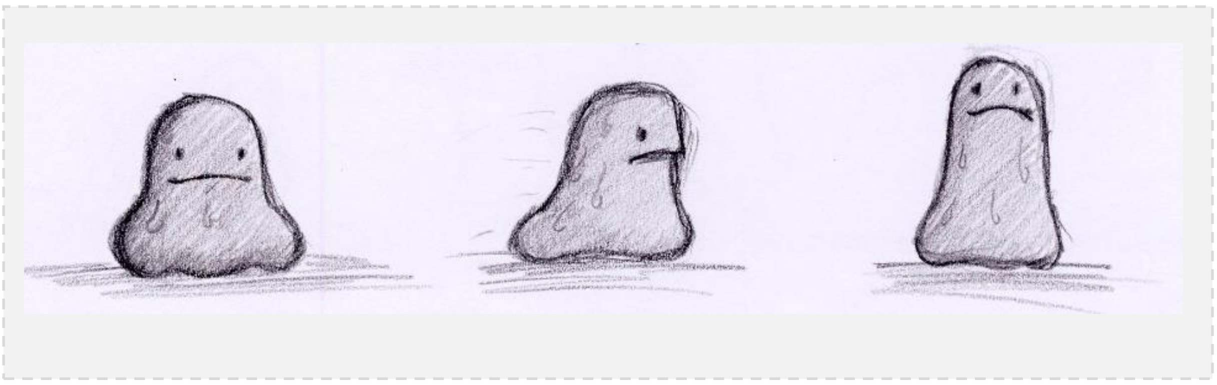
TEAM 5

Game Programming Laboratory

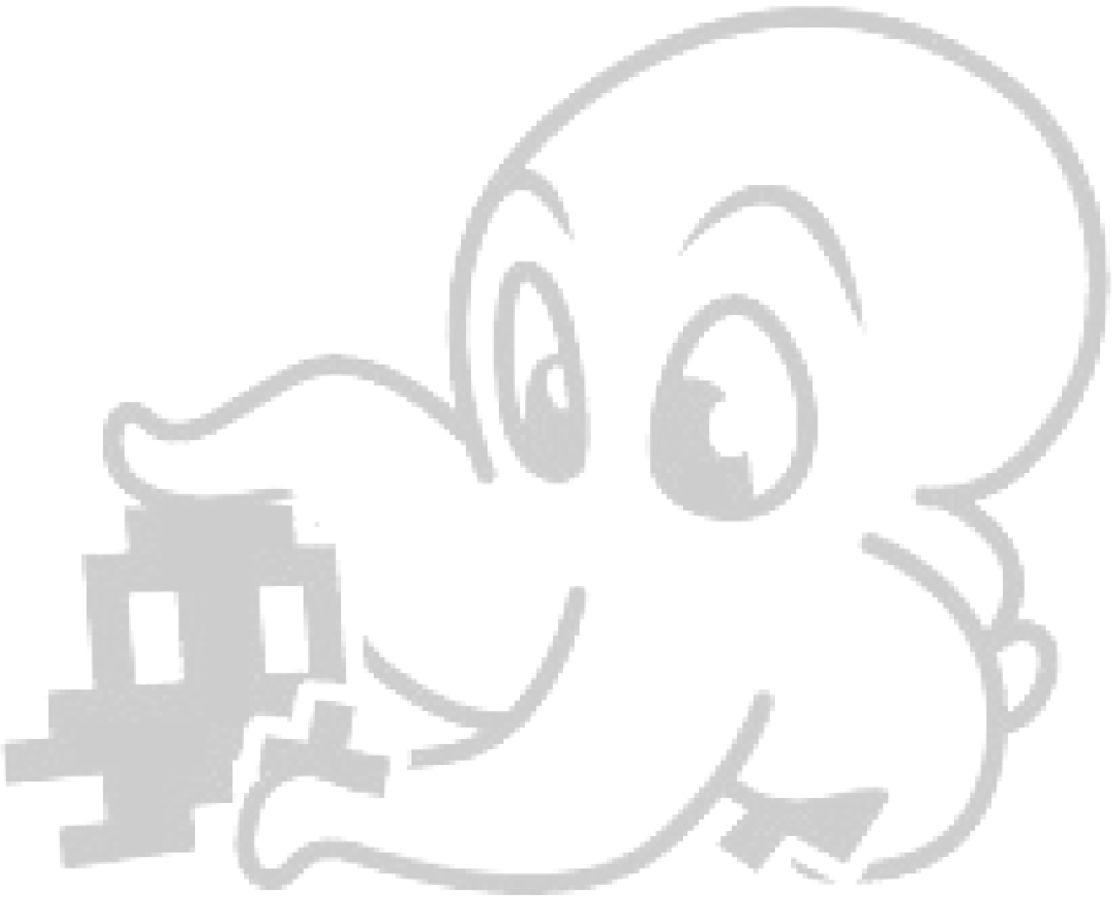
THE LEGEND OF BLURP

The Adventures of a Slime Blob



Virginia Ramp – PRODUCER, PROGRAMMER, DESIGNER, VISUAL ARTIST
Julian Collazo – PROGRAMMER, DESIGNER, SOUND ARTIST
Pascal Stoop – PROGRAMMER, DESIGNER, LEVEL DESIGNER
Markus Roth – PROGRAMMER, DESIGNER, QUALITY ASSURANCE

Contents



CHAPTER 1. FORMAL PROJECT PROPOSAL

1.1. GAME DESCRIPTION

1.1.1. OVERVIEW

"The Legend of Blurp" is a 2D puzzle platformer that can be played both in single-player and cooperative multiplayer. Additionally, a competitive multiplayer mode might be added, where players will compete to be quicker, cleverer or more skilled than their opponents.
The main character, Blurp, is a small slime blob with consciousness. It can split his own body, e.g. by spitting slime at his opponents, and can merge with slime puddles that lie around.
The main goal of the game will be to solve the puzzles in each level and complete the Jump n' Run challenges. Puzzles can be solved by using Blurp's ability to split and merge his body mass, which can then be used to activate buttons and other game elements.

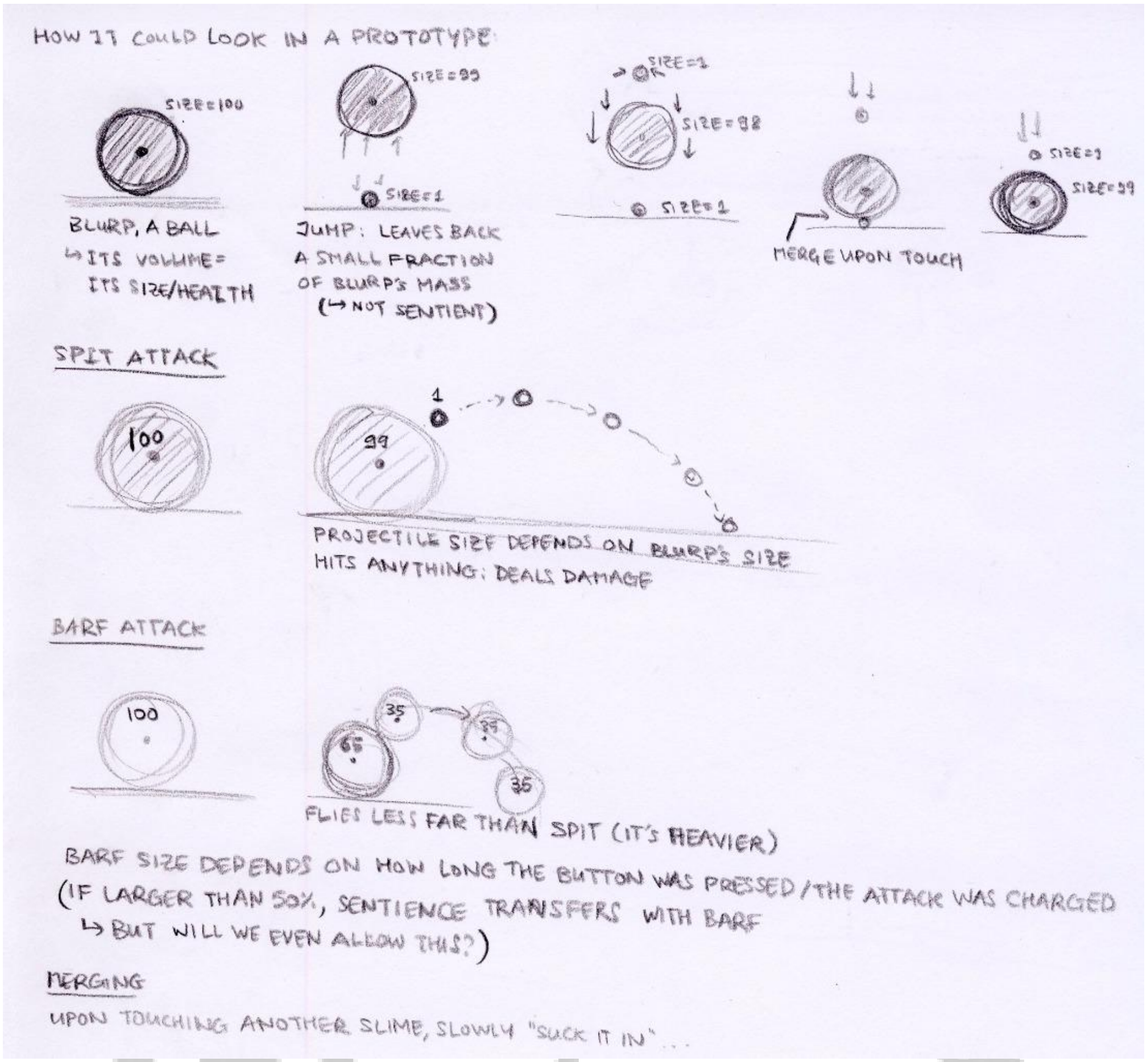
1.1.2. BACKGROUND STORY

(Cutscene / Intro)
In the year 1850, Alfred Escher wanted to make the young federal state of Switzerland the most advanced scientific country in the world. He gathered the best scientists in the world in a secret laboratory deep in the Swiss Alps. The main purpose of the lab was to invent new intelligent life forms that could fulfil practical tasks like digging tunnels.
So, the scientists set upon their work, scribbling down formulae, and handing them out to their assistants, which were mostly robots. The few humans that were there nodded appreciatively. They knew what they had to do.
They started sawing, milling, brazing and welding, mixing together bubbly liquids of all colors, while Escher stood and watched them, a hint of a smile visible through his beard.
Finally, the big moment came. They put everything in a chamber and closed the door. A huge laser fired a blinding blast of energy at it.
They open the door, full of excitement. The suspense is rising, as the smoke slowly clears away. And they see… A grey blob of slime.
They sigh. All tension released, they know they have failed to create what was originally planned.
They scoop up the blob of slime and put it into a trash bin, from where it is transported to, and dumped on the closest waste disposal site.
(Game Storyline)
What the scientists didn't know, is that their seemingly disappointing blob of slime was actually a highly complex life form which they have just accidentally created. We will call this life form "Blurp".
Blurp then goes on his quest to find out how he came to be. He explores the world while finding his path back to the laboratory, to his creators.
Only through many adventures and only by proving his intellect, does he manage to find his way back into the secret lab. On his way he has to surpass many security systems and fight robots that patrol the aisles. Coming in contact with different chemicals that happen to be in the lab, Blurp learns to integrate these chemicals into his body and use them to advance further on to the heart of the laboratory.
The researchers obviously notice that their robots are being destroyed by someone and increase their security. When Blurp manages to come close enough to observe them, he sees that they go to great lengths to try to create new life forms, and doesn't approve of their approaches.

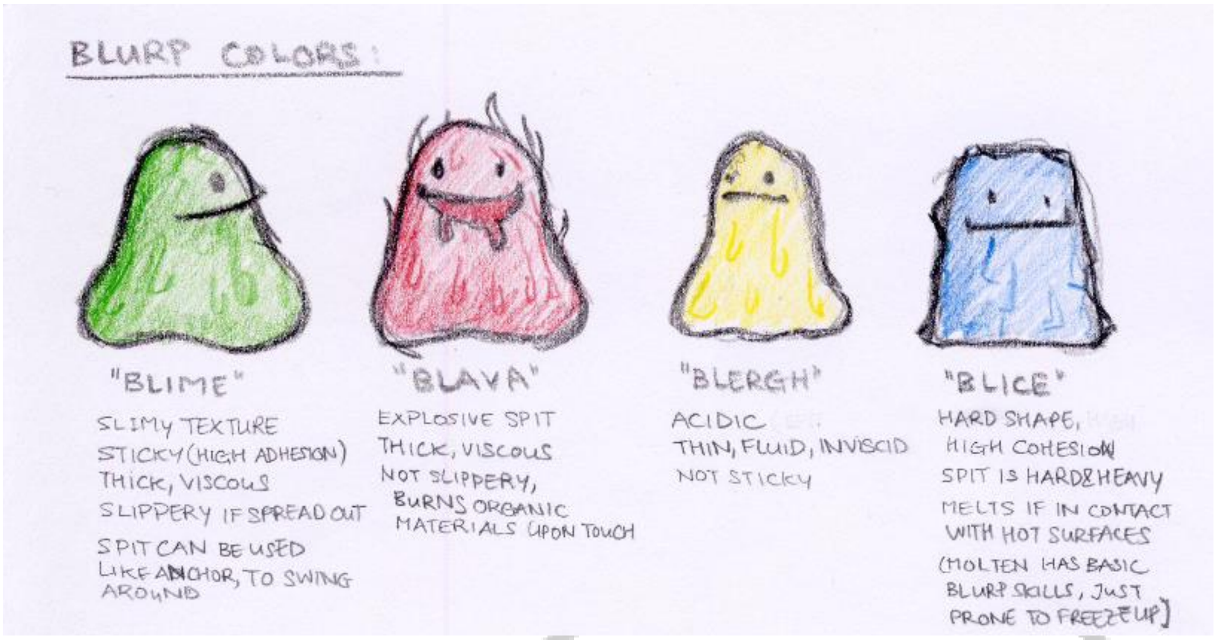
Blurp then manages to take out all robots, destroy the lab and accidentally kills a few of the researchers that try to catch and later kill him. Escher manages to flee.
(Cutscene / End of Game)
The accident was covered up, Escher himself turned to more political and educational matters and decided that it would be too early to bring any of their previous inventions to the world. He secured the lab, locked it tight, and initiated the self-destruction sequences specifically prepared for such an event.
Blurp himself toyed around with a few of the machines and hopped into something that was labeled "prototype time-space matter transmitter". He pushes a button and vanishes. A few minutes later, the lab explodes due to the self-destruction mechanism.
(Text / Closing Credits)
After the incident, the remaining scientists wanted to continue researching, which is why Escher decided to found the Polytechnikum. The other universities didn't specialize on these engineering subjects, and after their recent failure, it was ever so more important to Escher to push Switzerland's technological advancements forward.

1.1.3. Design Decisions

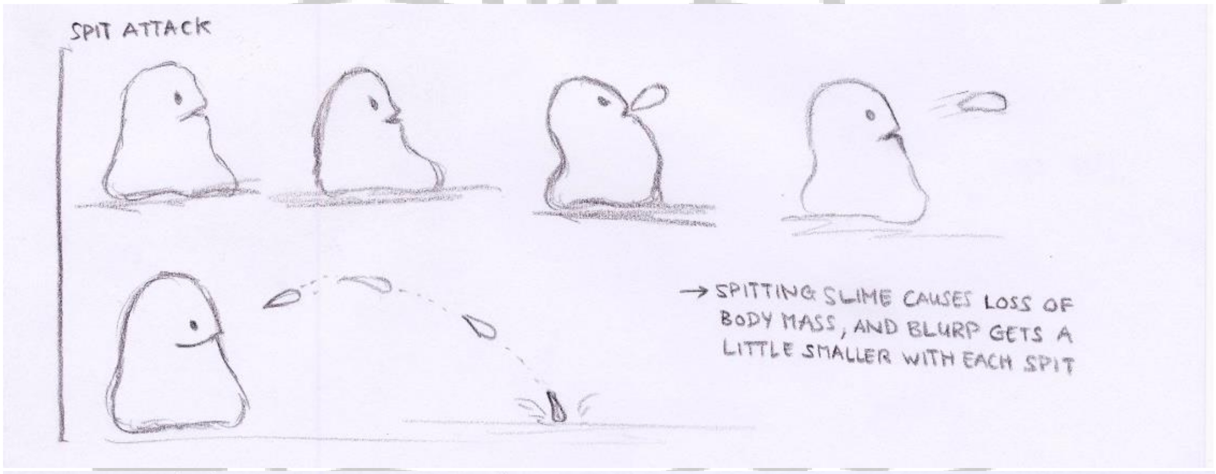
The core mechanics will be based on rigid body physics plus our slime model. We want the game to behave in a physically plausible way, so objects will be affected by some forces like gravity, and impulse, mass, torque and friction.
All our fix game objects (walls, for example) will be tile based, such that they can be easily placed in a level editor. They will mostly be squares, but some fix slopes at 45° and 22.5° angles will likely be added to make the environment less blocky.
Our game will have interaction objects like buttons, boxes to push around, seesaws that tilt according to how much weight is put on it, etc. They will correspond to our rigid body physics.
As a plus, we will have environment elements, which are forces like wind, areas with different gravity, etc. which will influence all movable objects. Gravitation itself is such an environment element.
Our playable character will be a slime blob with the ability to split and merge its body. The player can steer whichever slime blob is deemed large enough to contain "consciousness". In single-player only one blob can be controlled at once, but the player might get the ability to switch between these blobs.
For cooperative multiplayer, the main slime blob will split in equal halves and each player can control one of them. All large enough slime chunks can be chosen to play, as long as no other player controls them already.
The decision on which blob is large enough will depend on each level, and on how much slime is available in said level.
A player will die if the slime gets split up in such a way, that all separate blobs are too small to be played. In that case, the game will be reset to the last checkpoint.
The slime blob will be portrayed by a simple sphere in early iterations of the game. Later on, we want to switch to a more complex, particle-based model.

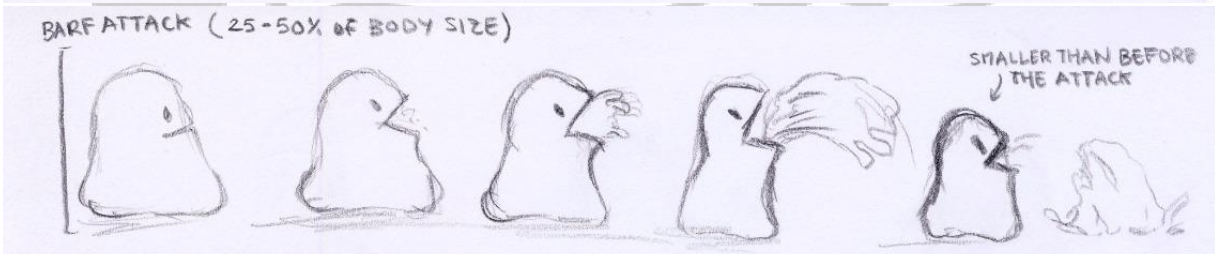

HOW IT COULD LOOK IN A PROTOTYPE:
SIZE=100
BLURP, A BALL
↳ ITS VOLUME = ITS SIZE/HEALTH
SIZE=99
SIZE=1
JUMP: LEAVES BACK A SMALL FRACTION OF BLURP'S MASS
(↳ NOT SENTIENT)
SIZE=1
SIZE=98
SIZE=1
MERGE UPON TOUCH
SIZE=1
SIZE=99
SPIT ATTACK
100
99
1
PROJECTILE SIZE DEPENDS ON BLURP'S SIZE
HITS ANYTHING; DEALS DAMAGE
BARF ATTACK
100
65
35
35
35
FLIES LESS FAR THAN SPIT (IT'S HEAVIER)
BARF SIZE DEPENDS ON HOW LONG THE BUTTON WAS PRESSED / THE ATTACK WAS CHARGED
(IF LARGER THAN 50%, SENTIENCE TRANSFERS WITH BARF
↳ BUT WILL WE EVEN ALLOW THIS?)
MERGING
UPON TOUCHING ANOTHER SLIME, SLOWLY "SUCK IT IN"...

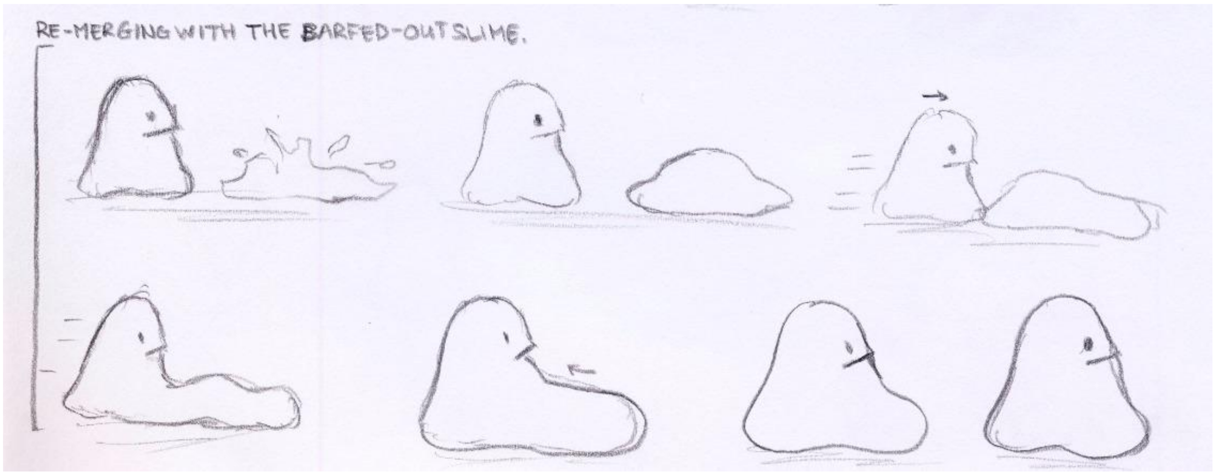
The slime can also take on some abilities from different materials which he encounters along the way. For example, by dousing himself in acid, he might become more liquid, but he also gets the ability to etch materials like wood (i.e. destroy them to open up new passages). Or he can become stickier and gain the ability to slowly climb walls, and even stick to ceiling.



Our slime will have two basic attacks, which can also be used to solve puzzles. In the spit attack, he spits out a small portion of his mass which might injure enemies, and in the more powerful barf attack, he spits out a large chunk of his own mass to, depending on his abilities, glue enemies to the ground, burn them, etch them, or make them lose their footing.
Since these attacks split off mass from the slime's main body, the resulting masses can be used to keep buttons activated or balance out some weights on a seesaw.


SPIT ATTACK
→ SPITTING SLIME CAUSES LOSS OF BODY MASS, AND BLURP GETS A LITTLE SMALLER WITH EACH SPIT


BARF ATTACK (25-50% of BODY SIZE)
SMALLER THAN BEFORE THE ATTACK

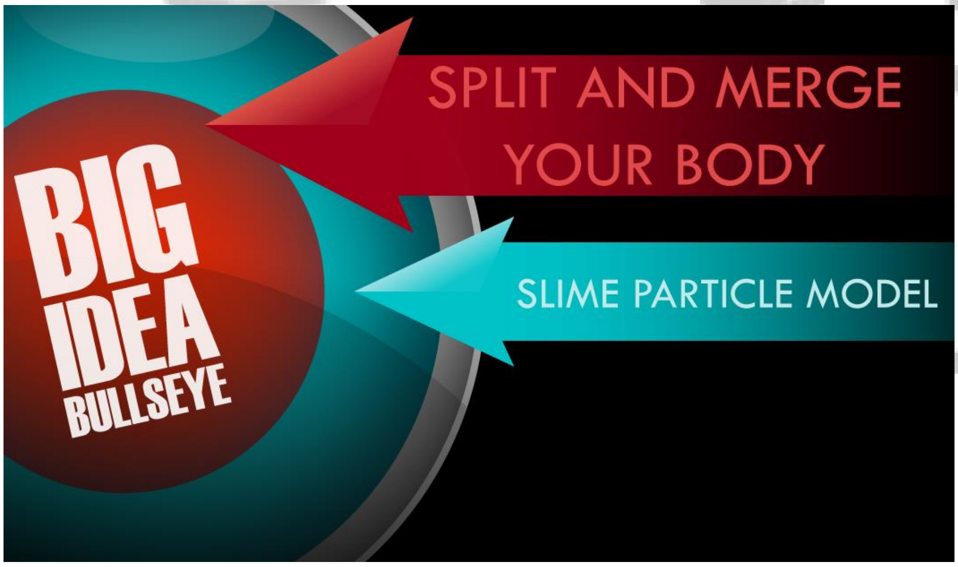


For the visual style we want to go for a colorful look, with some comic-like apparel. We intend to use the parallax 2D style, i.e. we have different layers moving at different speeds to create an illusion of depth.

1.1.4. Inspirations

Our game is inspired by famous titles like *Limbo* (for the puzzle aspect), *Ori and the Blind Forest* (for the platformer/Jump n' Run aspect), *World of Goo* (for the slime aspect) and *Osmos* (for the splitting and merging aspect).

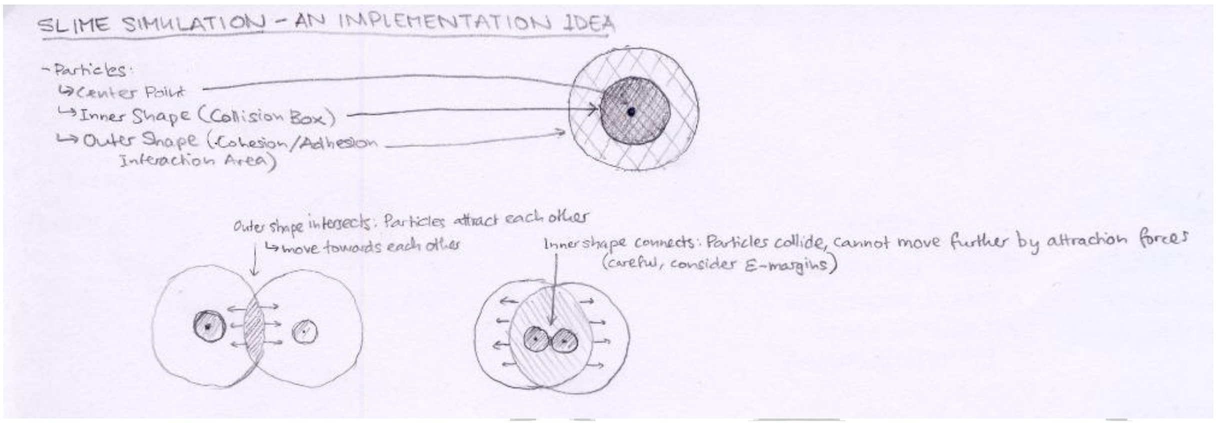
1.2. ‚Big Idea' Bullseye

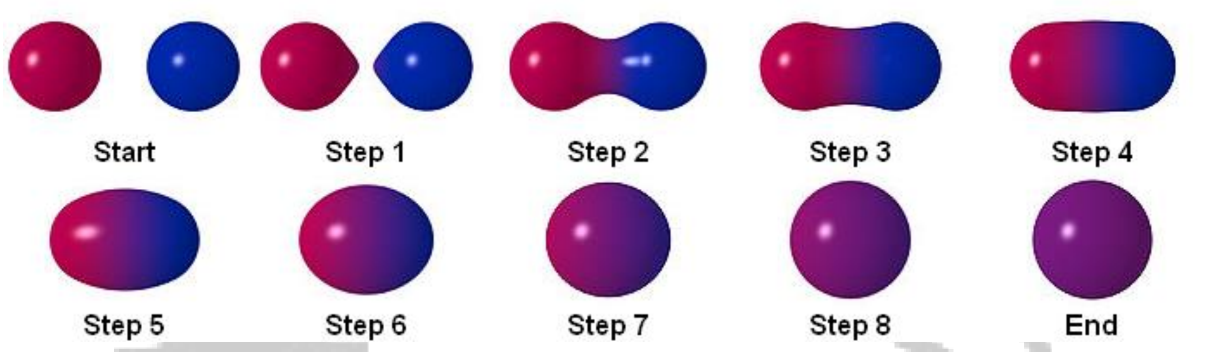


1.3. Technical Achievement

Our technical achievement will be the realization of the slime mechanics with a particle model. The particles will interact with each other based on their distance to each other. If their centers are very close together, they will repel each other (such that they don't collapse to one point). If they are within a certain distance to each other, they will attract each other (i.e. stick together). And if they surpass a certain distance, they will not influence each other.

The convex hull of these particles will then be surrounded by an implicit surface, and we are going to study metaballs to achieve that.
We are planning to define the areas of influence (repelling and attracting area) of the particles by circles to simplify collision handling and calculations.
The slime can then be controlled by controlling a fixed number of particles and letting the other particles move with them according to the stickiness of the material. The best way to do that will still be analyzed by us.


SLIME SIMULATION - AN IMPLEMENTATION IDEA
- Particles:
↳ Center Point
↳ Inner Shape (Collision Box)
↳ Outer Shape (Cohesion/Adhesion Interaction Area)
Outer shape intersects: Particles attract each other
↳ move towards each other
Inner shape connects: Particles collide, cannot move further by attraction forces
(careful, consider E-margins)



Metaballs. Image Source: Wikipedia

1.4. Development Schedule

1.4.1. Layered Task Breakdown

1.4.1.1. Functional Minimum

- Prototype sprites (spheres and cubes of different color)
- Basic physics for rigid body simulation
- Collision handling
- Simple slime prototype (not particle based yet) that can split and merge and has mass
- Some very basic interaction objects (button, door, box, (seesaw, depending on physics))
- Very very basic UI
- Basic camera that moves with the player
- One hard-coded level to test out physics, collisions, interactions

1.4.1.2. Low Target

- Some game art (backgrounds -> parallax 2.5D effect, sprites for interaction objects)
- Crude level design tool (just for us, at that point)

- Some first levels
- Simple enemy and basic fight mechanics
- More environment and interaction building blocks

1.4.1.3. Desired Target

- Game Art for pretty much every game element
- Working slime particle model (slime has no face, but controlling the slime works).
- Several levels that provide a good learning curve
- Cut-scenes (simple animations or just images) to tell the story (at least for the intro)
- Basic sound design (music for levels and sound effects for slime, interaction elements and enemies)
- Smooth camera motion
- Better enemy logic

1.4.1.4. High Target

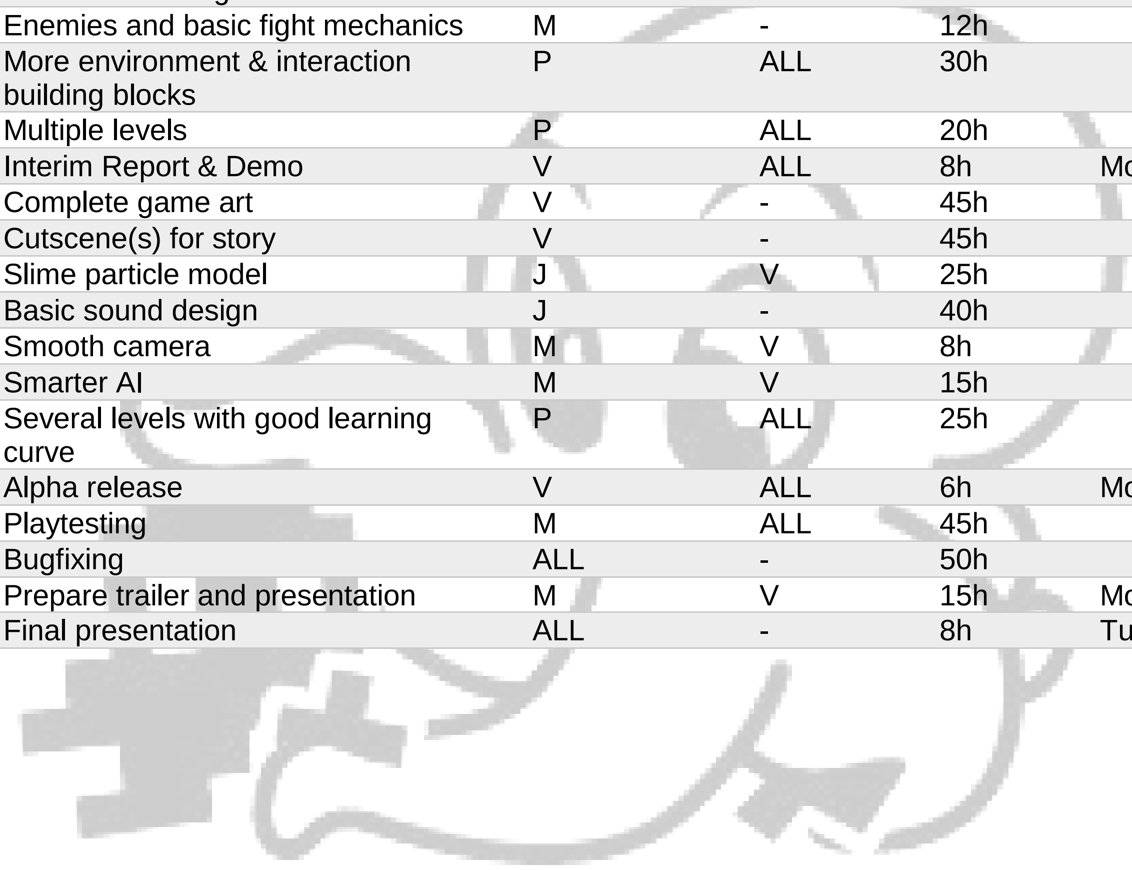
- Aesthetic slime looks with implicit surface/metaball creating clean surfaces
- Different slime colors/attributes
- Different level settings that tell the story
- More versatile sound design (more soundtracks, better sound effects)
- Dynamic split screen in multiplayer mode
- Including the level editor in the game (for players to use and create levels)
- Co-op Multiplayer mode and levels for it

1.4.1.5. Extras

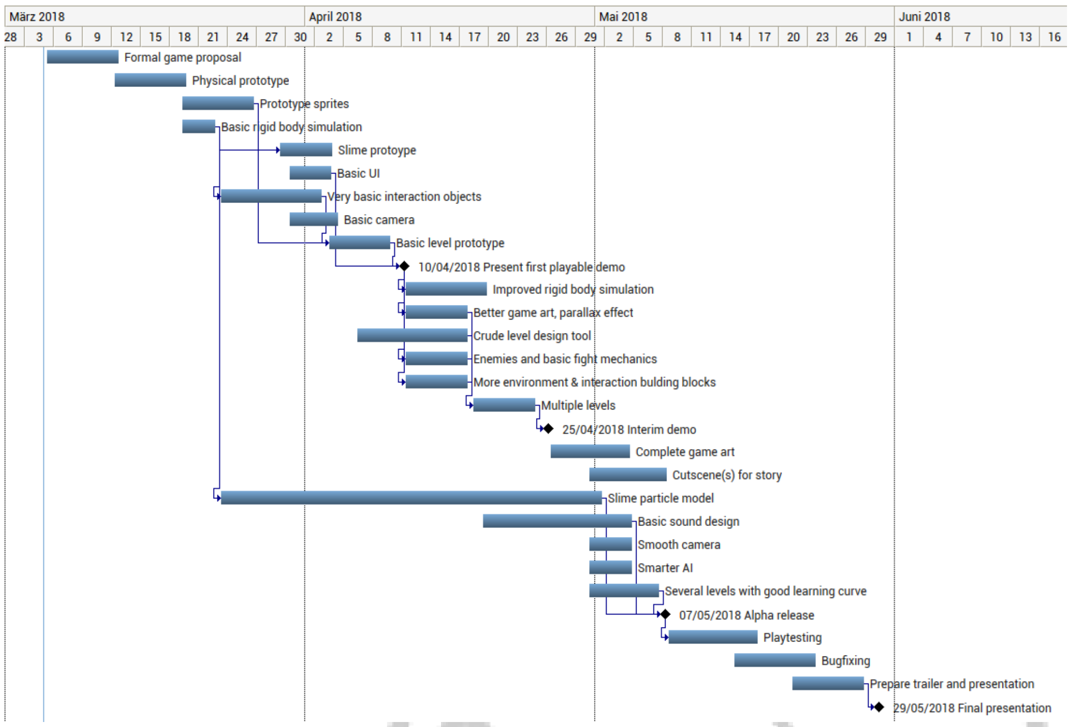
- Cute slime face that fits with the slime itself and makes Blurp more relatable to the players
- PvP Multiplayer Mode and levels for it
- Finished story arc (levels that work well from start to end) with cut scenes in the outro
- Cut scenes with spoken text (by a professional voice actor (Morgan Freeman?))
- Top-notch sound design (orchestral, including cooperation with Hans Zimmer and Jeremy Soule)
- Publish the game

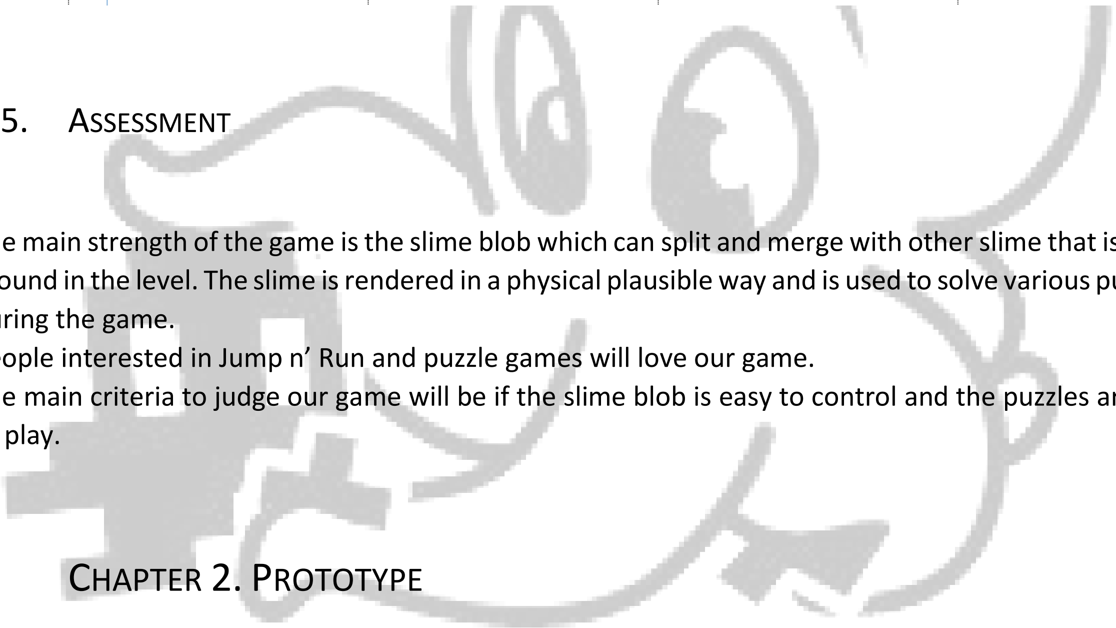
1.4.2. Task List

Task	Responsible	Team	Duration	Latest deadline
Formal game proposal	V, M	ALL	8h	Mo, 12.03.
Physical prototype	M	ALL	7h	Mo, 19.03.
Prototype sprites	V	-	3h	
Basic rigid body simulation & collision detection	J	-	12h	
Slime prototype	M	V	10h	
Basic UI (Menu, Settings, ...)	M	V	5h	
Very basic interaction objects	P	V, M	16h	
Basic camera	M	V, P	4h	
Basic level prototype	P	-	6h	
Present first playable demo	V	ALL	4h	Tue, 10.4.
Improved rigid body simulation	J	-	25h	
Better game art, parallax effect	V	M	20h	
Crude level design tool	P	-	15h	
Enemies and basic fight mechanics	M	-	12h	
More environment & interaction building blocks	P	ALL	30h	
Multiple levels	P	ALL	20h	
Interim Report & Demo	V	ALL	8h	Mo, 23.4.
Complete game art	V	-	45h	
Cutscene(s) for story	V	-	45h	
Slime particle model	J	V	25h	
Basic sound design	J	-	40h	
Smooth camera	M	V	8h	
Smarter AI	M	V	15h	
Several levels with good learning curve	P	ALL	25h	
Alpha release	V	ALL	6h	Mo, 7.5.
Playtesting	M	ALL	45h	
Bugfixing	ALL	-	50h	
Prepare trailer and presentation	M	V	15h	Mo, 28.5.
Final presentation	ALL	-	8h	Tue, 29.5.



1.4.3. Timeline





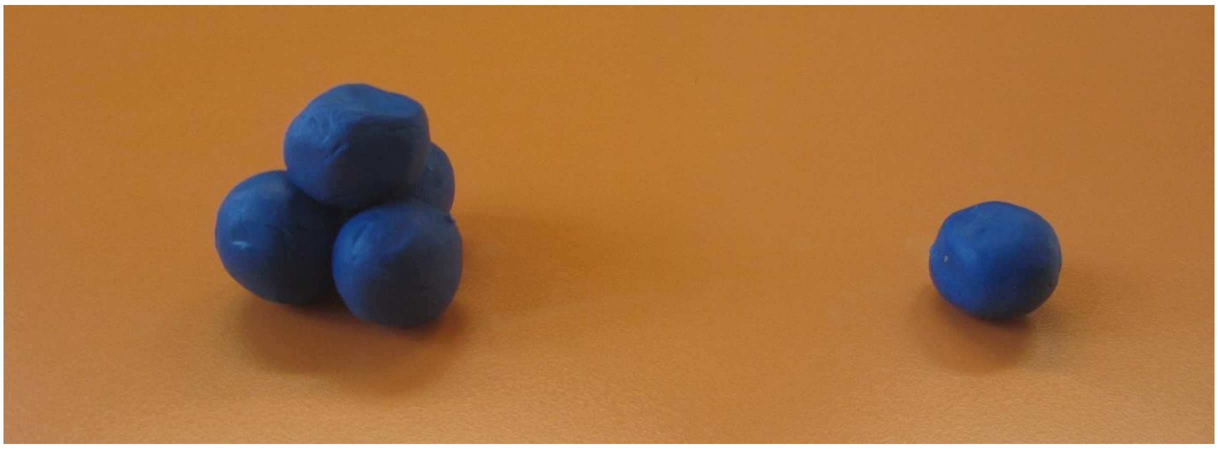
1.5. Assessment

The main strength of the game is the slime blob which can split and merge with other slime that is lying around in the level. The slime is rendered in a physical plausible way and is used to solve various puzzles during the game.
People interested in Jump n' Run and puzzle games will love our game.
The main criteria to judge our game will be if the slime blob is easy to control and the puzzles are fun to play.

Chapter 2. Prototype

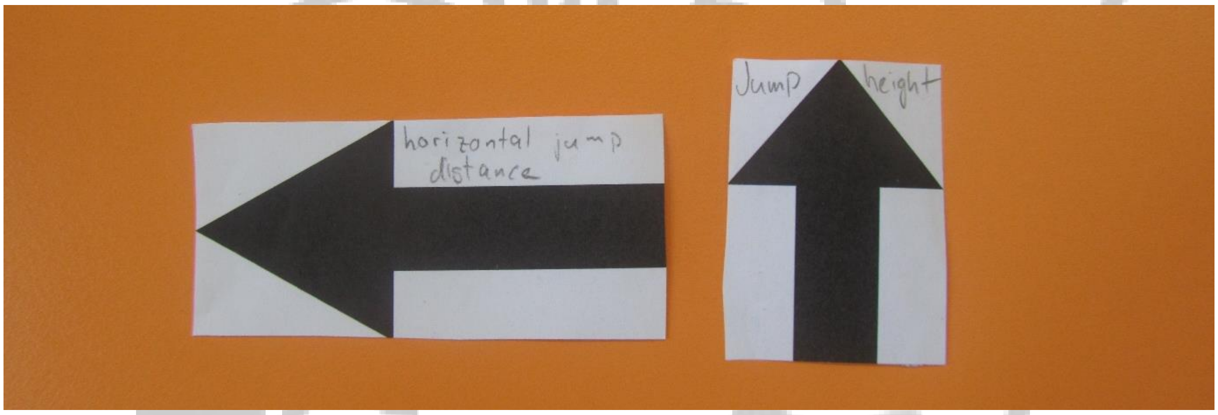
2.1. Prototype Setup

We modelled the slime with modelling clay. To make the split and merging easy and intuitive, we defined a lowest playable weight unit that consisted of one clay ball.



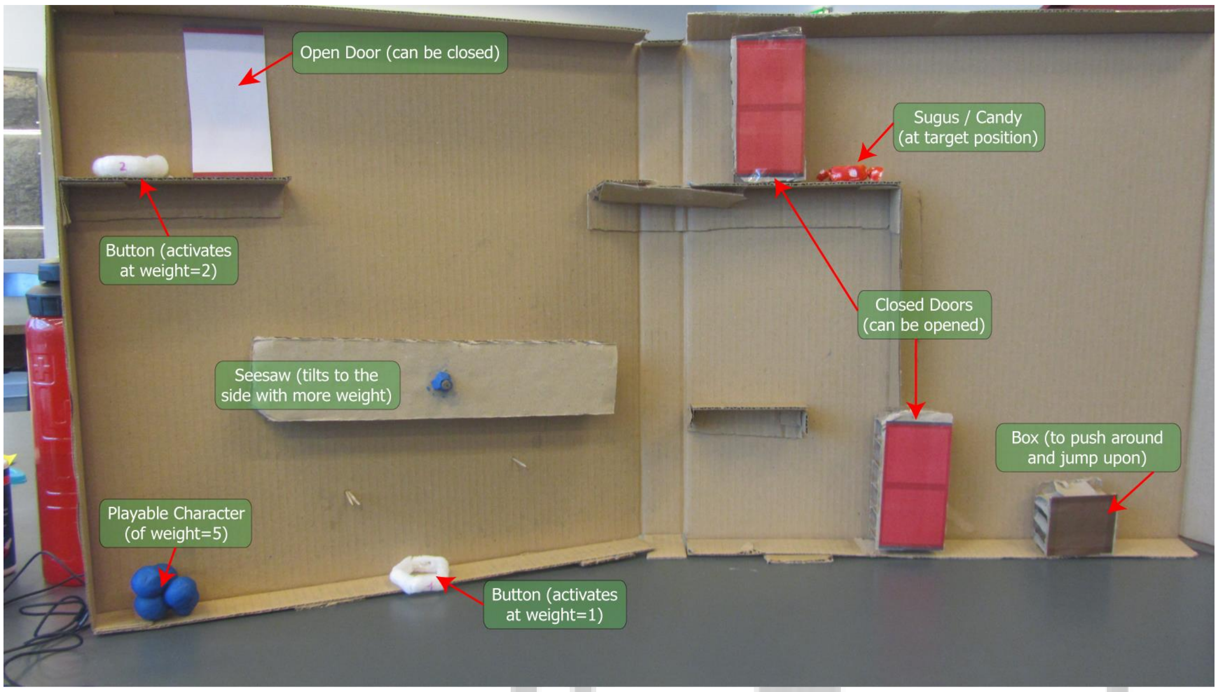
In the beginning the playable slime consists of five balls. During the game, the player can split the body into several parts. For simplicity, we allow our play testers to split off any amount of balls from the main blob without limiting rules like “spit attacks split off exactly one ball”. In the game, the splitting mechanics will follow clear rules, but the player himself doesn’t need to know them when he plays the game, which is why we don’t introduce them in the prototype.
If a body is split into multiple parts, only one of them can be played at once. The player can freely switch between the different parts.

We define a maximal height the player can jump and a maximal horizontal jump distance. To indicate these limits, we use arrows printed out on paper.

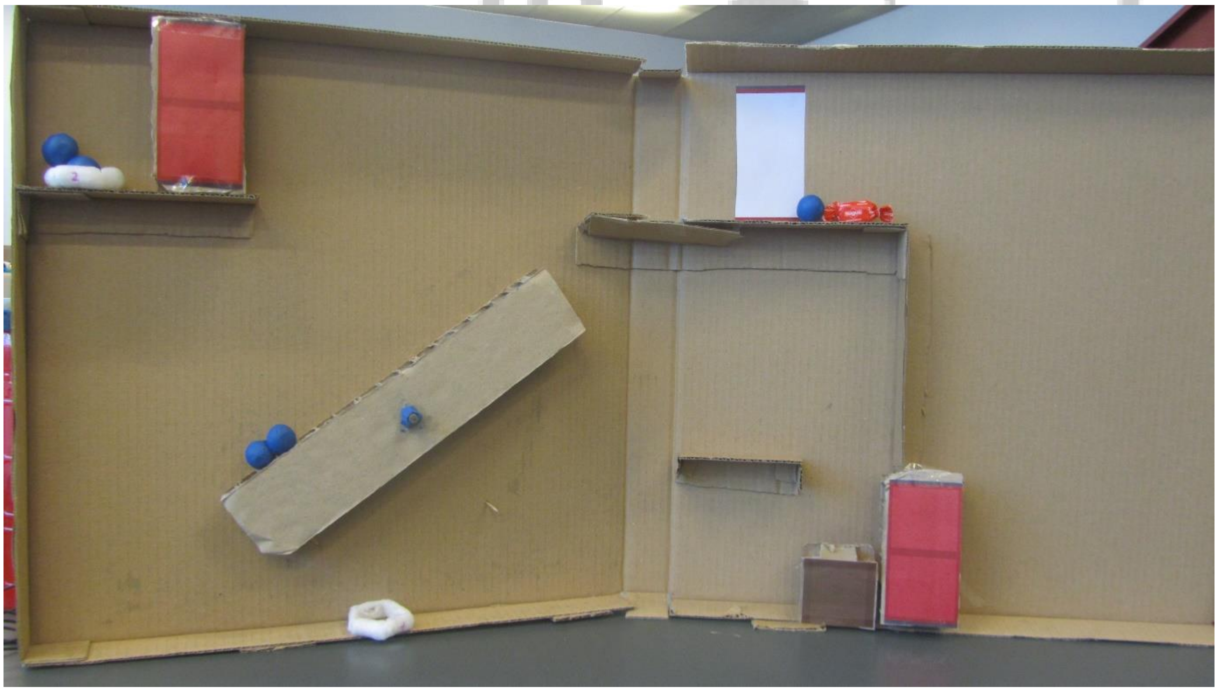


The level itself is modeled with cardboard. It consists of following interactive objects
- starting position
- target position (Sugus/Candy)
- buttons that need a certain weight to get activated
- doors that open when it's corresponding button is activated
- a movable box

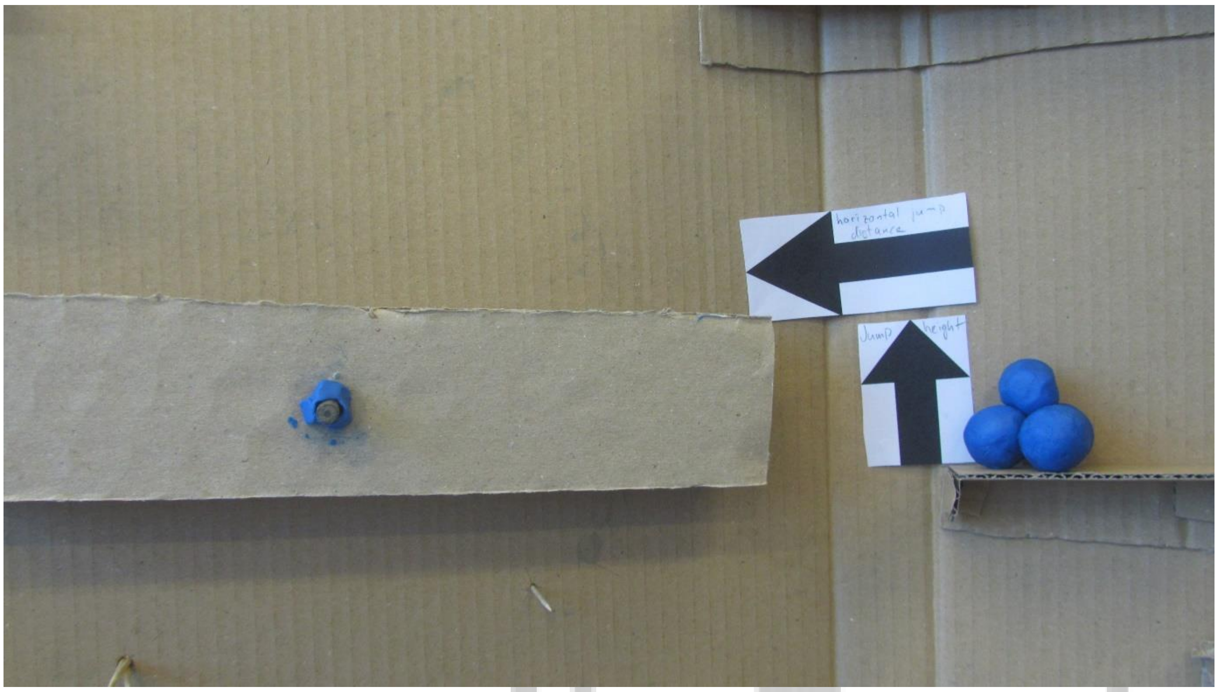
The goal of for the player is to reach the target position (indicated by a Sugus, i.e. a piece of candy) on top of the level.


Open Door (can be closed)
Sugus / Candy (at target position)
Button (activates at weight=2)
Closed Doors (can be opened)
Seesaw (tilts to the side with more weight)
Box (to push around and jump upon)
Playable Character (of weight=5)
Button (activates at weight=1)

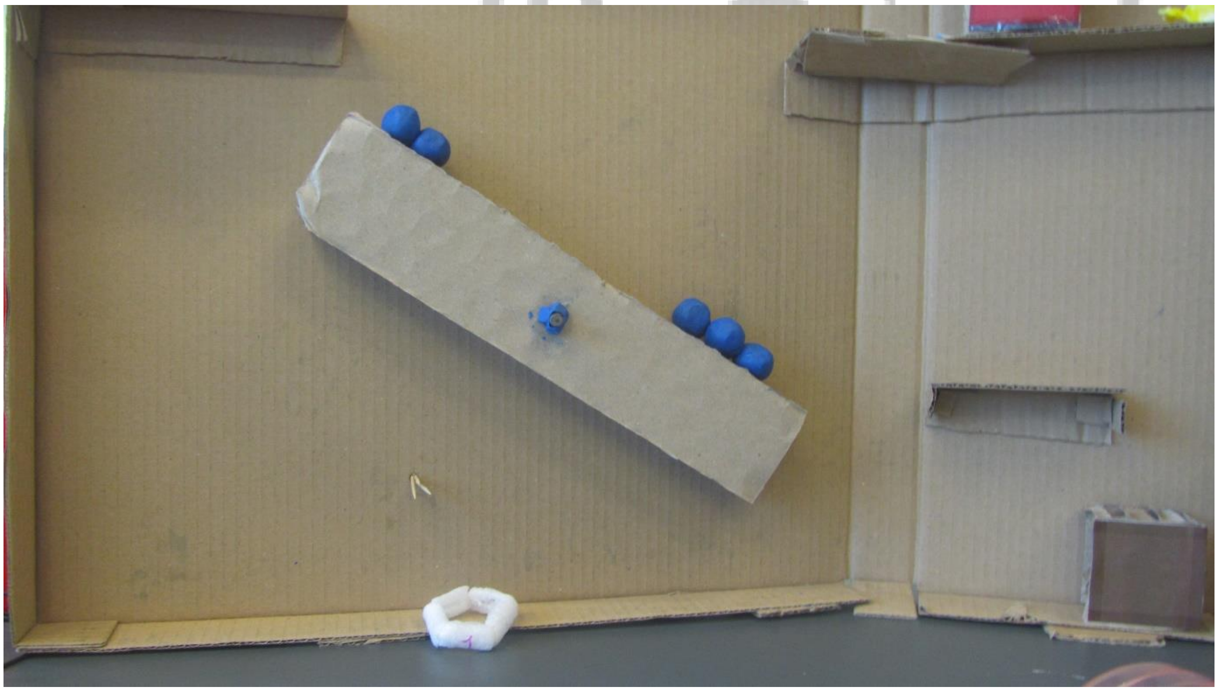
The level in its initial state. The top button activates both doors on the top floor. The bottom button opens the door at the bottom.



The level in its final state after it was solved.



The jumping distance, visualized with the arrows, to show that a jump from that platform to the seesaw is possible.



The seesaw, tilting to the side which has more mass on it. If both sides have an equal mass on it, the seesaw will be balanced.

2.2. Playing Experience

We asked three people to play our game. They liked it and could solve our prototype level in around five minutes.
We ourselves also had fun playing our prototype, even though the cardboard prototype was a bit shaky.

2.3. Findings and Conclusion

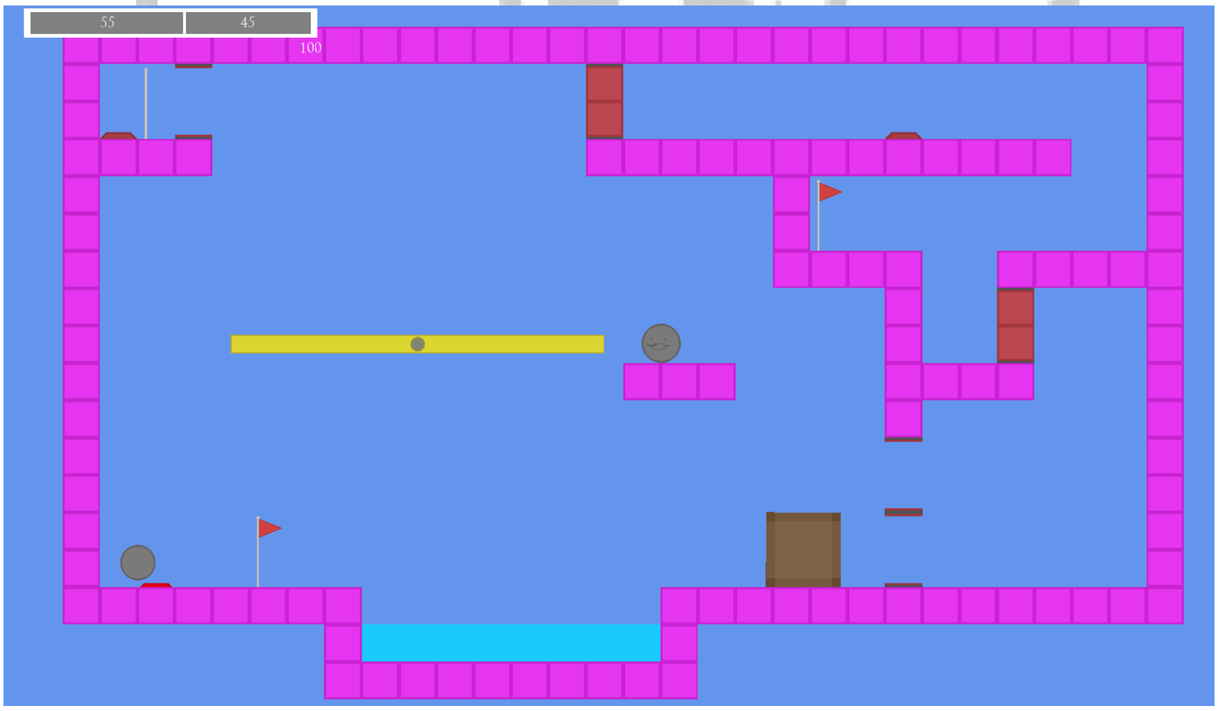
We found out that our physical prototype level was too complex for new players. In our final game we will need to introduce the game elements gradually for a good learning curve.
Designing a good level isn't straightforward. Even in our small prototype, players found multiple ways to solve it, using not all the provided game mechanics. To make sure that our test players would use all core game mechanics (splitting, merging, pushing boxes, switching between blobs, and buttons with a weight-activation), we had to introduce more doors and buttons to our final prototype level.
After playing our prototype for the first time, we decided to fix the jump height independent of the blob's body mass. This makes designing and playing levels more consistent.

Chapter 3. Interim Report

3.1. Progress

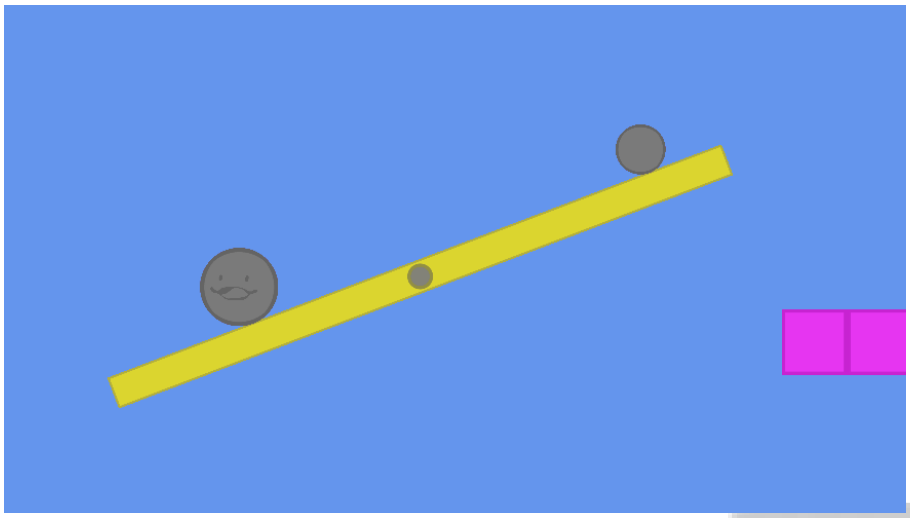
We completed all points from the functional minimum. In the low target we still have two missing points, one is the parallax effect (postponed due to missing game art) and the enemies.

We implemented a lot of new objects in the game such as boxes, buttons, seesaw, checkpoints, and water. We also have a temporary slime model that can merge and split itself.

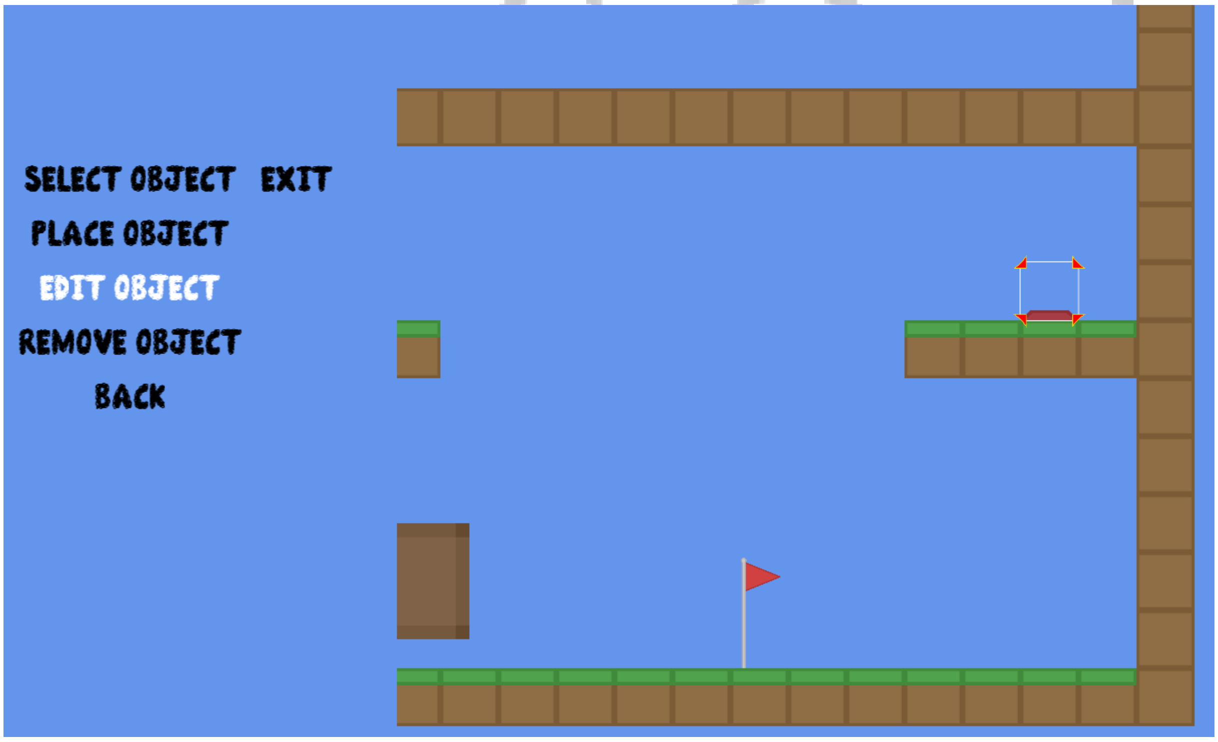


An in-game version of our physical prototype level, built with our level editor. We added water as another game element to it.

We have a fully working physics engine that is handling all collisions and simulating the mass of our slime.


Our slime has split itself into two parts and is using his mass difference to solve a puzzle with a seesaw

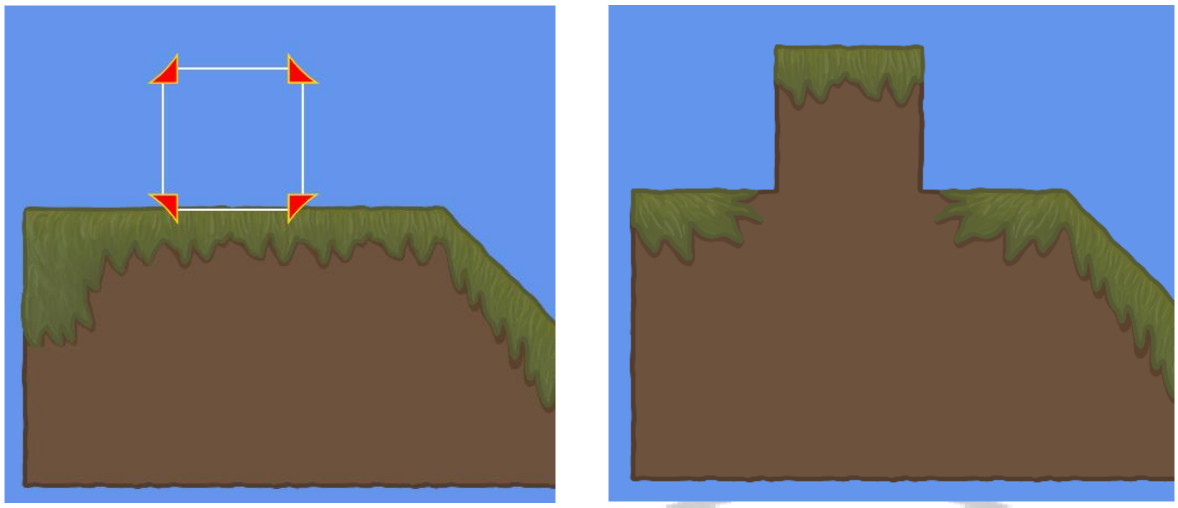
We also implemented a level editor that we can use on the PC, as well as on the Xbox One. The editor allows us to easily develop and test our levels.



The selection menu of the level editor on the left side with the preview and the cursor on the right side.

3.2. Challenges

Some parts of our level editor turned to be much harder than we thought. Especially dynamically drawing the sprites depending on their surrounding tiles was quite involving. Drawing the game art in general takes more time than we assumed.


When a new grass tile is placed in the level editor, its surrounding tiles get updated, including their boundary texture.

As a plus, writing our physics engine was trickier than we assumed, there were a lot of small physics bugs like boxes slowly sinking into the ground or bouncing around when our slime jumped on it. Fixing these was not always easy.

Tweaking the controls is also harder than we thought. In combination with our physics simulation, we have so many different parameters which can conflict each other. At first, our gravity was too low, so it felt as if our slime blob were jumping around on the moon, then we noticed that if we changed gravity, we also needed to change the jump height or the level layout.
Also, we noticed that a very high gravity can make the physics numerically unstable and as a consequence, it can make the game buggy.

3.3. Future Work

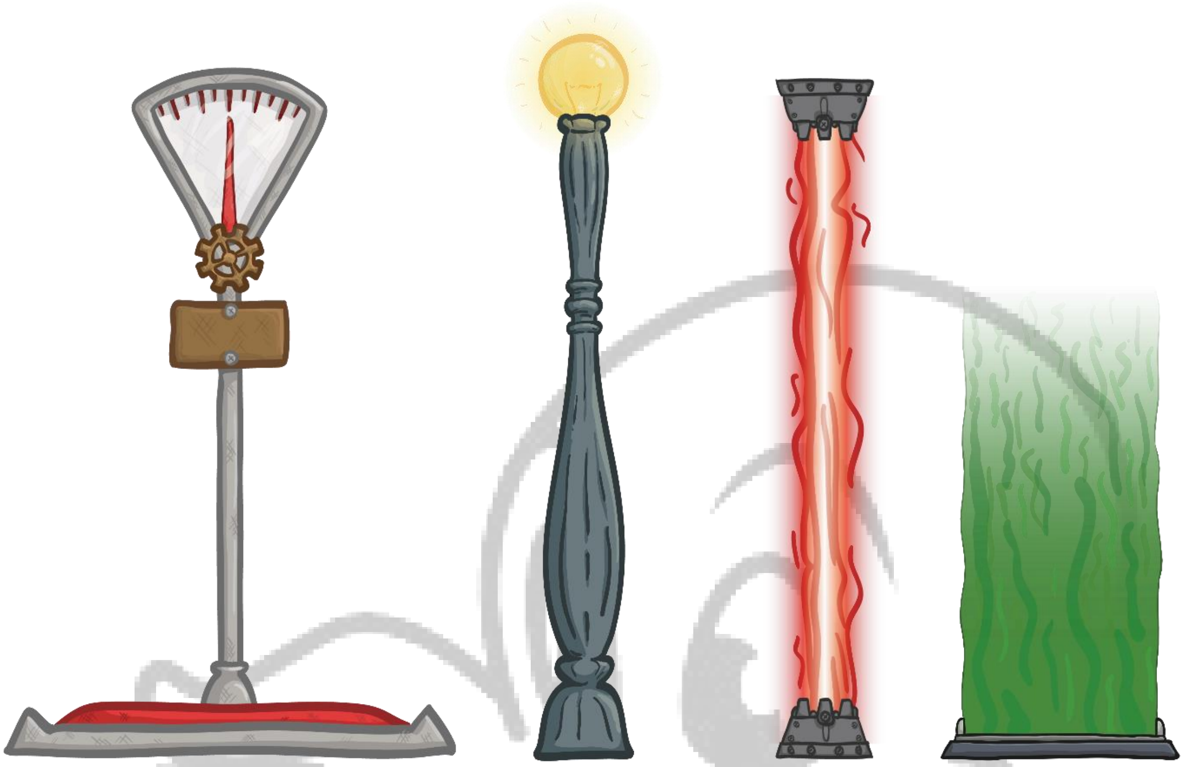
In the next two weeks we plan to work on the following things:

- Build new levels with our level editor and playtest the user experience
- Add more game elements such as enemies
- Draw more game art
- Work on our slime particle model

CHAPTER 4. ALPHA RELEASE.

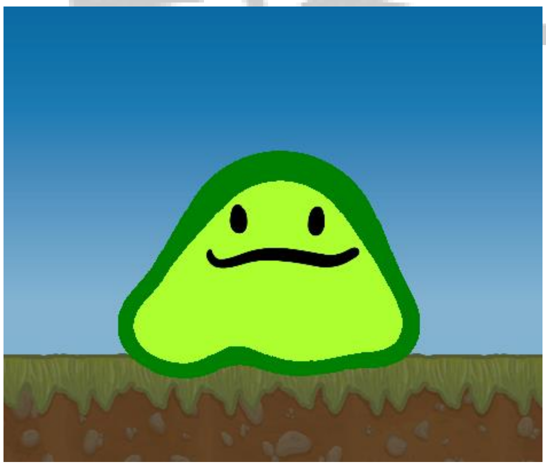
4.1. PROGRESS

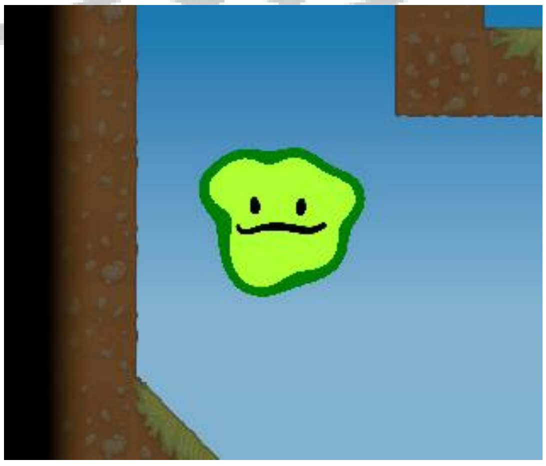
We added a lot of new game art and replaced our dummy textures from the interim demo.

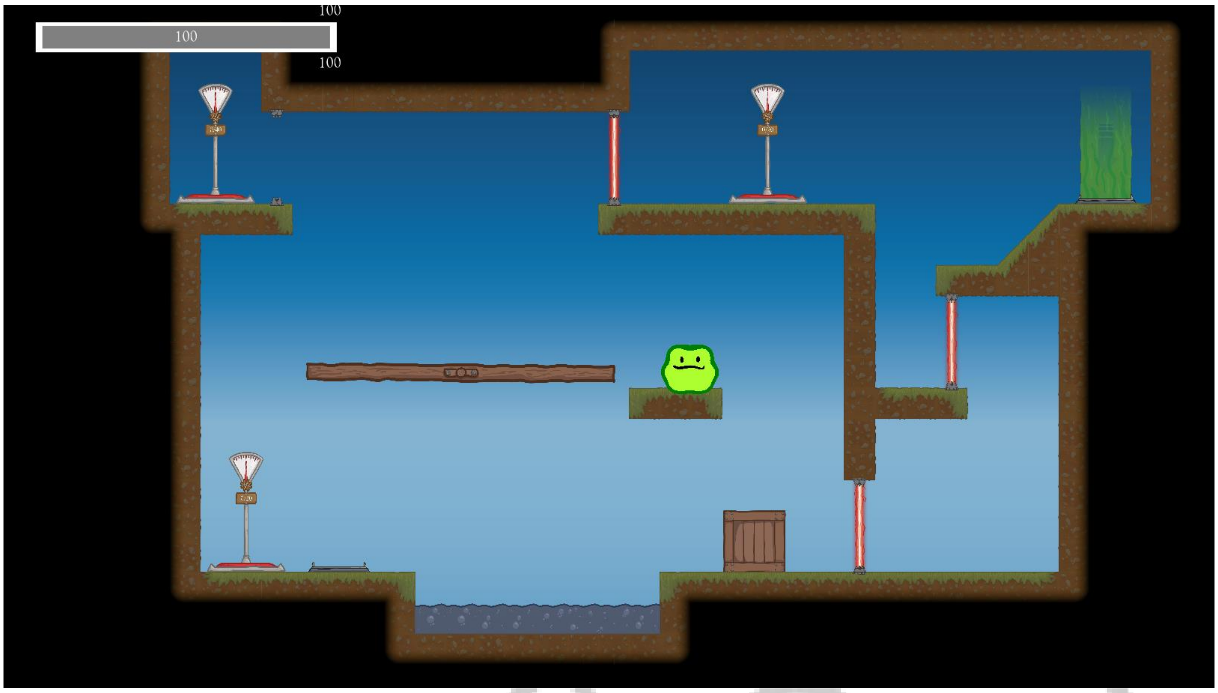


We replaced for example our button with a scale, the checkpoint flag with an electric lamp post, the doors with laser beams and the finishing position with a teleporting device to bring a more innovative feeling to the year of 1850.

Our biggest improvement is the implementation of our technical achievement – the slime particle model. We also implemented the implicit surfaces from our high target. Our character looks now much "slimier" than the version we had in our interim demo.







Our prototype level with the awesome new game art

4.2. Challenges

Implementing friction in our updated physics engine was quite difficult. Accurate friction models require the simulation on a molecular scale, which wasn't feasible for our game. It was hard to find good documentation about how to approximate friction. Just using naïve implementations as multiplying velocity with some constant close to one did lead to strange bugs (e.g. colliding objects that fall much slower than they should).
We could solve it the following way: if two movable objects collide, we ignore the effects on rotation and scale the velocity only orthogonally towards the collision normal vector, based on their relative velocity.

While drawing our new game art we changed the scale of most of our game objects. Because of this, our already designed levels had to be replaced. The change of scale was mostly due to the decision to change the size of Blurp with respect to the tile size, and due to proportion reasons. (E.g. lamp post needs to be taller than a scale, which needs to be taller than Blurp or a box, such that we can still see the scale's display.)

We decided to prioritize the current implemented game objects (weight activated scale, switch, lasers, teleporters and the seesaw) and don't implement enemies for our game in the current state. As our game is mainly a puzzle platformer, we want to focus on puzzles and not on fights with enemies.

4.3. Future Work

Until our release we want to focus on following points

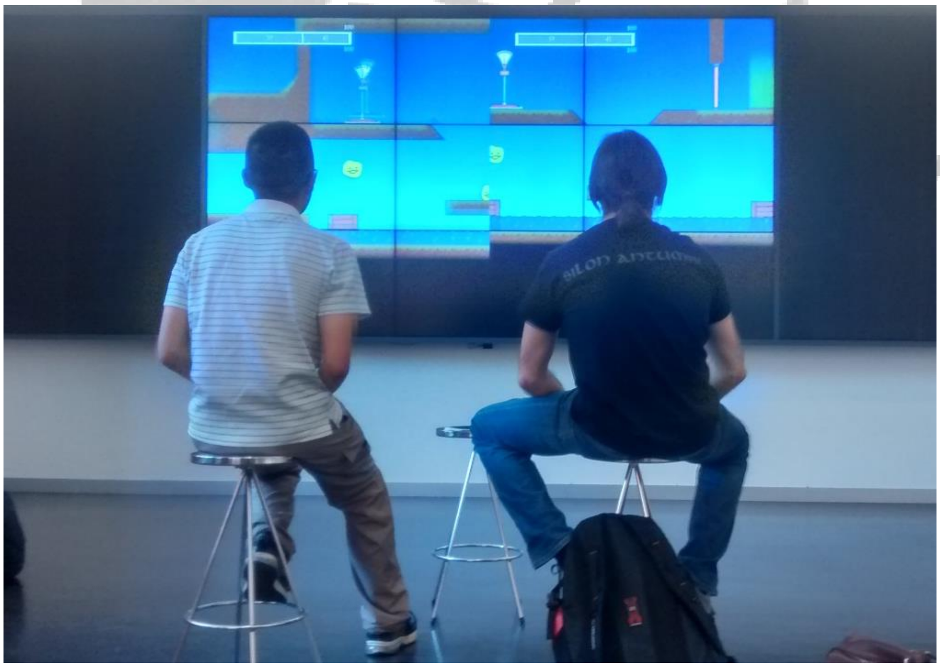
- Improve game-to-player feedback:
  If Blurp gets hurt, the player needs to get informed by a "ouch"-sound and some controller vibrations, as well as a visual hint, such as Blurp's face changing to a "hurt"-expression.
  The same goes for other sound effects, such as a humming/buzzing sound that gets louder if

you get closer to one of the laser doors, which makes it clear that this door will hurt/kill Blurp.
- Add better background music that fits better to the setting, and generally add more sound effects
- Polish current game art:
  Animations are not our top priority, however, for the scale we do think it would be good to animate the red pointer to indicate how much weight is placed on the button.
- Add more game art:
  Background textures like mountain ranges, clouds, or even heaps of technological trash (since we are on a junkyard), can be added to make the world look more populated and less bland.
- Draw story artworks:
  We need to connect Escher to our game, and visualize the storyline in a short sequence of images.
- Build more levels, that provide a good learning curve
- Hold some playtesting sessions to get feedback and adjust our game accordingly

CHAPTER 5. PLAYTEST

5.1. PLAYTESTING SESSION

We held a playtesting session in the CAB building on the big screens next to the computer science library. We advertised our session in the Gamelab Slack channel and to some of our colleagues at ETH. But thanks to the big screens we got a lot of attention and some people that were passing by stopped to play our game.
Due to the big number of playtesters, we had to provide two of our laptops to parallelize the playtesting, since some of our levels took a while to solve. Thanks to that, about 20 players could to test our game, and all the different modes (single player, cooperative and PvP) were played.


Two players testing the cooperative mode

Beside that session we also tested the game with our family members and flatmates.

5.2. Questions and Comments

Besides observing and talking to the players whilst they were playing the game, we also designed a questionnaire which they could fill out to give us more specific feedback.
The questionnaire is structured as follows: At the beginning, we get the players' general impressions of the game, and asked whether they got bored or frustrated, and after how much time they did so. Next, we asked some questions about the controls, whether they liked the button assignment and what we could change to make it better. Then we asked some questions about the camera movement, about our character and how well they could sympathize with it, before going on to the visual aspect. There we wanted to make sure that the visual representation of our game objects matches the players' expectations of what these objects do. Further on, we inquired about the level design, to get an idea about how difficult the players thought them to be, and which levels had some major issues. Finally, we checked how well the players could navigate through our menu, before finishing up with some last general questions.

Most players really enjoyed our game and the idea of splitting and merging the character in order to solve the puzzles.

The biggest criticism was given about the controls. It was said that the spit-attack and switching the slimes were confusing, since they were on the left and right shoulder buttons, and people had difficulties remembering which one to use for which action.
One useful suggestion was to maybe use both shoulder buttons to switch in both directions.
Another criticism was that the weight necessary to activate buttons was illegible, because the font was too small. Then, also, people criticized our seesaw, because it requires a lot of skill to balance the mass correctly.
In the multiplayer modes, players had difficulties seeing the edge of the split screen, so we need to add some separation bar in between them.
Some players requested more guidance in the form of an introductory tutorial and clearer indications of how much mass the currently played slime has.
For the artistic aspect, they criticized that the slime face only had 3 different states, and that we had too many black areas in our levels, and that the map felt a bit empty, since it lacked a proper background and some decorative elements.
In the level design, our last level was criticized because pushing the boxes into the water lead to non-deterministic outcomes from which solving the level might become very difficult, or even impossible to solve and had to be restarted.

We also got some cool suggestions for future game elements like speed surfaces, catapults, swings and moving platforms.

5.3. Design Revisions

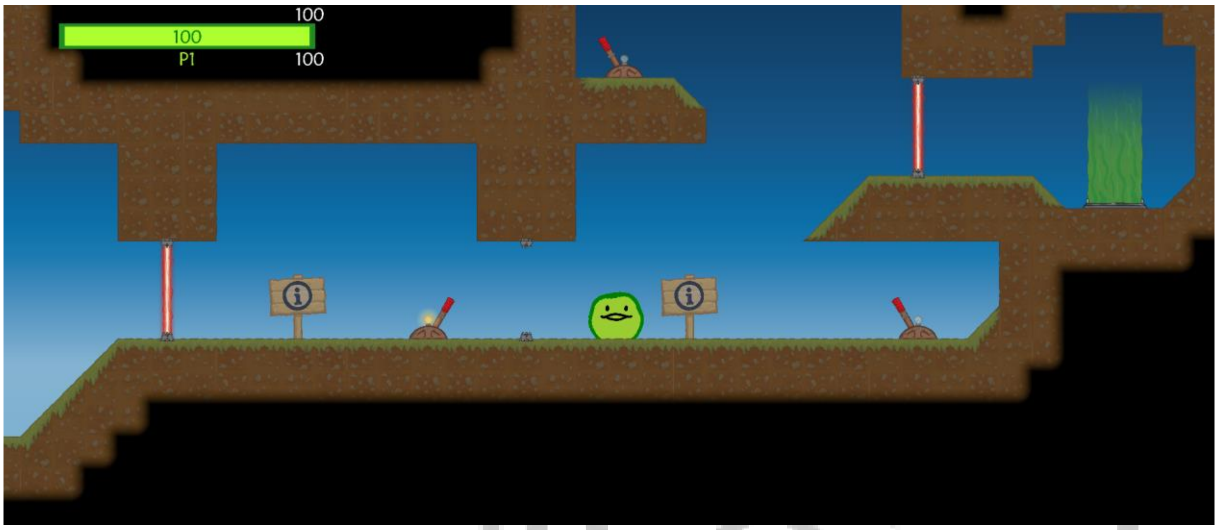
Currently, we have not yet had the time to fix a lot of these problems. However, so far, we have increased the font size on the buttons and implemented a rotation of their cursors to make it clearer how much weight these buttons need to be activated. We improved our slime mass overview where the players now can see the mass of their currently played slime.

We have also found, and fixed a bug that prevented any actions in the menu of the multiplayer mode if the second player opened the menu.

Chapter 6. Conclusion

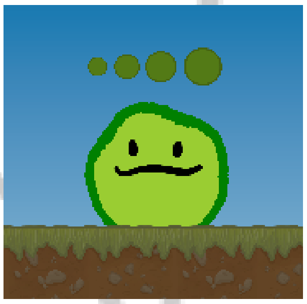
6.1. Final Results

Compared to our alpha release we added four tutorial levels to our game with signs that explain the controls and the game mechanics.



Our third tutorial level with its signs

We decided to remove the spit attack to make the controls of the game more intuitive. To give more feedback on how much the player can barf, we show now four circles with different sizes on top of the slimes.



We changed the background music of the levels and added a lot of sound effects to the game. Additionally, we drew many artworks for the story intro video. Unfortunately, MonoGame doesn't support video playback and we couldn't integrate it in the game.

6.2. Experience

Our final game is close to our vision that we had in the beginning and we are happy how it turned out.

The different milestones provided by the course were very helpful to stay on track with the project. As most submission dates were already announced at the beginning of the course we could plan the additional time that was needed for writing the report and preparing the presentations. We were a bit surprised when the submission for Studio Gobo was announced just one week before its due date. This deadline was on short notice and we didn't plan to have already a final version for the jury on that day.

6.3. Personal Impressions

We are very proud of our game and got a lot of positive reactions. It was a great feeling to watch people playing and enjoying our game!

On the technical side, one of our greatest success was the implementation of a working GJK/EPA collision detection.

The biggest technical difficulty was creating a physics engine from scratch.

Some of our group members did already have the idea of a game with a slime that can split and merge its body, before the course started. As we all loved this idea, we wanted to make such a game, but the connections we drew to the Escher theme felt a bit hollow. We did a lot to make the storyline as appealing as possible and spent a lot of time on drawing the storyline trailer. This time could instead easily have been used to implement the parallax effect, create a better-looking GUI and design some decoration elements to enrich the game visuals.
We agree that there needs to be some source of inspiration for the course, but the themes should only be treated as inspiration, not as a hard guideline to follow.

It was hard to plan the whole project ahead, as we didn't know how much time the different task will take. During the semester we realized that some tasks took much more time than we initially planned and had to shift some priorities. We didn't implement any enemies nor the parallax effect in the final game. As this was in the low target we didn't manage to completely meet the milestones. But we managed to include additional features that weren't planned in our schedule as the implicit surfaces for the slime and the multiplayer modes.

As an improvement for next year's course we would suggest that the Git repositories and the Slack chat get set up earlier. We already worked for more than one week on our Git repo before the official one was set up, which is why we didn't bother to switch over.
In the future, please make sure to communicate properly with the ZHdK such that the next GameLab students will again have the chance to work with them.

The MonoGame framework felt a bit primitive. For a large amount of time we were implementing a game engine rather than a game. If we would have had the chance to work with an engine that is more popular on the market (Unreal or Unity), we would have already learnt to handle these complex tools, which would give a benefit when applying to jobs in the game industry.