PART 3 – INTERIM REPORT

This chapter describes how the game developed from an early prototype to nearly the finished low target. It describes step by step the work that has been done and the changes to the original planning and schedule that have been made.

WEEK 1: FUNCTIONAL MINIMUM

CHANGES

One change as been applied to the requirements fulfilled by Milestone 2. The team agreed to delay the requirement "ReqP09 – Direct Combat 2" to the next milestone since it was not possible to implement the attack visualization without having a player model.

ACHIEVEMENTS

Compared to the prototype, the game has matured further. The major improvements that have been applied are in detail:

- ReqP13 and ReqP14: A new type of ranged weapon has been added: The flame thrower. Using the flame thrower, it is possible to either attack the opponent or to destroy islands. The flamethrower does not have the same range as the ice spike, but whatever is hit by its flames sustains heavy damage.
- ReqP12: The ice spike for which we had only a primitive implementation in the prototype has been redefined and improved. The aiming is now easier than before.
- ReqI07 and ReqI08: Islands now constantly lose height while carrying a player. As soon as a player jumps off the island it gradually regains its original height. This feature improves the dynamics of the game by making it faster.
- ReqI05: Islands can now collide with each other, allowing different islands on the same height.
- ReqUI06: The status strings have been replaced by a first and simple HUD.
- ReqP19: Failed

PROBLEMS

The game still has several shortcomings. Some of them have been mentioned in detail in chapter two. This is a short summary of the persisting problems:

- Navigation is not trivial
- Ice spike aiming could be better
- The game play is overloaded and needs to be streamlined
- The collision response has to be improved in certain places

THE PRODUCT

The working product features the moving islands in an already well fleshed-out form, but without any textures. Movement between the islands is still restricted to the jetpack, while a new gadget, the flame thrower, is available. It can be used to harm players, or islands. The ice spike aiming has been improved, but is still lacking accuracy. Collision detection is only done using simple collision primitives (cylinders and spheres).

WEEK 2: LOW TARGET PART 1

CHANGES

The realistic player model (ReqP03) has been moved to the desirable target, which further delays the direct combat animation (ReqP09). Some additional requirements were introduced, as a result of some additional play testing and findings from the prototype: ReqI14 is a new requirement for a visual indication of an island's health (it should glow when it gets damaged by the flame thrower). Similarly, ReqP21 is the visual indication of a player's frozen state (which could also be solved through the HUD). Also, ReqP04 (Island Attraction) has been extended to also include an easy way to jump from island to island.

ACHIEVEMENTS

The game made a huge step forward in terms of visuals compared to the functional minimum. Also, the problems of inter-island traveling have been addressed quite successfully in the form of island jump. The collision detection is also much finer grained compared to the simple primitives of Milestone 2. In detail, those changes are:

- ReqL03: A shader for realistic Lava rendering has been written, described in-depth in the corresponding section.
- ReqPi03: More sophisticated pillar models have been included, though they are not textured yet.
- ReqI03: Three different island models have been included.
- ReqUI04: An in-game menu has been added which will allow the selection of maps and players.
- ReqP04: Islands can be selected using the right analog stick; the closest island in the direction the stick points at is selected and the player can attract that island by pressing the right trigger. He can jump to that island by pressing the left trigger.
- ReqP05: A player can walk – or fly using the jetpack – to an attracted island.
- ReqI13: Power-ups respawn on a random island after a random amount of time after consumption.

PROBLEMS

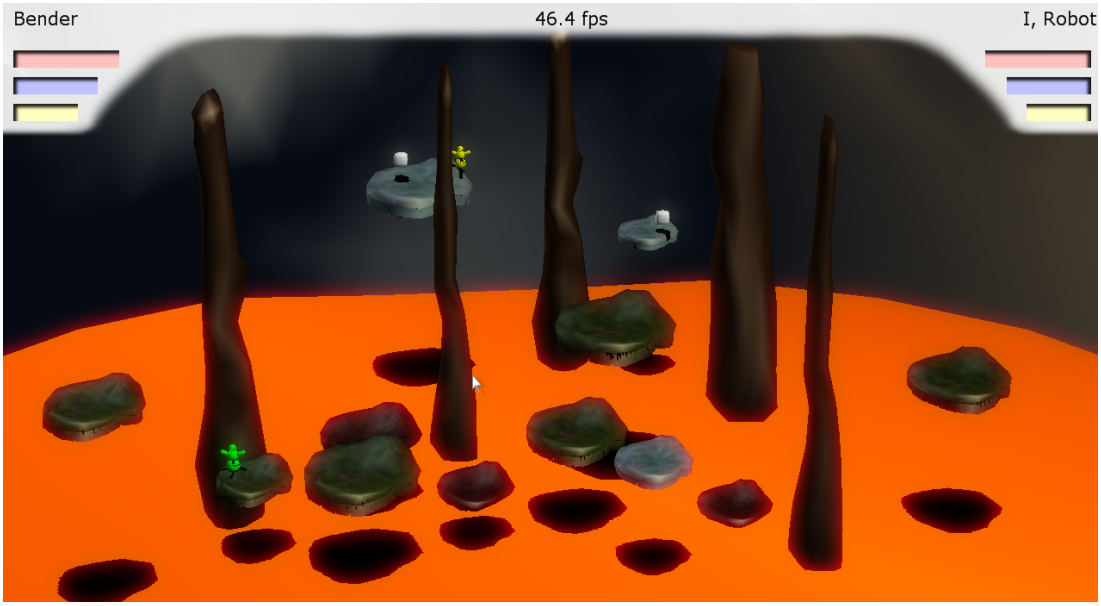
Some problems still remain, such as:

- The collision response for standing on top of an island has some flaws; it can happen that a player oscillates on top of an island or gets set on top although he collided with the island's border.

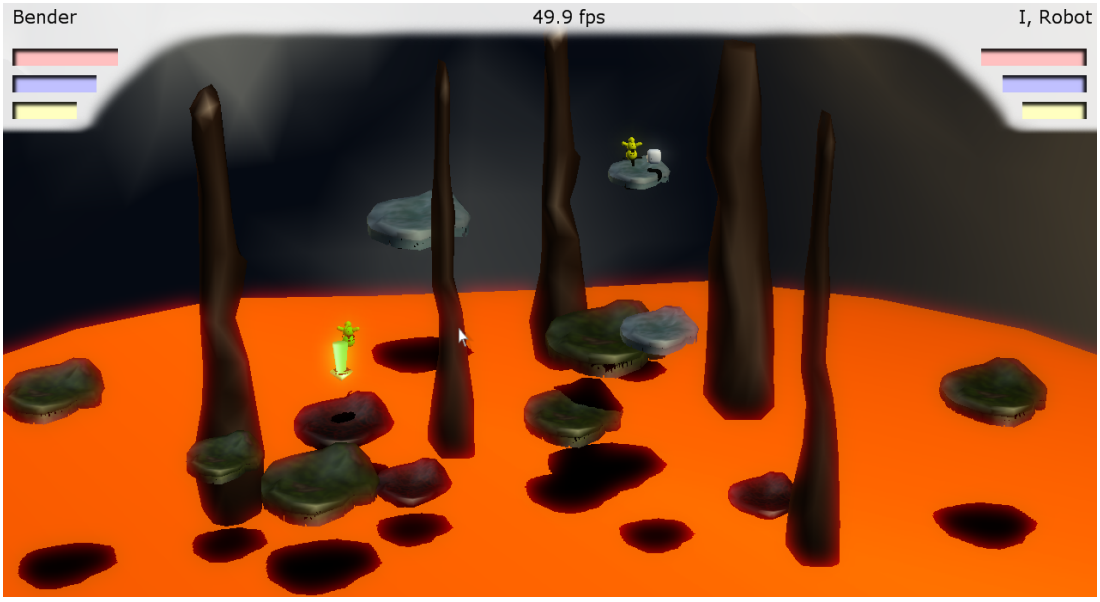
- Islands don't collide with the cave at the back, nor are they stopped from leaving the screen to the left, right or bottom.
- On island attraction, some collision response is not correct: Islands can sometimes go through pillars.

Collision response was particularly problematic, as it can heavily depend on the frame rate: if the frame rate drops and the time step increases, objects can fall through or collide again after the application of collision response – and the same (maybe inappropriate) response gets applied again. Therefore, collision response has to be fine-tuned and adapted to each object interaction combination which will take up quite some time. This frame rate dependence may also mean that we will have to multi-thread our engine, so draw and update code can be run on separate cores und a low update time step can be guaranteed.

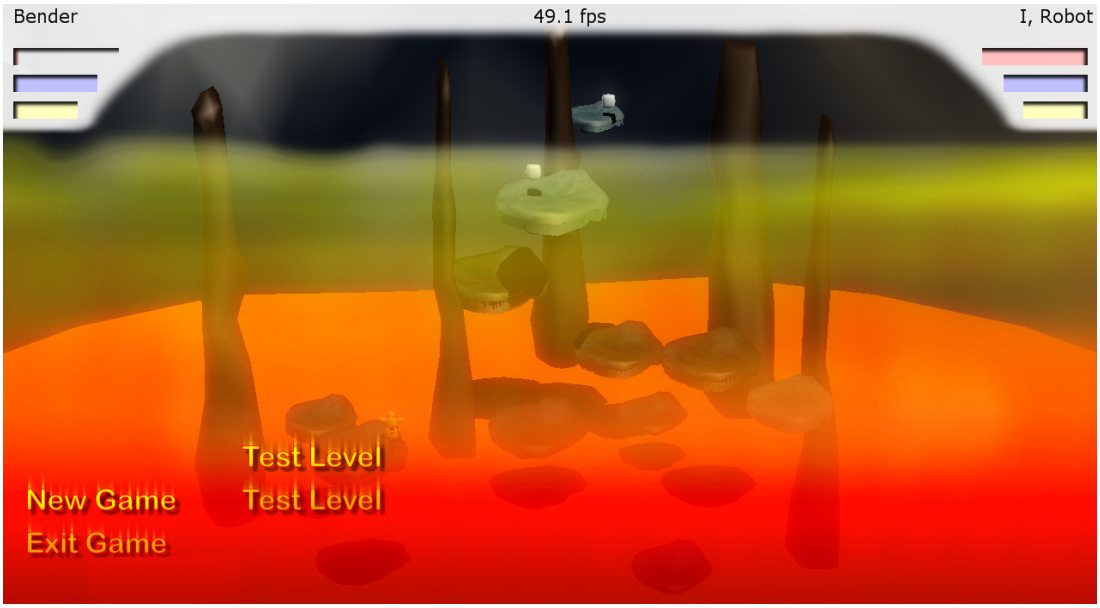
SCREENSHOTS



The current game screen including the HUD.



The green player jumping towards the selected island.



The Menu overlay.

PRODUCTION EXAMPLE - COLLISION DETECTION

OVERVIEW

To implement the game code in Project Magma, some sort of collision detection between a set of entities (namely the player, power-ups, islands and pillars) was needed. As with all the other parts of the software, the target was to keep the collision detection pluggable and easy to configure. As with all features, this allows for reconfiguring the collision detection at runtime. This has the advantage that we can test different collision volumes for different entities.

In accordance to all the other parts of the software, a new property, a collision entity, and a collision manager was introduced. Collision entities represent a “collidable” entity within space bounded by a collision volume. They are stored inside the collision manager which also tests for collisions between the entities. The binding between the simulation on one side and the collision entities and the collision manager on the other side happens inside the collision property which is attached to an actual simulation entity that should collide with other entities.

BROAD PHASE

Broad phase collision detection is currently not optimized. The collision manager uses the naïve approach testing each collision entity against each other entity.

COLLISION VOLUMES

Collision detection supports three different types of volumes:

- Bounding spheres
- Bounding cylinders aligned to the unit y-axis
- Triangle trees. These are trees of axis aligned bounding boxes containing triangles inside the leaf nodes. Each leaf contains up to five triangles.

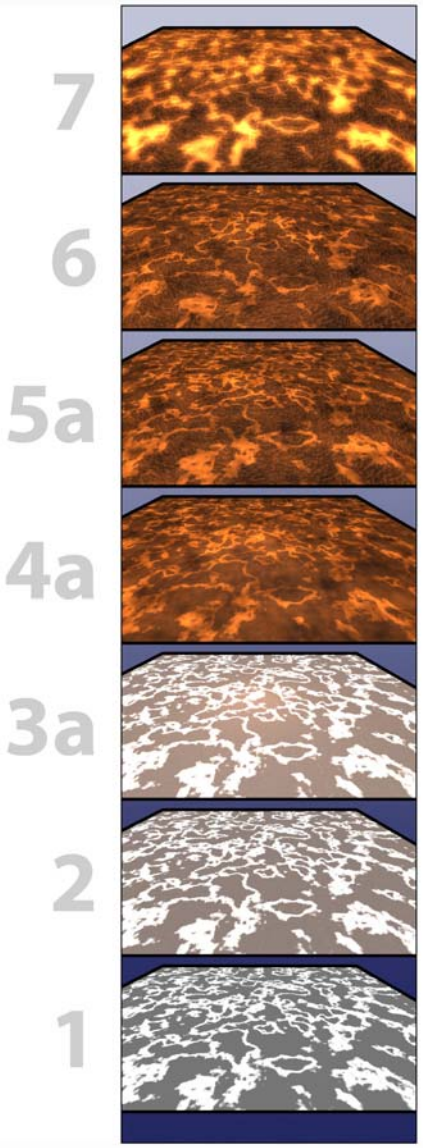
The content pipeline creates all three collision volumes for each triangle mesh. The level designer then chooses which bounding volume is assigned to a given entity.

PRODUCTION EXAMPLE - CREATING LAVA SURFACES

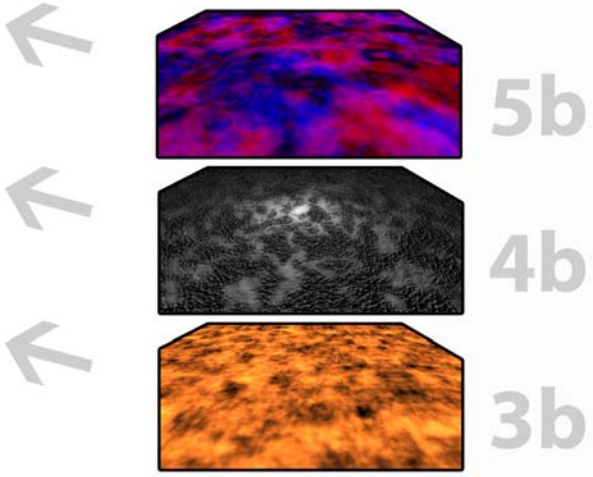
FIRST APPROACH: LAVA PLANES

We would like to show an example of a graphical element which we consider to be crucial for a credible ambience of our game. This serves both as a documentation for our own reference and for a work report for the lab.

We started our research in lava rendering by searching the web for tutorials describing how to create lava effects in offline rendering systems like Maya. We found http://en.9jcg.com/comm_pages/blog_content-art-94.htm to be the one with the nicest results and implemented it first in Maya and then as a GPU effect.



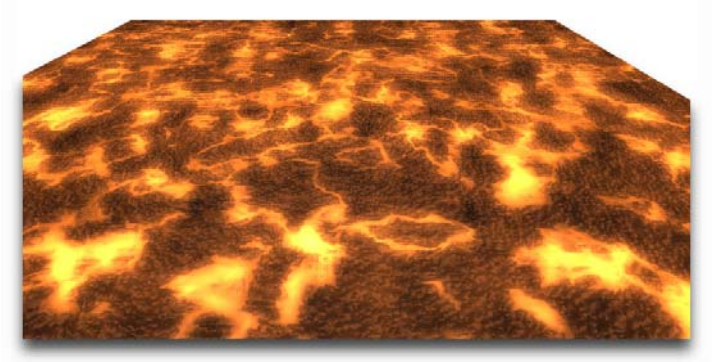
1 – An overlay of several fractal textures (Stucco) generated in Maya. Offsetting these with respect to each other will be used for animation. 2 – The dark parts of Stucco are replaced by a slight granite texture generated in Maya. 3a – We add diffuse shading to allow for normal mapping later on.



3b – Another fractal texture, generated in Maya and luminance-amplified in the shader. 4a – The fractal texture gets blended in and moves slowly over the plane which simulates moving fog. 4b/5a – A normal map is added to give the dark parts (stones) some structure. 5b/6 – Two cloud renderings, generated in Photoshop, are used to generate a pseudo-random field of UV vectors which are used to distort the texture coordinates. This simulates air flickering due to the heat. 7 – A final glow with Gaussian Blur is added.

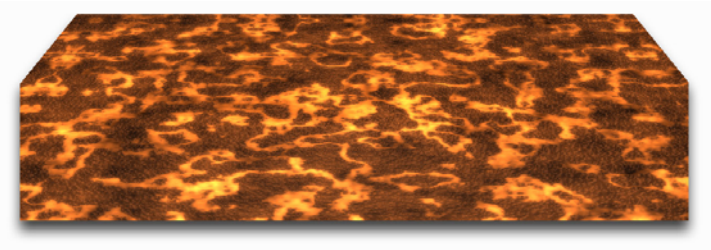
We had to omit the displacement part for now, but we got everything else to work with some tweaking. We added the heat flickering effect as described above by slightly distorting the texture coordinates.

GOING BEYOND PLANES: PARALLAX OCCLUSION MAPPING

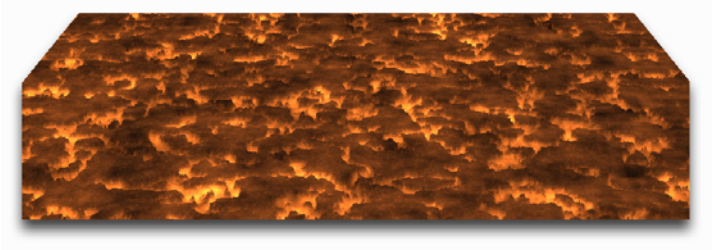


The effect of the shader described above looks already quite pretty when seen from a perspective projection like the one in the picture to the left.

However, we had to find out that the effect owes much of its dramaticism to the wide-angle perspective we've used during the development of the shader. As discussed in an earlier chapter, though, our gameplay requires an almost orthographic view onto the scene in order to maintain maximal clarity for the players navigating in the scene.



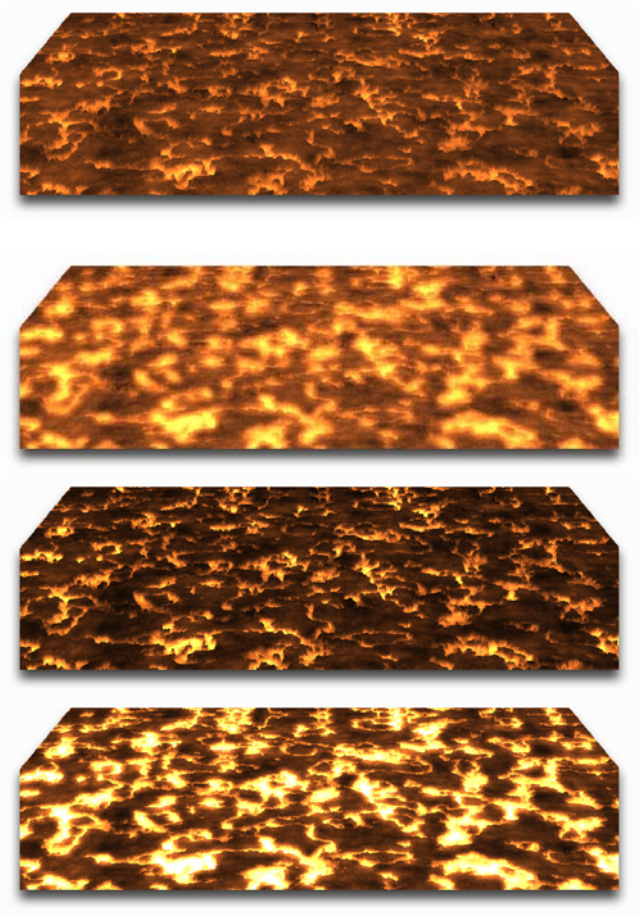
After using the camera parameters from the game itself, much of the effect is lost (see right). First, the pattern appears to be much more monotonous than before, and suddenly we miss the notion of depth. Since the angle between the camera and the ground plane is relatively flat in our setting, we thought that it would be nice to have some actual geometric structure in the lava instead of just plain normal mapping. To find out if this would help, we took an implementation of Parallax Occlusion Mapping and included it into our shader.



As we show on the left, we regained a large part of the depth of the scene we've lost previously due to the perspective change.

At this point, we started to get more creative by altering parameters of the individual layers. We inverted, compressed or luminance-scaled the height map, introduced new color mappings and changed the strength of PO mapping. Soon, it became apparent that small changes in individual parameters led to under- or oversaturation quite fast, and the need for some simple global tone mapping arose. As we already had a post-processing stage, this was easy to implement and it turned out that a 3rd order Lagrange polynomial with interactively modifiable parameters already does the trick. On the next two pages, we show examples of results we achieved with different parameter sets.

An increasing issue of PO mapping became the performance. We are currently working on emulating the same effect with several planes, alpha maps and alpha testing.



First – the original shader, just with Parallax Occlusion Mapping enabled.

Second – a very big glow radius and low-contrast settings in the HDR post-processing stage.

Third – low glow radius but relatively high contrast settings in the post-processing stage.

Fourth – higher glow radius, intentional oversaturation to emphasize the perception of a very bright light source in the lava.

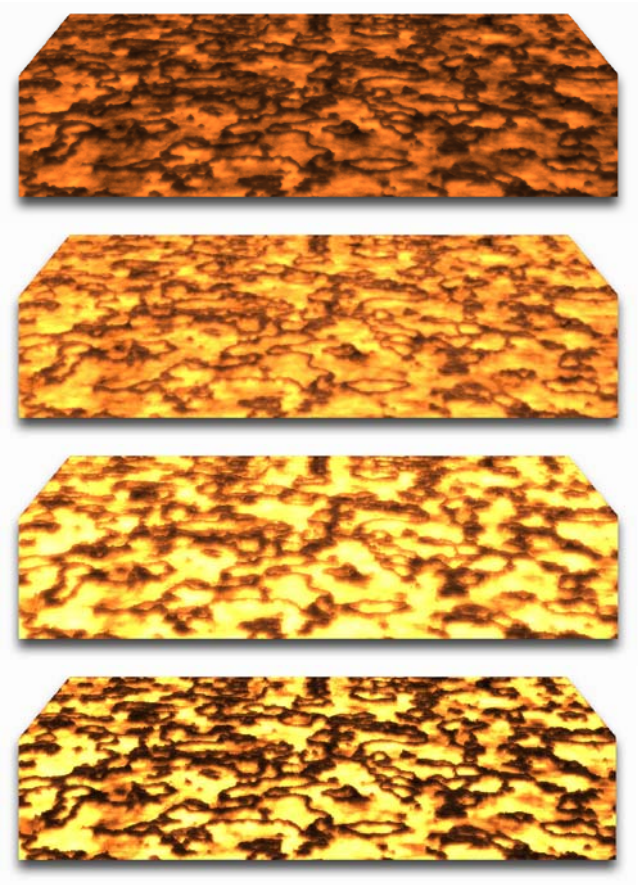
In this set, we inverted the depth effect of PO mapping by negatively scaling the occurring gradient term. The Stucco map which combines the textures (see earlier) is still unchanged, though.

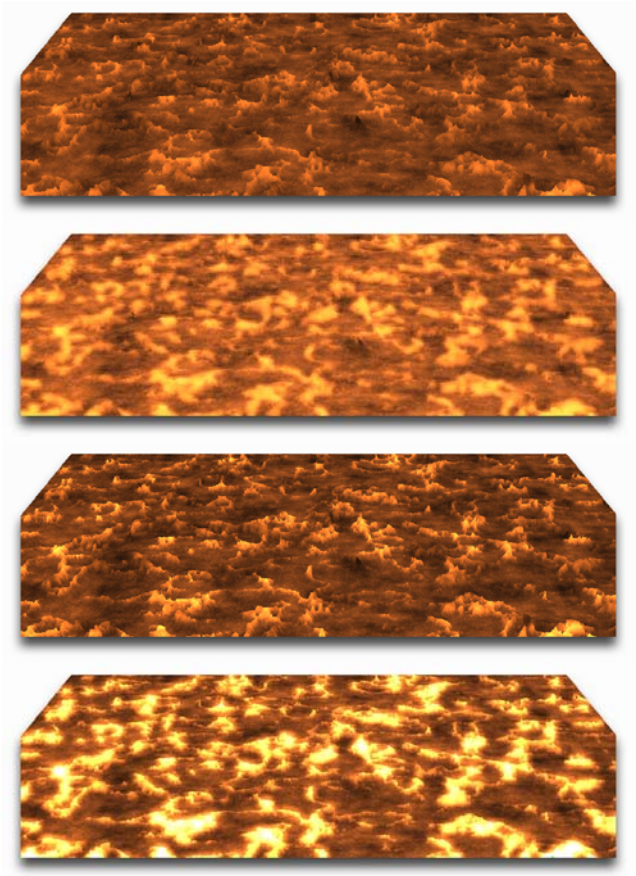
First – low glow radius and strength, linear tone mapping.

Second – all illuminations are scaled up to create an uniformly hot surface.

Third – exaggerated contrast.

Fourth – even more exaggerated contrast. The black ridges can be interpreted as floating ashes.





In this set, we inverted the Stucco texture which serves as both a height map and a blending operator between texture layers. Afterwards, we let the Gradient unchanged, so the entire effect is just inverted.

First – low glow radius and strength, linear tone mapping.

Second – a very big glow radius and low-contrast settings in the HDR post-processing stage.

Third – enhanced contrast. The bright structures can be interpreted as little flames which move along the surface.

Fourth – extreme contrast. The flame effect is exaggerated now to indicate that the fire is really bright.

In this set, we inverted both the Stucco texture and its gradient afterwards. This leads to big, bright, burning chunks on the surface.

First – low glow radius and strength, linear tone mapping.

Second – a very big glow radius and low-contrast settings in the HDR post-processing stage.

Third – enhanced contrast.

Fourth – extreme contrast.