BRING BACK WINTER!

Alpha Release Report

Benedikt Bitterli, Simon Kallweit, Marcel Marti

Current Status

The functional minimum, the low target and the desirable target have been fully completed. In addition, some of the high and extra targets have been implemented.

We also put effort into making levels and testing, and put lots of time into polishing and adding small graphical effects to make the game look like a finished product.

Implemented features		
	Functional Minimum	• Fluid Simulation • Editable terrain • Extremely basic, side view rendering • Sandbox style gameplay
	Low Target	• Season changers • Switches and gates • Season sensitive gates
	Desirable Target	• Water pumps • Water pipes • Textured terrain • Season dependent graphical effects
	High Target	• Procedural audio
	Extra Target	• Post process effects shaders

Design Revisions

On the original feature list, we had additional game elements such as acid and steam in the high target. However, after making a few levels and some discussion, we decided against including these items.
This is because it was only in the level making process that we realized how much leverage we could get out of the basic items - gates, pumps and pipes - and that we would have had to make a lot more levels if we were to add more complex items into the mix. The game is not fun if a new item is introduced every other level, and so ultimately we decided against adding more items.

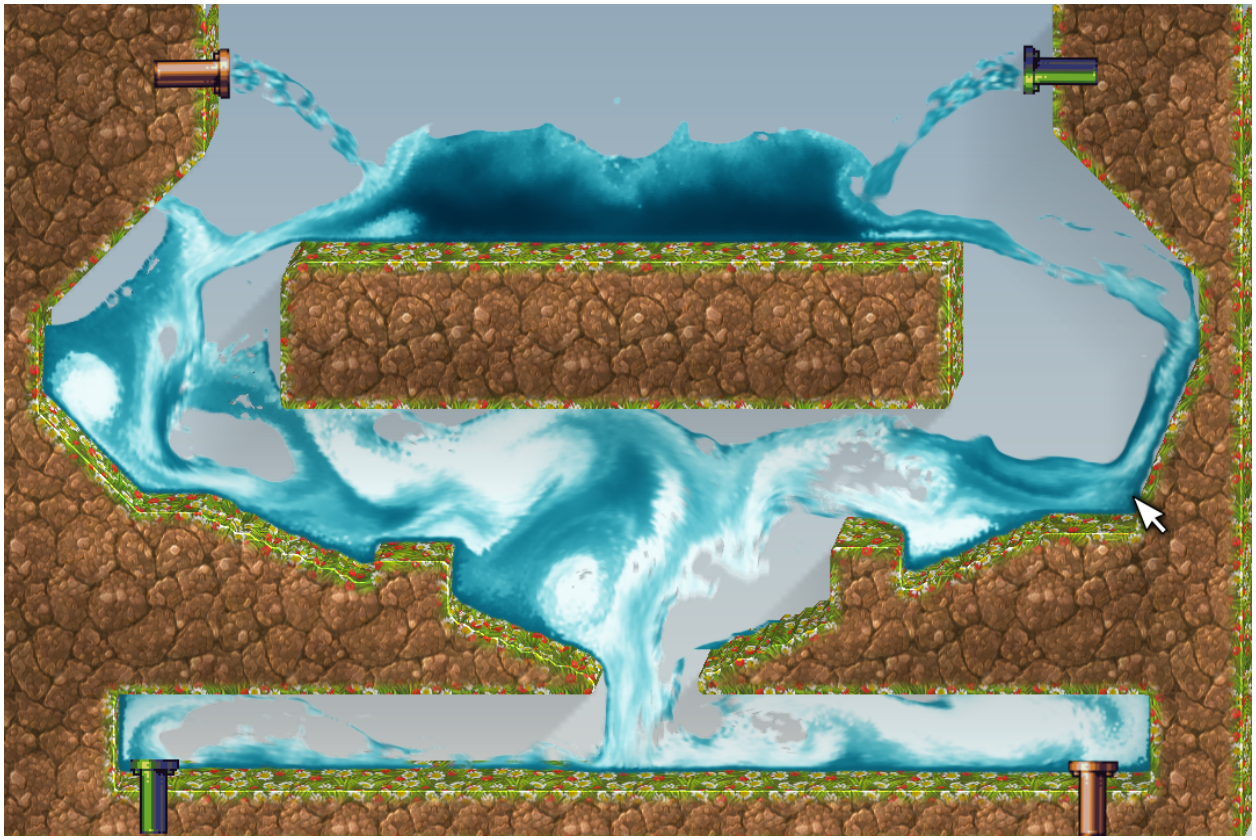
Retrospective: “Do one thing well”

During the course, the most frequent advice we were given was to “do one thing well”. The idea is to concentrate your efforts into one specific thing instead of spreading yourself too thin trying to get everything right.
During the project, we focused our time on two of these “things”; the results are shown in the next sections.

Fluid Simulation

Our initial “one thing” was the fluid simulation. Since the fluid physics are the core mechanic of our game, we had to make sure it was running fast, robust and predictable to avoid a frustrating player experience.
The initial implementation of the simulation was not very encouraging, since it was quite slow and frequently oscillated or blew up under pressure or shocks. Fortunately though, through the process described in our interim report, we were able to force the fluid to be fast and robust for the game, at the cost of physical correctness.

We are quite happy with the end result and cautiously believe that we did okay on our “one thing”:



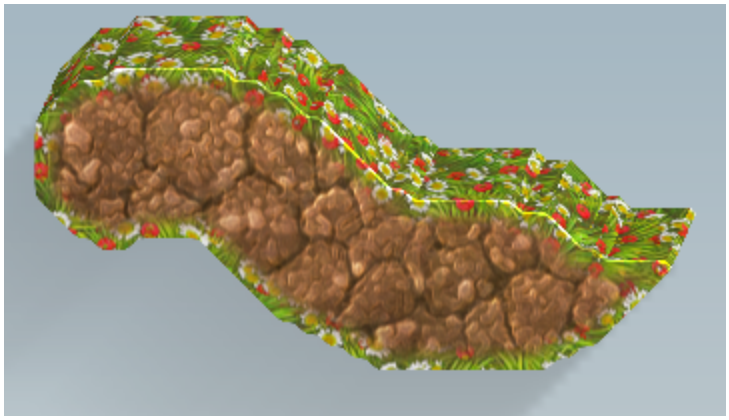
Graphics

The fluid simulation was deliberately completed very early in the game making process so that our basic building block was firmly in place. Apart from making the planning easier, this also meant that we had time to focus on a second “one thing” to try and do well.

For us, this meant trying to polish the graphics as much as we could. Since none of us are artists, this does not necessarily mean good assets, but we tried to add small, good-looking effects to get a nice and rounded look.

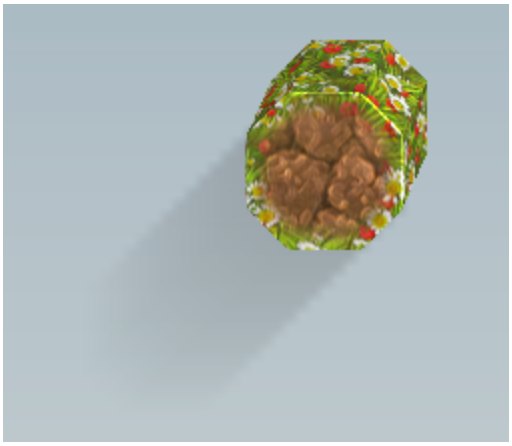
Extruded terrain

To make the landscape look less flat and boring, we decided to give it a “2.5D” look by extruding the edges of the landscape and adding a warped, textured top. The water is also slightly offset into the z axis so it can be occluded by terrain in the foreground. We also blend textures of the side- and top faces and add a bright, phong shaded rim to easier distinguish the top and side faces.



Shadows

To support the 3D look, we also added shadowing effects to the terrain. Blocks of terrain can shadow both the sky as well as the water, and terrain close to the border is slightly brightened to give a nice smooth gradient across the side face.



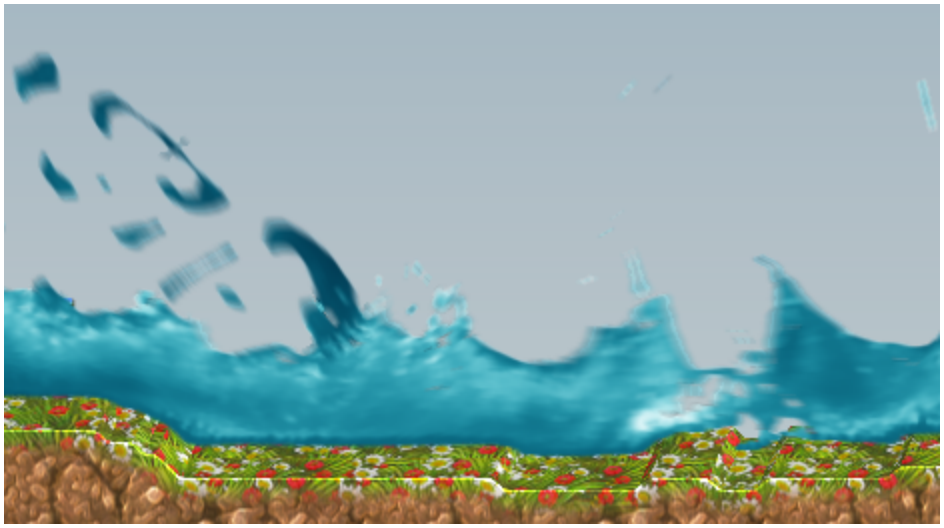
Dynamic particles

Particles help make the levels seem less empty and more alive. The particles change dynamically based on the current season; in autumn, leaves blow in the background, in spring butterflies fly accross the screen, in winter snowflakes gently fall etc.

Motion blur

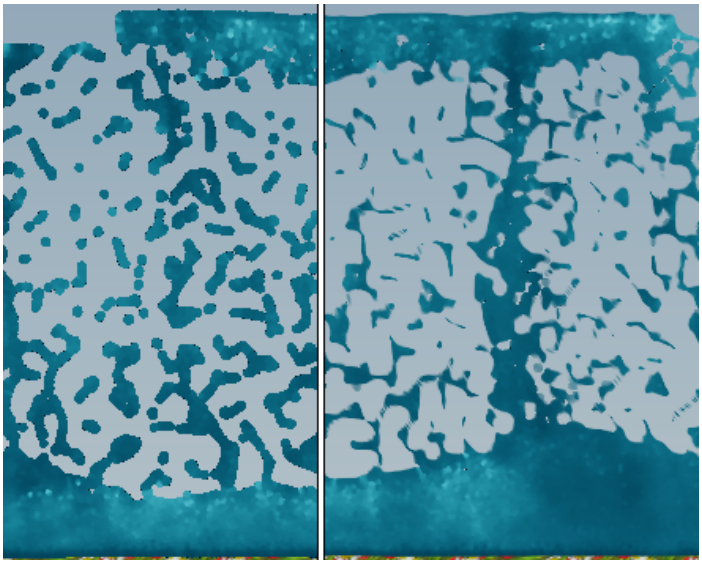
To make the water look nicer in motion, we also added a small amount of post process motion blur to emphasize interesting fluid flow. The particles are stretched along their movement direction, weighted with a metaball kernel, and then rendered into a velocity buffer. The velocity buffer then contains a weighted average of fluid velocities, which serves as input to a directional gaussian blur.

The motion blur can be slow on integrated graphics cards, and the user can turn it off in the options menu.



Anisotropic particles

To render the fluid, we use metaballs to model the surface. Unfortunately, naive metaballs make the entire fluid look “blobby”, since all particles are small circular objects. This amounts to losing a lot of the resolution that is provided by the simulation. To fix this, we implemented the paper “Reconstructing Surfaces of Particle-Based Fluids Using Anisotropic Kernels”, which aligns anisotropic kernels with the principal components of the local particle distribution. This creates large, circular blobs in the center of the fluid, but small and thin blobs aligned with the normal near the surface. The results are seen below; classic metaballs on the left and isotropic particles on the right:



Aeration model

Initially, the water had a single constant color throughout the medium, which didn’t look very good and hid a lot of interesting fluid flow from the user. To mediate this, we consulted literature and implemented an aeration heuristic that models the intake of air bubbles near splashes and surfaces and tracks the air fraction as it moves with the fluid. The color at a point in the fluid is then determined by the average air fraction, which controls an exponential color curve to model the absorption of light. This gives a pleasing color mixture and shows interesting swirls and eddies inside the fluid.