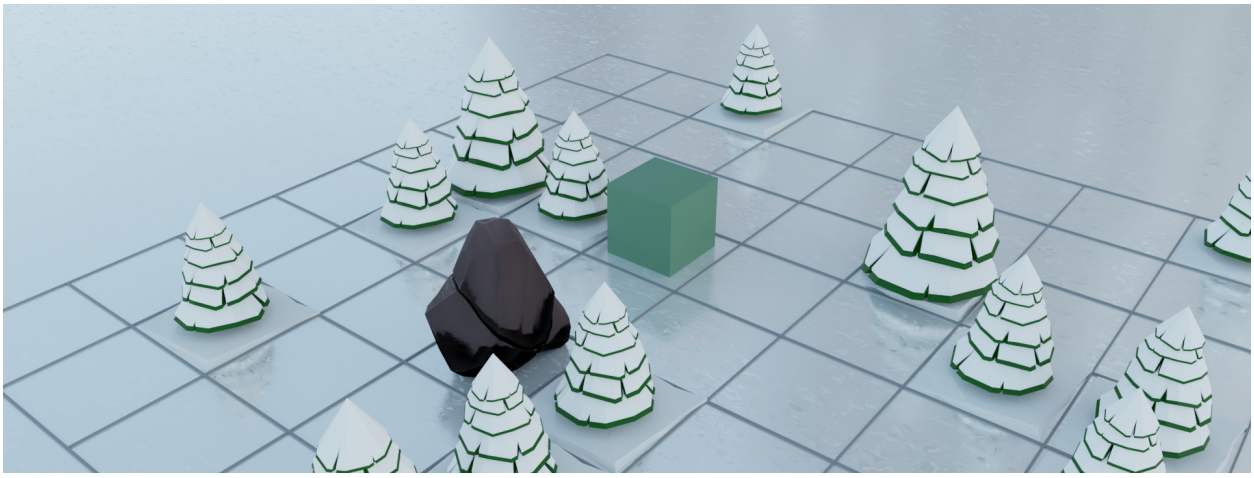
Alpha Release: qubi

Team *FünfKopf*:

Felix Brendel
Jonas Helms
Van Minh Pham

December 2020



Contents

1 Alpha Release Content

With the implementation of the last unfolding mechanic players now have a lot of options to approach solving levels. In levels involving all major gameplay mechanics this may lead to a bit of uncertainty. Despite this we look forward to how players experiment and play around with all the mechanics in these levels.

1.1 Gameplay Features

In terms of gameplay features we are finished. The cube can be moved freely and all tile types (slippery, dry, finishes) have been implemented. Finally players can now hold the space key in order to unfold the cube. Letting go of the space key will have the cube refold on latest tile the cube moved onto. This adds additional depth in gameplay options and level design, e.g. featuring multiple finish tiles that the cube has to be on at the same time. This feature is only accesible on dry tiles as it would make the game too easy and impossible to create harder levels. Furthermore unfolding is exclusive to specific levels later on as it may also trivialize earlier levels with dry tiles. Even then this mechanic may lead to unpredictable situations in certain levels. But we believe that having a lot options to play around with will lead to a more enjoyable experience as long these options do not get in the way of each other.
Should the player be stuck in a level and cannot progress, the 'R' key can be pressed to restart the entire level. If the player wants undo one single move, pressing the 'U' key will achieve this.

1.2 Levels

Levels have been ordered the same way the three main gameplay mechanics have been added. So the first part of the game only includes levels in which the player can merely slide on. This way players can familiarize themselves with setting as well as controls. The number of these levels is rather low as there is too little variety to work with. After this introduction levels include dry tiles as well as finishes (dry or slippery) that require a specific side of the cube to be on the bottom/top in order to finish the level. Lastly, with the highest amount of levels, we add the unfolding mechanic. These levels have the most options to play with.

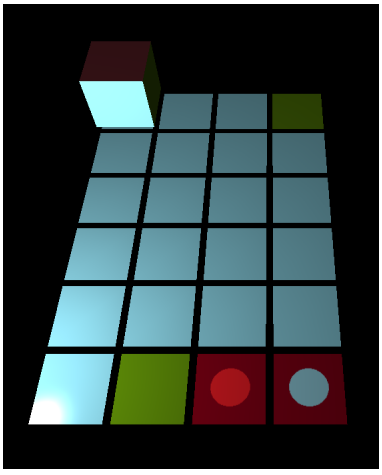


Figure 1: In order to beat this level the cube must be unfolded to get the red and blue segment to land on the corresponding finish tiles.

Designing such levels became a challenge as it was easy to lose a general view with the sheer amount of

options. As a result there is a good chance that players might find solutions in these levels that bypass most of the level in a way we did not predict. But as long as these solutions were found only using the provided gameplay mechanics, we do not see a problem in this.

This has led to a shift in terms of designing levels. Now the focus lies less in having levels maintain a steady difficulty curve. It is more about offering players a healthy amount of freedom to play around with all the gameplay mechanics. That being said we always make sure that levels are beatable.

1.3 Music

When thinking about the design of our game we had very specific criteria that we wanted our background music to fulfill. As we were designing a game based on spatial and geometric puzzles we wanted a slow and soothing soundtrack that does not distract the player and offers the space to think about the puzzles. This is also due our expectation to inhibit possible frustration that can occur in puzzle games and especially in puzzle games in a subject of math that is considered difficult for the general population. At the same time the music should capture an ethereal feeling, inspired by the soundtrack of the 2D platformer game "GRIS", which focuses on simple piano melodies in the early levels. The main loop for our music also uses a simple repeating eletronic piano melody that tries to convey a light feeling and is thus written to decrease distraction. The drum beat is oriented on typical Lo-Fi beats that are currently very popular as a background music for studying (LoFi to relax and study). The kick, hihat and clap for the drumbeat were chosen to provide a necessary amount of rhythmical texture for the otherwise repetetive melody loop. The general music loop is with 2 minutes playtime quite long but this allowed us to diversify the arrangement and spice up the composition with a sliding synth bass in between. The ethereal component of the music was achieved by using a modular synthesizer plugin that in our opinion fits very well into settings of dreamy scenes and are very well suited to create a background layer similar to the use of foley samples in modern producing. The melody from the modular synthesizer is a reoccuring part of the music loop but is also the theme for the not yet created main menu screen. As the music in the main menu it prepares the setting for the game and is due its lack of rhythm easily faded into the main loop that starts to play once the first level is loaded.

1.4 Sound effects

Sound effects are very important in games as they can provide audio feedback that let the player distinguish their interactions with the game mechanics. In qubi sound effects mostly fulfill this role but there is also an addition to give an insight for the activation of an end tile which is used as a tutorial tool. The rest of the sound effects like sliding and the knock that is played when the cube flips work like conformations similar to tactile feedback for keyboards but in the audio world. Sound effects can also affect how humans interpret the displayed scene. In the first iterations of the flip sound effect the frequency spectrum of the sample was considerably lower which made the cube appear heavier then what we had imagined for its feel in the game world. This was adjusted by decreasing the lower frequencies and boosting the mids by using a multi-band compressor plugin. Something to note is that it was surprisingly difficult to find a sound sample for the slide that was also able to loop indefinitely and still had the satisfying sound effect that we imagined during the playtesting of the physical prototype.

2 Implementation changes

The first step towards the implementation of the undolding mechanic, was splitting the cube into its 6 segments. Until then we just used a primitive cube mesh, but to allow for unfolding, we made a single

mesh for a cube segment and instantiate it for each of the cube sides, with the different materials. This of course made the movement logic a bit more complicated as now all segments have to be moved in sync. This is archieved, by parenting them all to an empty object, animating the empty object and then unparenting them again. While parenting and unparenting we calculate the new local transformation, so the object does not move in world spce when parenting or unparenting. Furthermore the parenting, animation and unparenting are scheduled as soon as the user presses a button, to avoid any simulation inaccuracies during the animation.

With these capabilities in place we could start to implement the unfolding. When unfolding, instead of flipping the whole cube, only flip the active segments and every segment that faces downwards becomes passive. Like this it became possible to unfold the cube. Of course we had to limit the directions that can be unfolded to, as not all directions are always possible, limited by the geometry of the cube. A possible unfolding configuration can be seen in Figure 2.

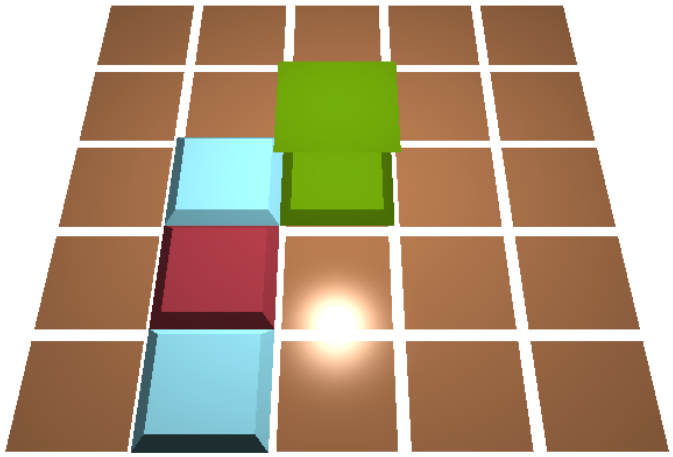


Figure 2: The resulting unfolding configuration following the input directions Up - Up - Right starting from the (0, 1) tile

To make the cube be able to refold, we additionally store the passive segments in a list, ordered by the time they became passive. So when it comes time to refold, we refold the segments in the same order, for them to reach the "head" position – the position where the player stopped the unfolding process. The refolding then works by first making all segments passive except the first placed segement. Determine in which direction it should flip, depending on where the next places segment is, because it should fold onto that. Then create an empty object at the appropriate edge, parent the segment to it, animate the empty to fold the segment, and unparent the segment again. This process continues, always additionally making the next segment active so all segments fold back together to the cube. Also all these actions and animations are scheduled as soon as the player decides to refold the cube, again to be able to produce a reliable and exact animation.

We discovered, that for some legal unfolding schemes, the refolding alogithm would produce an animation, in which geometry of the cube would penetrate itself. We found six unfolding configurations (disregarding all symmetries) in which an impossible refolding is generated. We then wrote code to detect each of the six cases and then add more animations to the chain for each case in such a way, that the geometry does not penetrate itself anymore. An example refolding which would penetrate itself, together with the

handwritten fix can be seen in Figure 3. The challenge here was to be able to detect the configurations and all ther symmetries in all directions reliably and also make the additional handwritten animations work for all of them.

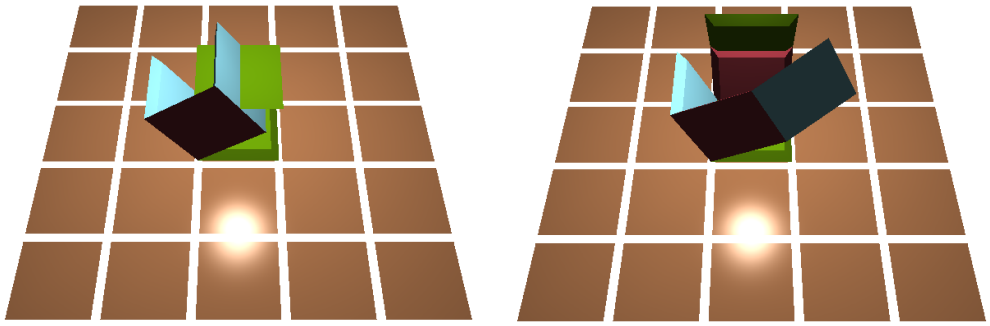


Figure 3: Left: The animation the default refolding algorithm produces can lead to geometry penetrating another part of the cube. Right: The handwritten animation for this specific unfolding scheme opens up the cube, so the segments do not collide

With unfolding and refolding in place we could then implement levels with multiple finishes onto which the player will have to unfold to activate them all. We extended the already existing `finish_check` routine to iterate over all the finishes in the level and check for each if the finish condition is satisfied. We also moved away from storing levels in a hashmap with their name as the key, as we want the levels to be loaded in order as the player plays through them. So now we just store them in an array list.

3 Engine Progress - Sound

We think that sound, especially in puzzle games is a major contributor to the overall feel of the game. As mentioned in the other project notebook chapters (e.g. Physical Prototype) one major focus for our development was to capture the correct feeling of our vision as succinctly as possible so implementing a sound system was a high priority after the initial engine was work finished (even thought we forgot to add it to the layers of development in the first chapter).

3.1 irrKlang Sound Engine

For the sound eninge in this project we chose to use the irrKlang sound library. IrrKlang is a object oriented interface designed for games and supports all current sound formats such as .wav .flac and .mp3. It is able to run on Linux and Windows which was an important aspect to consider to ensure our cross platform compability.

3.2 Additional work

Most of the basic function that irrKlang provides were enough for this game because our use for now will not exceed simple sound effects and background music. The only work on top of the irrKlang functions that was needed was the timing of the effects using the scheduling system of the engine. One additional interesting fact to note was the possibility to skip the use of a random number generator for choosing a random sound effect. We thought that this would be necessary as playing the same sound effect repeatedly for an action such as sliding is very repetetive. Instead we tried to use a longer sound sample

of a slide and continously loop through it which worked much better than we initially expected and sounds realistic and varied. A function to chose a random sample from a group of will still be necessary for the future but the possibility of implementing the sliding sound effect in such a way is something to take not of especially due to the fickleness of sound loops in general.

4 Outlook

Regrettably we are still behind in terms of visuals as these are still on a rudimentary level. This means while we have a stable enough version of qubi that we could use for playtesting, we will have to focus on working on the visuals in the meantime as well. We will further have to add a UI system to provide an options menu.

Basic instructions on how to play the game will have to be added as well as an indicator showing which level the player is in.