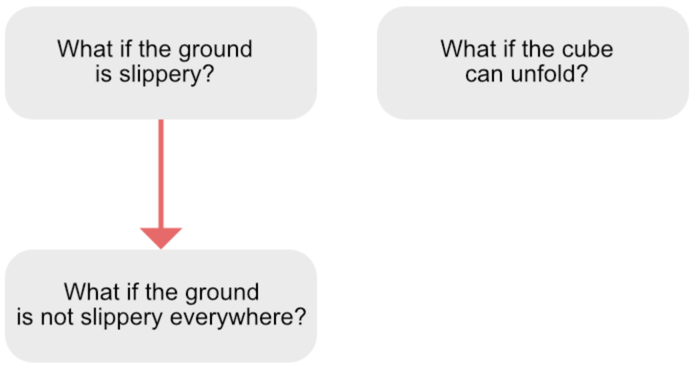

What if the ground is slippery?
What if the cube can unfold?
What if the ground is not slippery everywhere?

Figure 4: The explorative design decisions of qubi

## 1.3 Setting of qubi

At first glance an ice setting of some kind would seem to be the most obvious choice given the theme of *wet and slippery*. For the slippery tiles ice surfaces can be used while for "dry" tiles snow surfaces seem appropriate. Additionally we have made the decision to widen the spectrum of settings to more nature themes such as setting levels in a jungle with water on the slippery tiles. With levels being structured in chapters new nature settings can be used for different chapters, e.g. whenever a new gameplay mechanics is introduced.
qubi is a rather abstract game as it revolves around moving a cube in order to solve arbitrary puzzles. Not being bound to anything except the theme of nature and *wet and slippery* allows us to be more

creative with the settings for different chapters perhaps even giving the opportunity for a desert setting.

## 1.4 Visual Design of qubi

The focus of qubi is solving puzzles and therefore we try not to be intrusive with our visual design decisions. To introduce the controls of the game, we might only show a picture of the buttons the players can press to move the cube[2]. A puzzle should never be hard to solve because something was hard to see.

## 1.5 Audio Design of qubi

The gameplay of qubi will be enriched with a calm background music that let's the players focus on solving the puzzles. It won't have a strong rythm or a fast tempo, to not build any pressure on the player and let them solve the puzzles in their own pace – the player should feel cozy when playing the game. To convey this feeling the soundtrack will be mainly composed of ethereal sounding synths, plucks, bells and string instruments. We also want to accompany the music with sound effects for the movements of the cube: sliding, flipping, unfolding, hitting obstacles. These sound effects should be crisp and convey the nature of the movement.

## 1.6 Sample levels

The layout of a simple sliding level can be seen in Figure 5. This is a rather simple level as it does not leave much room for the players to make mistakes.

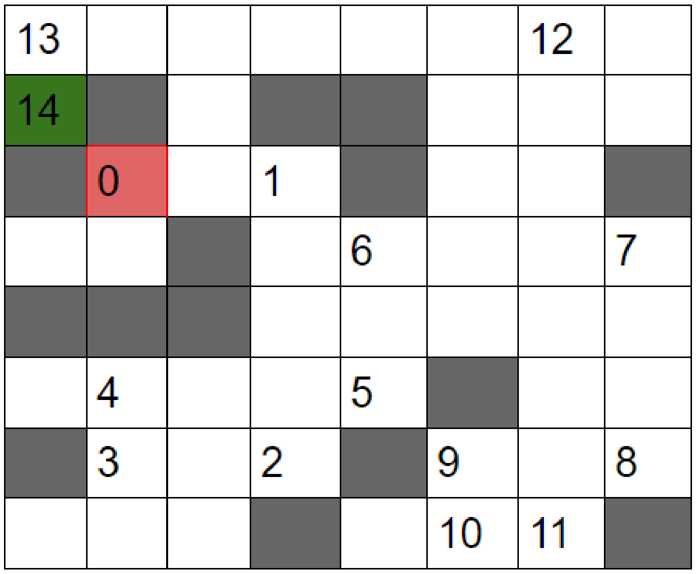


Figure 5: Example level that makes use of the sliding mechanic. The red tile marks the start and the green tile the finish. Gray blocks are obstacles. The numbers indicate a possible path the players can take to solve the level.

An example puzzle layout that also makes use of dry tiles can be seen in Figure 6. Note that taking different routes on the dry tiles impact the resulting cube orientation on the finish tile.

[2]controls

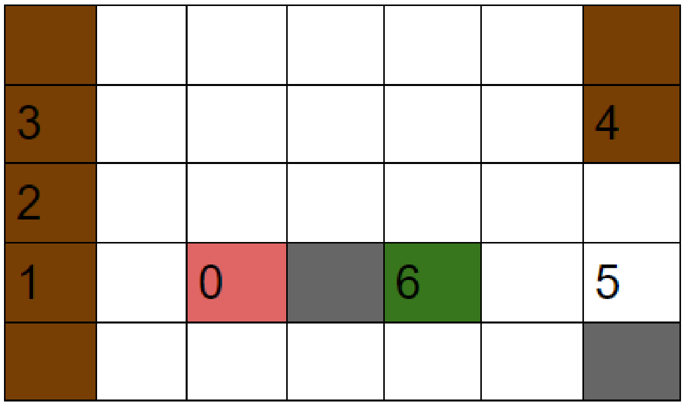


Figure 6: Example level where the user has to make use of both sliding and and flipping to reach the finish in a certain orientation. If another path is taken, the orientation when landing on the finish will be different.

## 2 Technical Achievement

### 2.1 Introduction

The central secondary big bullseye idea for our project is to develop our game idea in our own game engine. Our group always wanted to build their own game engine from scratch and we thought that this practical provided the perfect opportunity to put this into reality. The fact that we can use the second mandatory practical course to further expand on the engine only affirmed this notion as we are then able to segment the development of the engine over two semesters and can leave more focus to the development of the game and it's optimization.

### 2.2 Motivation

The main motivation to build our own engine stems from the fact that we believe that we can reduce the overhead and therefore provide better optimization for our games on all levels of the engine, from the graphics pipeline to resource allocation and automatic memory management. Furthermore we believe that building a game engine from the ground up presents a perfect learning opportunity, especially when trying to find suitable optimizations that fit our design philosophy.

### 2.3 Game Engine

In the following sections we will provide a small overview of the components of the game engine that we want to develop for this semesters project and how we try to optimize these. Furthermore we will go over the features of the game engine that we will most likely tackle in the follow-up project and how we solve the interim solutions for this semesters game.

#### 2.3.1 Graphics pipeline

The game engine will use the Vulkan Graphics API to implement a rendering pipeline. Vulkan is a relatively new API developed by the Khronos Group (maintainer of OpenGL) with a focus on overhead reduction and was released in 2016. Vulkan provides a low-level control over the rendering process when compared to other Graphics APIs and has several advantages that also align with our overall philosophy in the design of the engine:

- The ability to run on all operating systems and devices
- Explicit control over memory management
- Decreased CPU workload due to reduced driver overhead and batching
- Making use of the driver independent Vulkan Loader to access Vulkan API entry points

The Vulkan Loader is responsible for transmitting Vulkan API calls to the appropriate graphcis driver. This means that we just have to connect to the Vulkan loader in our engine and do not have to worry about drivers. Furthermore we can pre-compile our shaders into the SPIR-V binary format instead of compiling the shaders at runtime. This allows the use of a larger number of different shaders per scene and reduces application load times. We want to utilise the ability to use a high amount of different shaders and put this feature into to Extras for the game development (Layer 5) but we will most likely first employ this in the follow-up project.

### 2.3.2 Overhead reduction in the engine

The game engine is developed in the C++ language that all of our team members are familiar with due to our TUM Bachelor courses such as Game Engine Design. We have also taken further steps into the direction of our core concept of overhead reduction by omitting the C++ standard library.

### 2.3.3 Resource Loading & Automatic Memory Management

To increase the performance of the engine we want to make sure that the loading of resources such as a texture map or a mesh is never done redundantly, which is likely the case in a puzzle game as key components are similar between different scenes. In order to implement this we allocate buffers upfront to store all our resources and a hashmap that maps the file paths of the loaded resources to their pointers in memory. If a resource becomes necessary in a scene, we can cross check whether the file path has already been loaded and then reuse the already loaded file instead of reloading it. This means that we will only load the difference between two levels which will reduce load times and create a smoother gameplay experience for the player. The Hashmaps also provide further advantage for the memory management as we can free the memory and GPU memory for the texture resources by iterating over the hashmap and can incorporate this in the scene load/unloading process.

### 2.3.4 Sound System

Sound is very important to our design goal of creating a casual and cozy puzzle game as we believe that it has a relaxing or even focusing effect on the player. We will try to implement our own sound system for the engine but are also considering using an API for example OpenAL if we realize that it would take up too much time of the development process.

### 2.3.5 Physics System

The current point of view in our team is that we will not implement a physics engine as part of this semesters project as it would exceed the scope of the engine building aspect. We will instead use keyframe animations and bake the limited number of physics interactions directly into the animations or generate them procedurally. This also comes with the advantage of having a tighter control over the cube behavior as we want the players to struggle with the puzzles instead of controls of the cube. Further expanding the engine by implementing a physics engine is something that may be tackled in the follow up project.

#### 2.3.6 Animation system

The animation system will be a very important part of the engine as it will substitute our physics interactions and help to increase the graphical fidelity of the game. Implementation of the animation system will start very early on and the core functionality of keyframe animation will be finished for the interim demo.

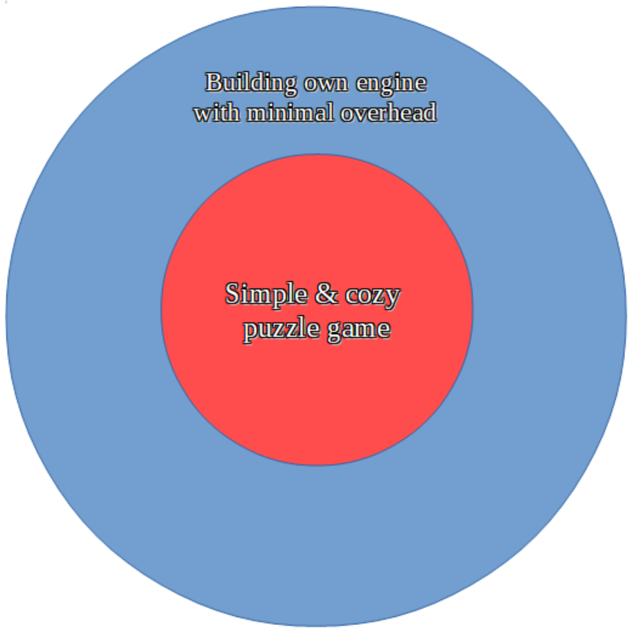
#### 2.3.7 Particle System

A robust particle system would be a nice addition but we currently assume that we might have to substitute it using keyframe animations aswell. Current status is that we will develop a particle system if there is time left over after finishing our High Target goals (Layer 4) of the development.

#### 2.3.8 Physically based rendering

One goal of the High Target of our project (Layer 4) is to increase the graphical fidelity of our game by implementating a physically based rendering system. The term "physically based renderer" is purpously left ambiguous as we want to check out how many components of a physically based renderer we can implement while still keeping a smooth performance.

## 3 Big Idea Bullseye



## 4 Development Schedule

### 4.1 Layers of Development

1. Functional Minimum:

- One basic level(including start and finish tiles)
- movement of cube
- slippery floor type
- Engine Goals: Graphics pipeline, loading meshes, textures, animation system, interface, particle system, tangent space normals

2. Low Target:
   - floors with grip
   - have sides differently colored
   - finish condition: cube must land on a specific side on the finish tile
   - Load Levels from .txt file
3. Desirable Target:
   - cube can be folded open
   - multiple finish tiles
4. High Target:
   - different worlds
   - implementation of physically based rendering
5. Extras:
   - have players set floors to floor types themselves
   - make use of hight amount of different shader
   - implement particle system in the engine

## 4.2 Task Distributions

For every milestone we schedule the following:

| Task | Member | Spent hours |
|---|---|---|
| Project documentation | All | 3 |
| Presentation | All | 2 |

In general the tasks are distributed as such:

| Components | Tasks | Member(s) | Planned hours |
|---|---|---|---|
| Brainstorming | | All | 3 |
| Prototyping | | All | 10 |
| Engine Work | Implementation of the 3D graphics engine for loading meshes, Animation System . . . | All | 200 |
| Inputs | Ensure movement of cube through buttons presses | Minh | 20 |
| Gameplay | - Cube can be moved<br>- Cube can be folded open | Minh<br>Felix | 23 |
| Win Condition | Cube has to land on finish tile<br>- on a specific side<br>- on multiple finish tiles at the same time (by folding open) | Minh<br>Felix | 30 |
| Level Design | Designing puzzles, challenges | Felix<br>Jonas | 40 |
| Animations | Ensure different movement behavior on different tiles | All | 20 |
| Art | - Environmnet Meshes&Textures<br>- Particle effects<br>- Original music | Felix<br>Jonas | 60 |
| UI | - Convey basic information to player<br>- Keep it rather simplistic | Jonas | 10 |
| Playtesting | Testing and fixing | All | 10 |
| Trailer | | All | 30 |
| Additional Content | - Different world designs<br>- Players setting floors to specific type themselves<br>. . . | All | leftover time |

The exact timeline can be observed in timeline.pdf (which will be updated regularly).

## 5 Assessment

qubi is designed to be a cozy and fun puzzle game for people to enjoy regardless of prior experiences of puzzle games or even video games in general. To achieve that, the game will have to be easily accessible not requiring a lot of prior knowledge. Initially players will merely have the cube slide on slippery tiles in order to get to the end of the levels. Levels in later chapters will add more and more mechanics which effectively raises the difficulty level. We hope to keep players invested that way. The difficulty curve in form of the levels provides one of the biggest challenges in the development as a sudden difficulty spike can lead to a lot of frustration while a low curve may bore players.

Although we intend to provide players with healthy challenges along the levels, we generally want qubi to be a relaxing game to be played from time to time.