

Game Notebook Project Magma



Janick Bernet

Dominik Käser

Christian Oberholzer

29.05.2009

TABLE OF CONTENTS

| | |
|--|----|
| Table of Contents | 2 |
| Part 1 – Formal Game Proposal | 4 |
| Intro | 4 |
| Informal Description | 4 |
| Overview | 4 |
| Game elements | 4 |
| Concept Sketches | 7 |
| Formal Requirements | 11 |
| General | 11 |
| GUI AND HUD | 11 |
| Lava | 11 |
| Pillars | 11 |
| Floating Islands | 12 |
| Player | 12 |
| Development Schedule | 13 |
| Deliverables | 14 |
| Milestones | 15 |
| Task Assignments and Work Estimation | 16 |
| Development Timetable | 17 |
| Assessment | 19 |
| Part 2 – Prototype | 20 |
| Gameplay Screenshots | 21 |
| Findings | 22 |
| Positioning | 22 |
| Player movement | 22 |
| Island Travel | 23 |
| Parameter Tuning | 23 |
| Part 3 – Interim Report | 24 |
| Week 1: Functional Minimum | 24 |
| Changes | 24 |
| achievements | 24 |
| Problems | 24 |
| The Product | 24 |
| Week 2: Low Target Part 1 | 25 |

| | |
|---|----|
| Changes..... | 25 |
| achievements..... | 25 |
| Problems | 25 |
| Screenshots..... | 26 |
| PRODUCTION EXAMPLE - Collision Detection..... | 28 |
| Overview | 28 |
| Broad Phase | 28 |
| Collision Volumes..... | 28 |
| PRODUCTION EXAMPLE - CREATING LAVA SURFACES | 29 |
| FIRST APPROACH: Lava planes..... | 29 |
| GOING BEYOND PLANES: PARALLAX OCCLUSION MAPPING | 30 |

PART 1 – FORMAL GAME PROPOSAL

INTRO

For reasons still being researched, volcanoes started to appear all over Antarctica, flushing resources of unprecedented value onto the earth's surface. Although the resources legally belong to the state of Antarctica, the immense value of said resources led other fractions to claim ownership. Day after day, new gatherers arrived, trying to capture as much as they could. As the situation got out of control, the world union decided to legally distribute the resources all over the planet. In a time of great decadence it was decided that shares shall be dispensed based on the outcomes of deadly robot-matches inside the volcanoes. Since then, engineers all over the world have constantly been working on improving their robots in order to be able to explore and to claim the deadly depths of Antarctica's volcanoes.

INFORMAL DESCRIPTION

OVERVIEW

The game features 2-4 players competing against each other (mainly in death match, but other modes such as control point or capture-the-flag are also conceivable) on one screen, viewed from a fixed angle (no scrolling, but automatic zoom has to be tested). The screen wraps around: if a player leaves to the right he will enter from the left and vice versa.

The competition takes place around a lake of lava. Large pillars stick out of the lava into the sky. Between the pillars, there are islands hovering on different heights. The players can stand on these islands, change the paths of the islands and go from one island to another. Islands can collide with each other and pillars, which can result in islands and/or pillars falling down and taking other objects with them. When a player stands on an island, it will slowly lose height because of the added weight. If a player leaves the island before it eventually sinks into the lava, it hovers back to its original position. Sunken islands can be replaced by new ones using a ray of cold water. Periodically, eruptions from the lava in the form of fireballs will appear and hurt players if they get hit.

GAME ELEMENTS

ENVIRONMENT

The game environment consists of a rectangular field where all the action takes place. The borders wrap around, meaning that everything disappearing on one side reappears on the other side. This battle ground basically consists of the following three different elements:

- A sea of lava covers the ground and is - of course - deadly to the players
- Rock pillars of different sizes stick out of the lava
- Rock islands hover on a specific height above the field of lava.

A more precise definition of these elements follows.

PILLARS

Pillars just stick out of the lava. Islands can collide with them and tilt them over. When a pillar falls, it can take other pillars or islands with it. On the top, the pillars are covered in ice which is constantly melting – therefore, water runs down along the pillars.

HOVERING ISLANDS

Islands hover on a specific height (Y axis) on a specific path between the pillars. When islands collide with each other or pillars, they are only deflected from their path on the XZ plane and never leave their fixed position on the Y axis. Players can stand on islands, but they will slowly lose height and eventually melt in the lava below. Islands are covered by grass and other flora. Islands in the upper heights can also be covered in ice.

PLAYER CHARACTERS

Players control characters, which have a certain amount of health and energy. A player can move between the islands and attack other players. While melee attack is free, energy is consumed if a player performs some special attack (see Indirect combat (Chicken tactics)). Health is deduced when a player gets hit by another players attack. If a player's health is zero or below, he dies and loses. A player also dies when falling into the lava.

POWER-UPS

Simple power ups for health and energy are distributed over the islands. They will randomly re-appear if collected.

PLAYER INTERACTIONS

Every player can perform the following actions without using any finite resource:

WALKING (BORING BUT NECESSARY)

Players can walk around the islands, though they cannot fall from them just by walking.

COLLECT POWER-UPS (RED BULL GIVES YOU WINGS!)

If a player gets in contact with a power-up he can collect it and will receive the power accordingly.

ISLAND ATTRACTION (USE THE FORCE, LUKE)

Islands can be attracted using some fancy force which makes them slowly move towards the island the player is standing on, so he can switch to the other island.

ISLAND JUMP (UP AND AT THEM)

The player can activate his jet pack for a very short amount of time which allows him to go from one island to another.

ISLAND REPULSION (GASSY EMISSION)

The player can change the path of an island either temporarily or completely. He does so by grasping the island and emits a burst of air using his jet pack.

DIRECT COMBAT (MANO-A-MANO)

Every player has a melee attack ability which costs no energy. A melee attack will both deduce health from his enemy as well as physically push the opponent away from the attacker. The latter one can be exploited to push an opponent over the edge of an island.

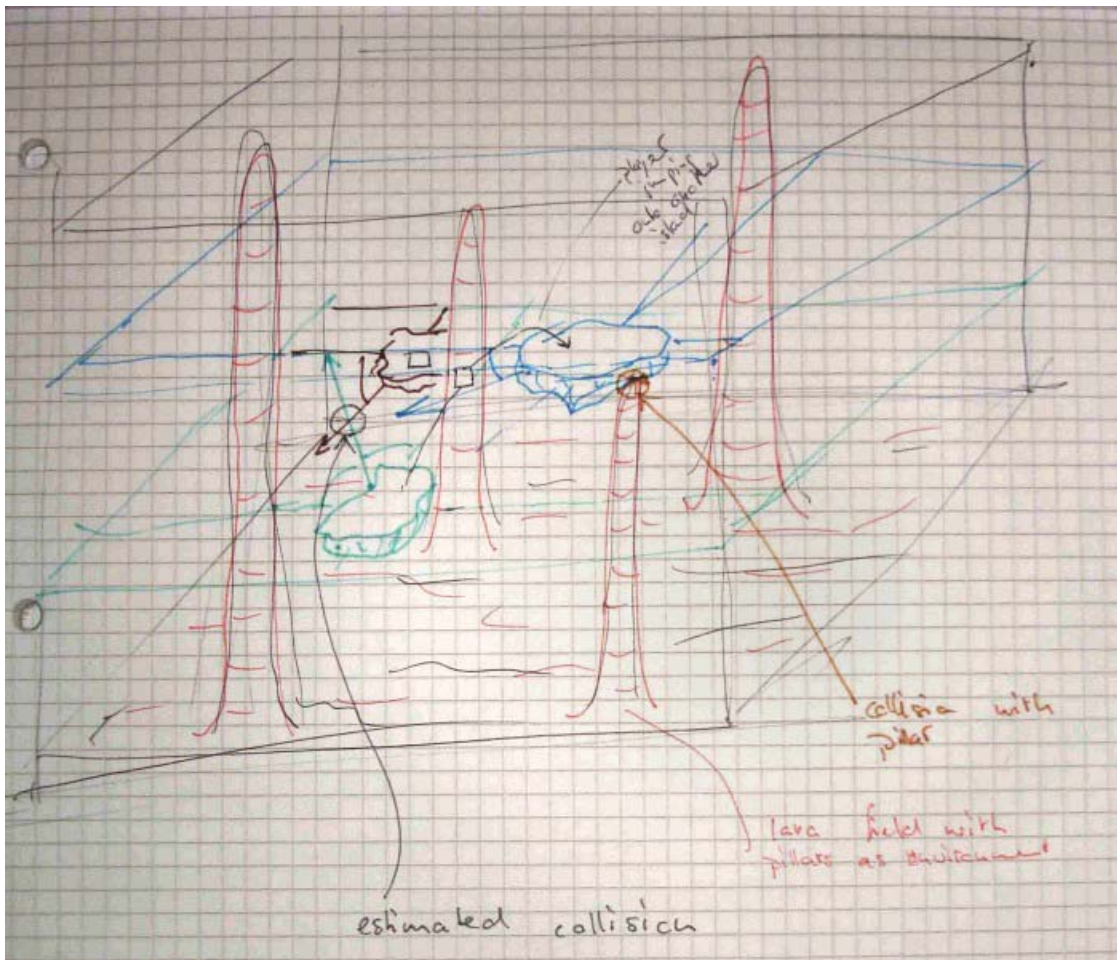
Furthermore, every player has energy as a resource. Energy will recharge itself with time and can be used to perform the following actions:

INDIRECT COMBAT (CHICKEN TACTICS)

A player has various means of indirect combat in the form of special abilities:

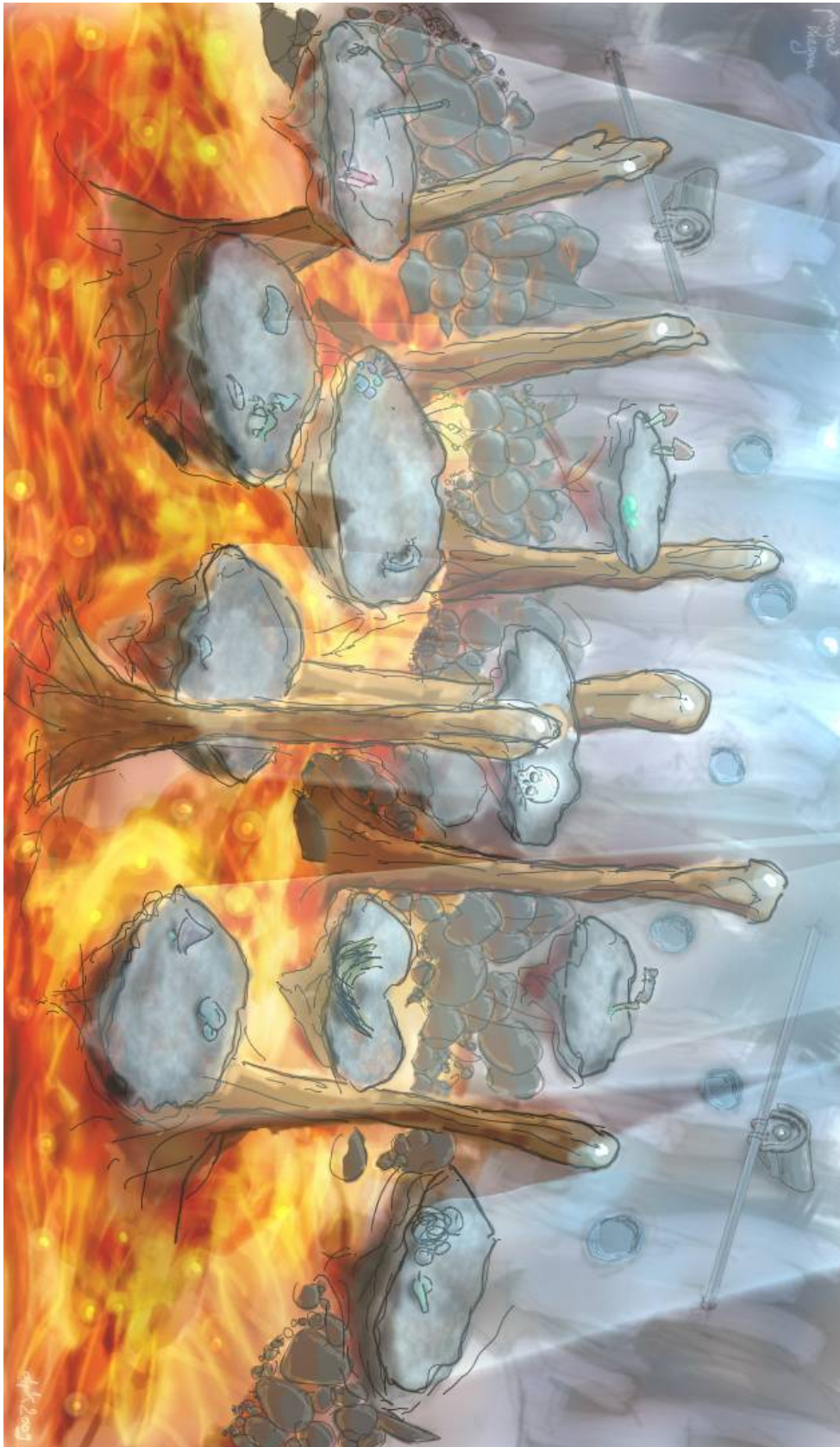
- **Ice spike:** The player can specify a direction in which, subsequently, a spike is sent off. If the spike hits an enemy, he will get hurt and frozen for a short period. If the spike hits lava, an island will be created.
- **Snow storm:** The player can specify a point in range; a cloud will appear and start snowing on the creatures below it, causing damage.
- **Fire wall:** The player can lighten up a fire on the floor which will remain there for a fixed amount of time. Players stepping on the fire will be hurt.
- **Small robot spawning (aka binary fission):** The player can spawn a robot on the current island which will be there for a fixed amount of time and attack all enemies stepping on the island.

TYPICAL IN-GAME SITUATION

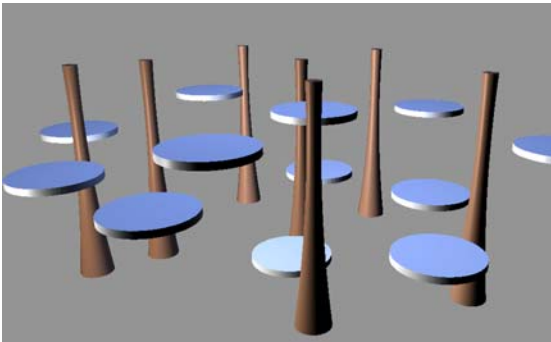


On this image one sees:

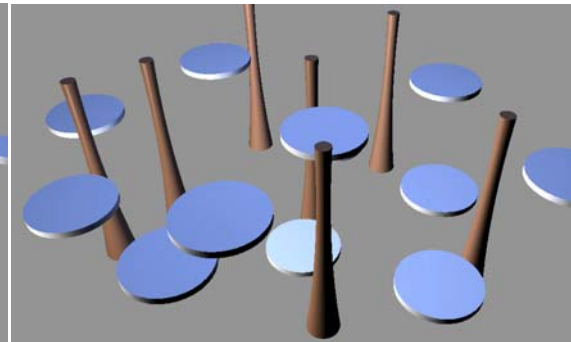
- 4 pillars
- 3 Islands floating, two of them on the same height
- Estimated collision point between the green and the brown island. After the collision, they will change their movement direction.
- Players are visualized by rectangles. Player 1 sits on the brown island waiting to shoot at player 2. Player two on the other hand flees from the crash onto the blue island.



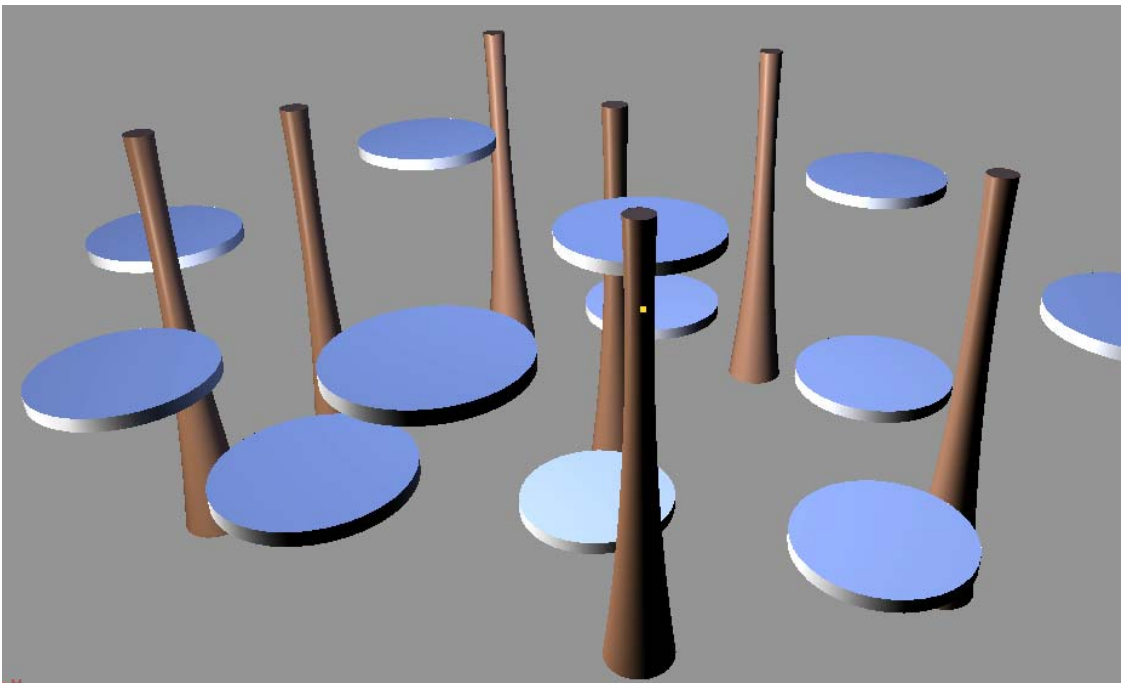
How the game could look when it is done.



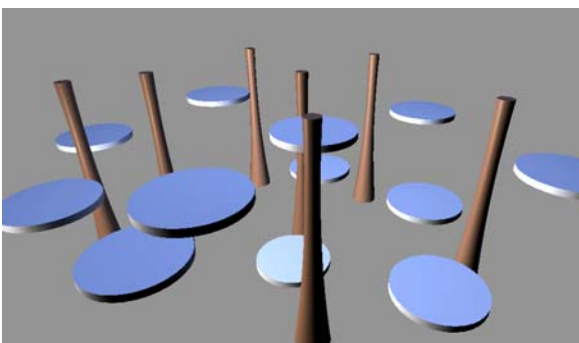
An alternate view angle of 18 degrees. It is difficult here to navigate in the XZ plane.



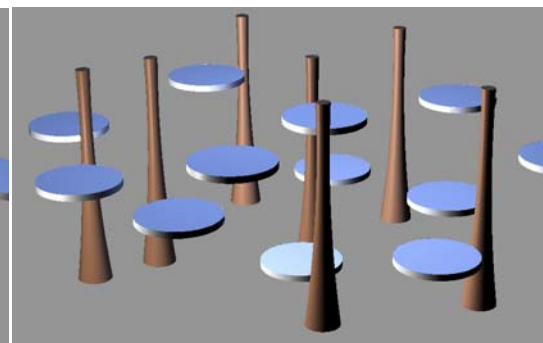
An alternate tilt angle of 38 degrees. The notion of height is difficult to grasp here.



We deemed this view to be optimal in both perspective ($f=21$) and tilt angle (26 degrees).

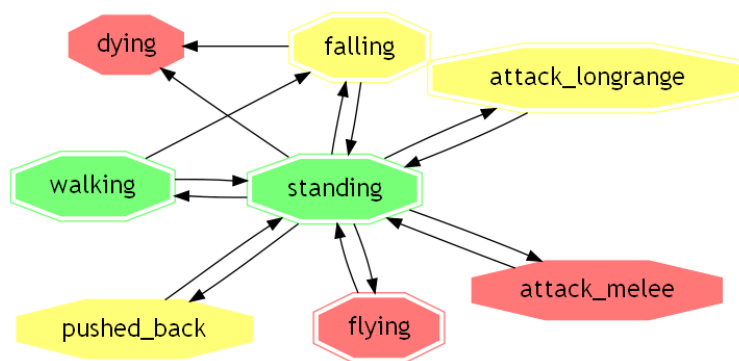


An alternate perspective, $f=21$. The distortion is too large, players would stay in the front.



A more orthographic perspective, $f=71$. There is no dramatics, the look and feel is too static.

MODEL ANIMATION STATES

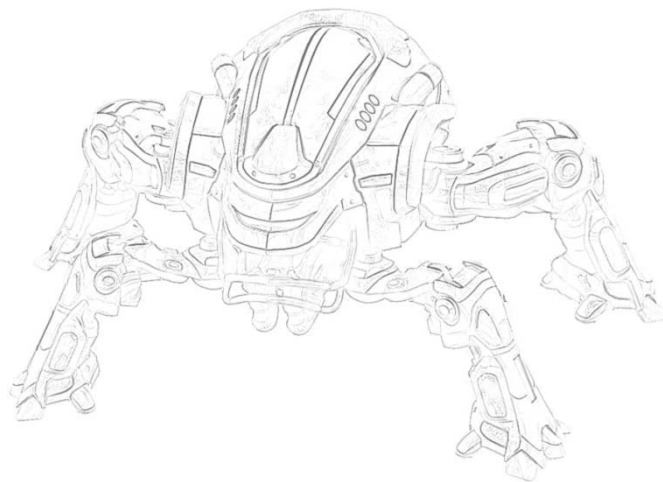


A finite state automata model of player animations. Colors denote priorities of realization (green is high, red is low). Outlined states denote looping animations.

ROBOT MODELS



We particularly like the look and feel of this robot we found on the web. The head is over proportional to the body which yields a more comic look and feel. We might want to go for a longer head to make it look more aggressive, though.



A concept of little prop robots which are spawned on islands to make an island hostile (high target).

FORMAL REQUIREMENTS

GENERAL

| ID | Requirement | Description |
|--------|--------------------------|--|
| ReqG01 | Basic Camera | The basic camera captures the scene from a predefined position. The whole game area is always visible. |
| ReqG02 | Advanced Camera | The camera films the scene from a varying position. It always films from the same side, but height and zoom may vary depending on the optimal setting. |
| ReqG03 | Basic Software Framework | Setting up a generic framework that is expandable, embeds the game logic, graphics and similar. The framework should be built as much on XNA as possible. But still every new feature should be addable as a separate component. |
| ReqG04 | HDR Rendering | Setting up the renderer to render with high definition textures and effects. This feature significantly improves the visual appearance of the game. |
| ReqG05 | Shadow Rendering | Rendering the scene with shadows using a state-of-the-art technique. |
| ReqG06 | Statistics | Keep track about players win and losses, their longest live, their fastest kill and their fastest death. |

GUI AND HUD

| ID | Requirement | Description |
|---------|------------------|--|
| ReqUI01 | Start Screen | There is a start screen from where one can start a new game and view the high score. |
| ReqUI02 | High Score | The high score features statistics (defined in ReqG06) about past games. |
| ReqUI03 | Text Input | Text can be entered using the controller. |
| ReqUI04 | Player Selection | Players can select their desired character and enter their name. |
| ReqUI05 | Map Selection | The first player can select a map to play in. |
| ReqUI06 | Simple HUD | A HUD showing each player's health and energy has to be available. |
| ReqUI07 | Fancy HUD | A beautifully designed HUD that nicely integrates with the game environment has to be available. |
| ReqUI08 | Intro | An intro explains the game's background story. |

LAVA

| ID | Requirement | Description |
|--------|------------------------|--|
| ReqL01 | Lava Ground | The ground is covered by lava. This requirement represents the game-logic of the lava. |
| ReqL02 | Basic Lava Effect | A basic effect to render the lava lake. A basic red rectangle is enough for a first prototype. |
| ReqL03 | Polished Lava Effect | A polished and nice effect to render the lava lake. This includes advanced shaders. |
| ReqL04 | Deadly Lava | If the player gets into contact with the lava he dies. |
| ReqL05 | Fire Eruptions | At random there are fire eruptions emerging from the lake. |
| ReqL06 | Harmful Fire Eruptions | If such a fire eruption hits a player he endures damage or dies. If the eruption hits an island it throws the island off its course. |

PILLARS

| ID | Requirement | Description |
|---------|-------------|--|
| ReqPi01 | Pillars | Pillars of different sizes stick out of the lava. This requirement |

| | | |
|----------------|--------------------------------|---|
| | | represents the need to model pillars with respect to in-game logic. |
| ReqPi02 | Basic Pillar Rendering | There is some model representing pillars which stick out of the lava. |
| ReqPi03 | Sophisticated Pillar Rendering | Realistically rendered pillars stick out of the lava. |
| ReqPi04 | Tilt Pillars | Pillars can be tilt over by islands. The resulting fall can affect other islands and pillars. |
| ReqPi05 | Icy Pillars | Pillars have a top consisting of ice, which melts to water that runs down the pillar and drops into the lava. |

FLOATING ISLANDS

| ID | Requirement | Description |
|---------------|--------------------------------|--|
| ReqI01 | Floating Islands | Initially there is a set of floating islands of rock. The islands hover above the lake of lava in different heights. |
| ReqI02 | Basic Island Rendering | A basic rendering such that the islands are visible and useable inside a game. |
| ReqI03 | Sophisticated Island Rendering | A polished and nice effect to render the islands. |
| ReqI04 | Moving Floating Islands | Islands have the ability to move. They move with a given velocity. |
| ReqI05 | Crashing Islands | If an island crashes into another island the collision will be resolved according to physics. The resulting movement should be locked onto the x/z plane the resulting rotation only respective to the y-axis. |
| ReqI06 | Islands and Pillars | If an island crashes into a pillar the collision will be resolved according to physics. The resulting movement should be locked onto the x/z plane the resulting rotation only respective to the y-axis. |
| ReqI07 | Sinking Islands | If a player stands on an island it will lose height. |
| ReqI08 | Rising Islands | If the island does not carry the player it regains its original height. |
| ReqI09 | Melting Islands | If an island gets into contact with lava it melts. |
| ReqI10 | Destructible Islands | If an island takes enough damage, either by a players special ability or by falling pillars it will fall apart. |
| ReqI11 | Icy Islands | Islands hovering above a specific height are slightly or fully covered in ice. |
| ReqI12 | Power-Ups | Power-Ups are lying on the islands. |
| ReqI13 | Power-Up Re-spawn | Power-Ups re-spawn if consumed on a random island |
| ReqI14 | Island Health Indication | If islands are being destroyed by heat the progress of destruction shall be indicated by an increasing glow. |

PLAYER

| ID | Requirement | Description |
|---------------|----------------------------|---|
| ReqP01 | Player | The player has to be represented within the game-logic. |
| ReqP02 | Basic Player Model | A model for the player is available. |
| ReqP03 | Sophisticated Player Model | A realistic model for the player is available. |
| ReqP04 | Island Attraction | A player can use attract an island so it floats to the side of the island the player is standing on. As soon as the island is not attracted anymore, it hovers back to its original position. |
| ReqP05 | Island Walking | The player can walk to an island he attracted. |
| ReqP06 | Island Jumping | A player can use the jetpack to move from one island to another. |
| ReqP07 | Island Repulsion | A player can use the jetpack to emit bursts of air which will for a short period of time get an island to drift off its original course. If it |

| | | |
|---------------|----------------------------------|---|
| | | collides with a pillar it could change its course completely. |
| ReqP08 | Direct Combat 1 | Every player has a melee attack ability which costs no energy. This will deduce health from his enemy. |
| ReqP09 | Direct Combat 2 | A realistic attack animation is displayed. |
| ReqP10 | Direct Combat 3 | Melee attacks will also physically push the opponent away from the attacker. |
| ReqP11 | Energy | Every player has an energy bar which is displayed in the UI. Energy will recharge itself with time. Every used skill will use a fixed amount of energy. |
| ReqP12 | Ice Spike | The player can specify a direction in which, subsequently, a spike is sent off. If the spike hits an enemy, he will get hurt and frozen for a short period. |
| ReqP13 | Flame Thrower Damage | The player can use a flame thrower to cause damage to another player. |
| ReqP14 | Flame Thrower Island Destruction | The player can use a flame thrower to target and destroy islands. |
| ReqP15 | Building Islands with Ice Spikes | If the spike hits a rising fire ball, an island will be built. |
| ReqP16 | Snow storm | The player can specify a point in range, a cloud will appear and start snowing on the creatures below it, causing damage. |
| ReqP17 | Fire Wall | The player can lighten up a fire on the floor which will remain there for a fixed amount of time. Players stepping on the fire will be hurt. |
| ReqP18 | Small Robot Spawning | The player can spawn a robot on the current island which will be there for a fixed amount of time and attack all enemies stepping on the island. |
| ReqP19 | Aiming Aids | Visual aids for helping the player aim (during ranged combat or islands jumping) shall be implemented to simplify controlling a player. |
| ReqP20 | Collecting Power-Ups | Players can collect power-ups and get their respective bonuses. |
| ReqP21 | Slow Indication | If a player has been slowed, this state shall be indicated graphically. |

DEVELOPMENT SCHEDULE

The development shall be divided into consecutive layers. All of the requirements defined under are classified and assigned to one of them. Those layers are:

1. **Prototype:** The prototype serves to play test the central game-logic and contains only the most minimal graphical features needed to represent the game state. If any feature is removed from this part the prototype will degrade from a game into a technical prototype.
2. **Functional minimum:** This first layer contains the set of requirements minimally required to play the game and also some first simple visuals. The functional minimum is the first milestone.
3. **Low target:** The low target is the second layer and also a milestone. Though it contains more features than the bare minimum, it is still essentially not what should be achieved during the timeframe of fourteen weeks. Still it will serve as a good indicator if the development is still inside the timeframe laid out in this chapter.
4. **Desirable target:** This layer and milestone is what the project aims at. It contains all the requirements that make up a well polished and fun to play game.

5. **High target:** The high target contains additional features that will make it into the final deliverable if the team has some free time to implement them. There is no milestone defined for it. After finishing the Desirable Target it will be decided which features of this target will make it into the gold version milestone.
6. **Extras:** This part of the schedule defines some additions to the game that would be fun but are not realistic to achieve. However in a future project they could be added.

The layers then are assigned to milestones to be reached on a specific date. Those milestones contain a detailed timetable determining when each requirement will be implemented and who is responsible for the implementation. This timetable shall be filled out iteratively during the projects development.

DELIVERABLES

PROTOTYPE

| ID | Requirement |
|---------|--------------------------|
| ReqG01 | Basic Camera |
| ReqG03 | Basic Software Framework |
| ReqL01 | Lava Ground |
| ReqL02 | Basic Lava Effect |
| ReqL04 | Deadly Lava |
| ReqPi01 | Pillars |
| ReqPi02 | Basic Pillar Rendering |
| ReqI01 | Floating Islands |
| ReqI02 | Basic Island Rendering |
| ReqI04 | Moving Floating Islands |
| ReqP01 | Player |
| ReqP02 | Basic Player Model |
| ReqP06 | Island Jumping |
| ReqP08 | Direct Combat 1 |
| ReqP10 | Direct Combat 3 |
| ReqP12 | Ice Spike |
| ReqI12 | Power-Ups |
| ReqP20 | Collecting Power-Ups |

FUNCTIONAL MINIMUM

| ID | Requirement |
|---------|----------------------------------|
| ReqI05 | Crashing Islands |
| ReqI06 | Islands and Pillars |
| ReqP09 | Direct Combat 2 |
| ReqI07 | Sinking Islands |
| ReqI08 | Rising Islands |
| ReqP13 | Flame Thrower Damage |
| ReqP14 | Flame Thrower Island Destruction |
| ReqP11 | Energy |
| ReqUI06 | Simple HUD |
| ReqP19 | Aiming Aids |
| ReqG05 | Shadow Rendering |

LOW TARGET

| ID | Requirement |
|---------|---|
| ReqL03 | Polished Lava Effect |
| ReqPi03 | Sophisticated Pillar Rendering |
| ReqI03 | Sophisticated Island Rendering |
| ReqP03 | Sophisticated Player Model (may be moved) |
| ReqUI04 | Player Selection |
| ReqUI07 | Fancy HUD |
| ReqI14 | Island Health Indication |
| ReqP21 | Slow Indication |
| ReqP04 | Island Attraction |
| ReqP05 | Island Walking |
| ReqI13 | Power-Up Re-spawn |

DESIRABLE TARGET

| ID | Requirement |
|---------|------------------|
| ReqG02 | Advanced Camera |
| ReqP07 | Island Repulsion |
| ReqG06 | Statistics |
| ReqUI03 | Text Input |
| ReqUI01 | Start Screen |
| ReqUI02 | High Score |
| ReqUI05 | Map Selection |

HIGH TARGET

| ID | Requirement |
|---------|----------------------------------|
| ReqG04 | HDR Rendering |
| ReqL05 | Lava Eruptions |
| ReqL06 | Harmful Fire Eruptions |
| ReqPi04 | Tilt Pillars |
| ReqPi05 | Icy pillars |
| ReqI09 | Melting Islands |
| ReqP15 | Building Islands with Ice Spikes |
| ReqUI08 | Intro |

EXTRAS

| ID | Requirement |
|--------|----------------------|
| ReqI10 | Destructible Islands |
| ReqP16 | Snow Storm |
| ReqP17 | Fire Wall |
| ReqP18 | Small Robot Spawning |
| ReqI11 | Icy Islands |

MILESTONES

| ID | Milestone | Description | Due Date |
|------|---------------------------|---|---------------|
| MS01 | Prototype Chapter Written | With this milestone the prototype chapter must have been written and added to the game notebook. Everyone in the team should also have installed and experimented with XNA in order to be ready for | March 16, 5pm |

| | | | |
|--|-----------------------------|--|----------------|
| development. | | | |
| Additionally a game prototype according to the prototype specification has been created. | | | |
| MS02 | Functional Minimum | With this milestone the functional minimum must be implemented, working and tested. | March 23, 12pm |
| MS03 | Interim Report Written | With this milestone the chapter with the interim report must have been written and added to the game notebook. | April 6, 5pm |
| MS04 | Low Target | With this milestone the low target shall be hit. | April 13, 12pm |
| MS05 | Desirable Target | With this milestone the team must have fulfilled the requirements for the desirable target. The prototype must be tested and in presentable order since it is needed for play testing in the week after. | May 4, 12pm |
| MS06 | Play test Chapter Written | With this milestone the play test chapter must have been written and added to the game notebook. This concludes that to this date all the play testing must be done. | May 11, 5pm |
| MS07 | Gold Version | With this milestone the development must have been concluded. All testing must have been finished and some of the high target functionality should be implemented. | May 25, 12pm |
| MS08 | Conclusion and Presentation | With this milestone the conclusion chapter must have been written and added to the game notebook. In addition the public presentation of the game must be ready to be held. | May 29, 5pm |

TASK ASSIGNMENTS AND WORK ESTIMATION

PROTOTYPE

| ID | Requirement | Assignee | Work Estimate |
|----------------|--------------------------|----------|---------------|
| ReqG01 | Basic Camera | cob | 2h |
| ReqG03 | Basic Software Framework | cob | 8h |
| ReqL01 | Lava Ground | jab | 3h |
| ReqL02 | Basic Lava Effect | cob | 2h |
| ReqL04 | Deadly Lava | jab | 3h |
| ReqPi01 | Pillars | cob | 3h |
| ReqPi02 | Basic Pillar Rendering | cob | 2h |
| ReqI01 | Floating Islands | jab | 2h |
| ReqI02 | Basic Island Rendering | dpk | 4h |
| ReqI04 | Moving Floating Islands | jab | 4h |
| ReqP01 | Player | dpk | 10h |
| ReqP02 | Basic Player Model | jab | 4h |
| ReqP06 | Island Jumping | jab | 4h |
| ReqP08 | Direct Combat 1 | jab | 1h |
| ReqP10 | Direct Combat 3 | jab | 2h |
| ReqP12 | Ice Spike | jab | 3h |
| ReqI12 | Power-Ups | cob | 2h |
| ReqP20 | Collecting Power-Ups | cob | 1h |

FUNCTIONAL MINIMUM

| ID | Requirement | Assignee | Work Estimate |
|----|-------------|----------|---------------|
|----|-------------|----------|---------------|

| | | | |
|---------|----------------------------------|-----|-----|
| ReqI05 | Crashing Islands | cob | tbd |
| ReqI06 | Islands and Pillars | cob | tbd |
| ReqP09 | Direct Combat 2 | jab | tbd |
| ReqI07 | Sinking Islands | dpk | tbd |
| ReqI08 | Rising Islands | dpk | tbd |
| ReqP13 | Flame Thrower Damage | jab | tbd |
| ReqP14 | Flame Thrower Island Destruction | cob | tbd |
| ReqP11 | Energy | jab | tbd |
| ReqUI06 | Simple HUD | jab | tbd |
| ReqP19 | Aiming Aids | dpk | 8h |

LOW TARGET

| ID | Requirement | Assignee | Work Estimate |
|---------|--------------------------------|----------|---------------|
| ReqL03 | Polished Lava Effect | dpk | |
| ReqPi03 | Sophisticated Pillar Rendering | dpk | |
| ReqI03 | Sophisticated Island Rendering | dpk | |
| ReqP03 | Sophisticated Player Model | | |
| ReqUI04 | Player Selection | jab | 3h |
| ReqUI07 | Fancy HUD | jab | 3h |
| ReqI14 | Island Health Indication | cob | |
| ReqP21 | Slow Indication | cob | |
| ReqP04 | Island Attraction | jab | 4h |
| ReqP05 | Island Walking | jab | 2h |
| ReqI13 | Power-Up Re-spawn | jab | 1h |
| None | Advanced Collision Detection | cob | |
| None | Gamplay testing | jab | 4h |

DEVELOPMENT TIMETABLE

WEEK 11: 9.3.-15.3. WORKING TOWARDS MS01

| ID | Requirement | Assignee | Mo | Tue | Wed | Thu | Fri | Sat | Sun |
|---------|--------------------------|----------|----|-----|-----|-----|-----|-----|-----|
| ReqG01 | Basic Camera | cob | | 2 | | | | | |
| ReqG03 | Basic Software Framework | cob | 8 | | | | | | |
| ReqL01 | Lava Ground | jab | | 3 | | | | | |
| ReqL02 | Basic Lava Effect | cob | | | 2 | | | | |
| ReqL04 | Deadly Lava | jab | | 3 | | | | | |
| ReqPi01 | Pillars | cob | | | 3 | | | | |
| ReqPi02 | Basic Pillar Rendering | cob | | | 2 | | | | |
| ReqI01 | Floating Islands | jab | | | 2 | | | | |
| ReqI02 | Basic Island Rendering | dpk | | 4 | | | | | |
| ReqI04 | Moving Floating Islands | jab | | | | 4 | | | |
| ReqP01 | Player | dpk | | 4 | 4 | 2 | | | |
| ReqP02 | Basic Player Model | jab | | | | 4 | | | |
| ReqP06 | Island Jumping | jab | | | | | 4 | | |
| ReqP08 | Direct Combat 1 | jab | | | 1 | | | | |
| ReqP10 | Direct Combat 3 | jab | | | 2 | | | | |
| ReqP12 | Ice Spike | jab | | | | 3 | | | |

| | | | | | | | | | |
|---------------|----------------------------------|-------------|---|--|--|--|---|---|--|
| ReqI12 | Power-Ups | cob | 2 | | | | | | |
| ReqP20 | Collecting Power-Ups | cob | 1 | | | | | | |
| None | Testing | jab/dpk/cob | | | | | 4 | 4 | |
| None | Work Estimates and Plan for MS05 | jab/dpk/cob | | | | | 1 | 1 | |

WEEK 12: 16.3.-22.3. WORKING TOWARDS MS02

| ID | Requirement | Assignee | Mo | Tue | Wed | Thu | Fri | Sat | Sun |
|----------------|----------------------------------|-------------|----|-----|-----|-----|-----|-----|-----|
| ReqG01 | Basic Camera | cob | | 2 | | | | | |
| ReqG03 | Basic Software Framework | cob | 8 | | | | | | |
| ReqL01 | Lava Ground | jab | | 3 | | | | | |
| ReqL02 | Basic Lava Effect | cob | | | 2 | | | | |
| ReqL04 | Deadly Lava | jab | | 3 | | | | | |
| ReqPi01 | Pillars | jab | | | 3 | | | | |
| ReqPi02 | Basic Pillar Rendering | cob | | | 2 | | | | |
| ReqI01 | Floating Islands | jab | | | 2 | | | | |
| ReqI02 | Basic Island Rendering | dpk | | 4 | | | | | |
| ReqP01 | Player | jab | | | | 4 | | | |
| ReqP02 | Basic Player Model | dpk | | 4 | 4 | 2 | | | |
| ReqP06 | Island Jumping | jab | | | | 4 | | | |
| ReqP08 | Direct Combat 1 | jab | | | | | 4 | | |
| None | Testing | jab/dpk/cob | | | | | | 4 | 4 |
| None | Work Estimates and Plan for MS05 | jab/dpk/cob | | | | | | 1 | 1 |

WEEK 13: 23.3.-29.3. WORKING TOWARDS MS03 AND MS04

Exact schedule to be determined.

WEEK 14: 30.3.-05.4. WORKING TOWARDS MS03 AND MS04

Exact schedule to be determined.

WEEK 15: 06.4.-12.4. WORKING TOWARDS MS04

Exact schedule to be determined.

WEEK 16: 13.4.-19.4. WORKING TOWARDS MS05

Exact schedule to be determined.

WEEK 17: 20.4.-26.4. WORKING TOWARDS MS05

Exact schedule to be determined.

WEEK 18: 27.4.-03.5. WORKING TOWARDS MS05

Exact schedule to be determined.

WEEK 19: 04.5.-10.5. WORKING TOWARDS MS06

Exact schedule to be determined.

WEEK 20: 11.5.-17.5. WORKING TOWARDS MS07

Exact schedule to be determined.

WEEK 21: 18.5.-24.5. WORKING TOWARDS MS07

Exact schedule to be determined.

WEEK 22: 25.5.-29.5. WORKING TOWARDS MS08

Exact schedule to be determined.

ASSESSMENT

The game features various possibilities of interaction with the game world and other players. Thus, it offers a very varied game play and diverse tactics a player can employ in order to ingeniously defeat its opponent. On the other hand, it should still be simple enough for everyone to learn the controls in a matter of minutes and enjoy playing.

A game world mainly consisting of lava is a challenge, but should reward us - and the player - with a beautiful, animated environment. Additionally, there is some cool physics involved when islands collide with each other or pillars.

We regard the game to be successful if players can make real use of the floating islands - and the involved physics - to fight each other.

PART 2 – PROTOTYPE

This chapter describes a first software prototype of the main game mechanics and shows our findings based on its evaluation. The prototype already incorporates the following concepts of the final game:

- Pseudo-randomly moving islands with colliding pillar interactions.
- Players who can move in the XZ plane and jump from platform to platform using a jetpack.
- Long-range attacks of players using the ice spike skill.
- Melee attacks of players.
- Visualization aids assisting players to navigate in the 3D space using shadows.
- Power-ups which are placed on islands.

We decided to approximate all the game elements with very simple geometric primitives. Although later islands might not have a flat surface in the final game, this simplified contact and collision detection a lot. Their movement is based on two forces: First, they get attracted by all the pillars whereas the force is quadratic to the distance. Second, we add a random force in each frame to prevent them from converging at one point.

The player's movement is divided into two parts: using the gamepads left analog-stick he can move in the XZ plane, while pressing A activates the jetpack allowing him to move up the y-axis. This movement is calculated by a simple acceleration of the jetpack, which is added to the player's velocity vector in each time step. Gravity acceleration works against the jetpack and keeps a player standing on an island – and (in the worst case) falling down into the lava. If players walk into each other they receive a minor velocity-based pushback. A stronger pushback is encountered, if one player hits another.

Shadows are realized by real-time shadow maps. At the moment, they use no interpolation in the look-up stage which leads to very jagged artifacts at steep angles. However, the sole purpose of a shadow implementation at this stage was to determine whether or not shadows would serve well to support players navigating on the islands.

The ice spike implements a homing mechanism. After the spike is set off, the spike gets slightly pulled into the direction of an enemy. However, we have an upper bound for this force in order not to make aiming too easy.

For easier tracking of hits (either melee or through ice-spikes) appropriate sounds were also added.

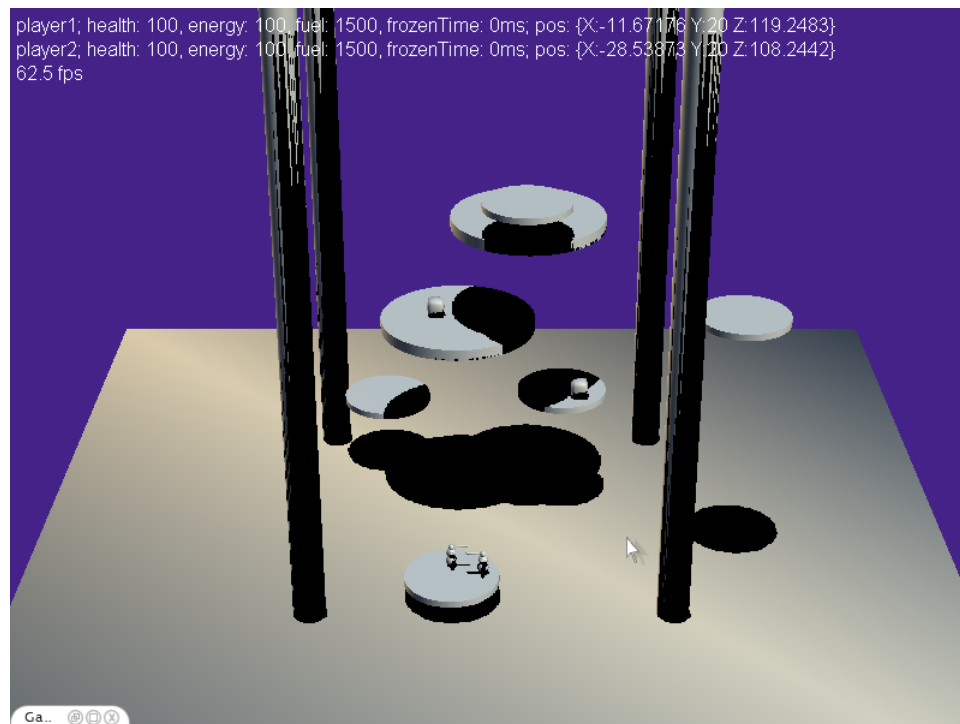
EVALUATION

We have tested the game (and will continue to do so a lot within the upcoming days) with respect to the following criteria:

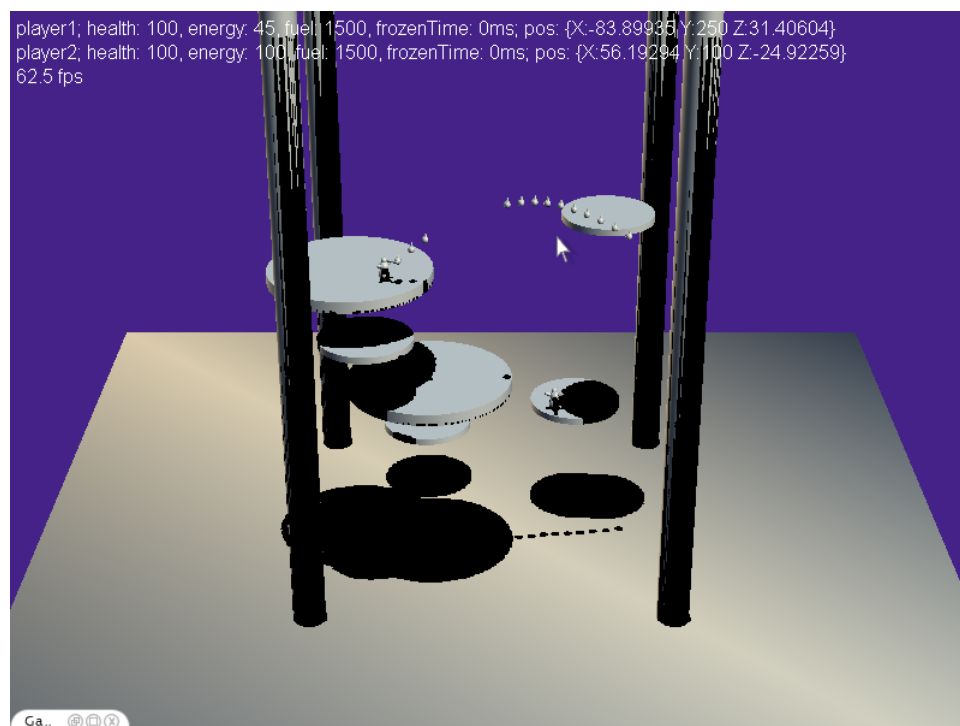
- Is it easy and intuitive to move the player and perform attack actions?
- Does the core game play make fun, even after playing it for several minutes?

While the latter question is common and crucial for every game concept, the former one is one raised by multiple reviewers of our original concept. By testing this point in a very early phase, we want to react properly to the feedback we've got in the first stage.

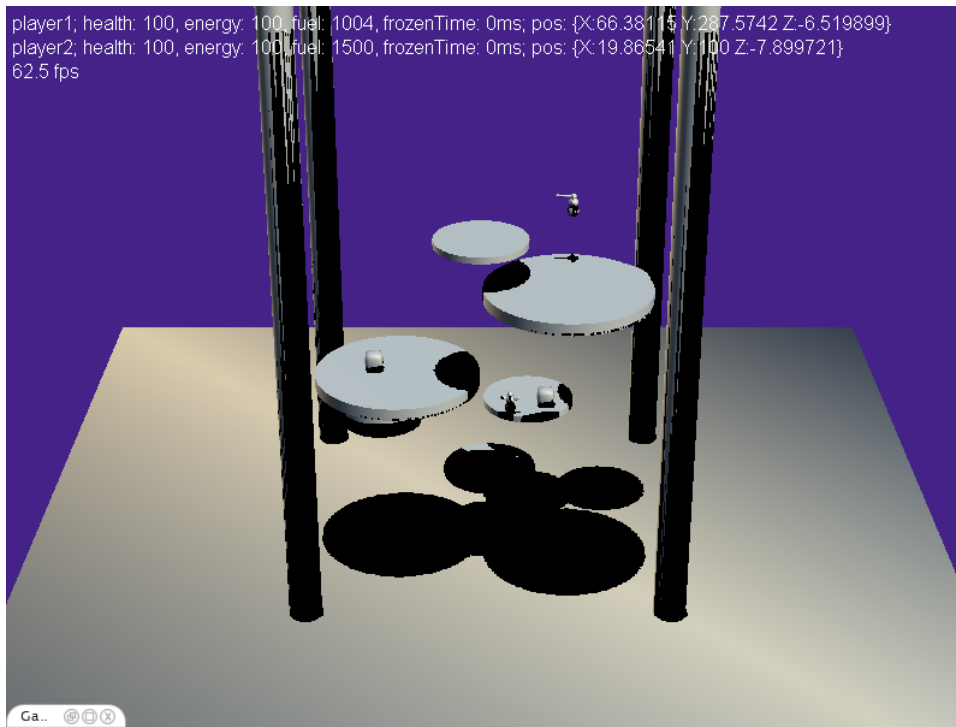
GAMEPLAY SCREENSHOTS



Two players are standing on a moving island. Two collectable power-ups are on other islands. A player's health, energy and fuel level is currently shown as a text label. Later, this will be replaced by a graphical HUD.



A player is shooting a bunch of ice-spikes, although missing his enemy.



A player is using his jetpack to move to the smaller, upper island. As visible in the text on top, using the jetpack needs fuel.

FINDINGS

POSITIONING

It is still quite tough to position yourself in the 3D environment. To make the task easier, we added shadows to enable the player to look at the projection of the island and his robot to more easily track his position. To control a player hidden behind an island or another object, we will implement some feature showing his contours projected onto such an object. This could also be combined with shadowing in a way a player gets a marker on all islands below and above him.

The addition of shadows unfortunately leads to the problem that a player is completely in the dark and not visible. Nevertheless, this can easily be solved by having the player emit his own light and adding ambient lights.

Finally, we will have to do further experiments with the angle and focal length used for the camera to reduce positioning problems.

PLAYER MOVEMENT

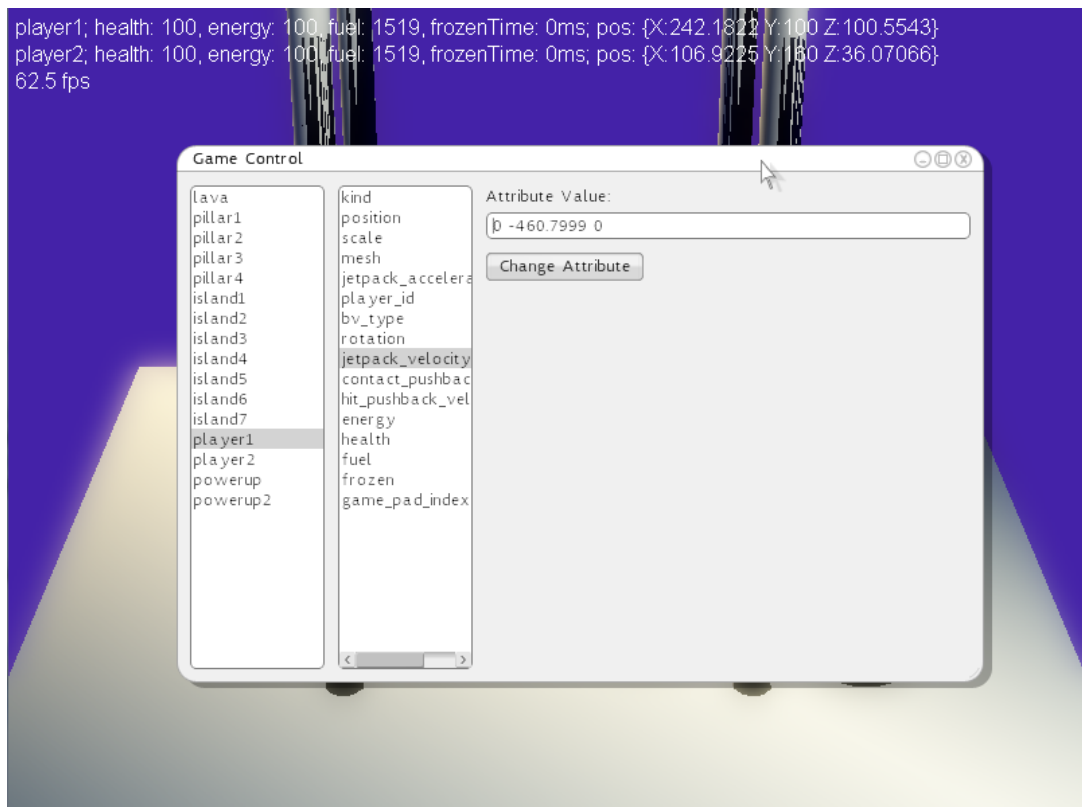
Currently, a player can walk off an island, which is a very unfortunate and leads to sudden death. It would make sense to only allow the player from falling of an island if he explicitly uses his jetpack or other means of traveling between the islands. Otherwise, He should not be able to walk beyond the edge of the island. We will implement this behavior in a further version.

ISLAND TRAVEL

Using free jetpack movement, it is nearly impossible to move between islands, because of mentioned 3D positioning and tracking problems. Therefore, a more passive approach (like selecting the island and automatic flying) should be taken. This could also have the advantage that a player can look around and shoot spikes at his opponents while flying. Additionally, the path on which an island moves should be visualized in the future (for instance by small rings of dust). Therefore the path of an island will have to be fixed or at least calculated in advanced to visualize where an island will move to some time from now.

PARAMETER TUNING

Without the need to drastically change certain implementations and aspects of the game, one can heavily improve the experience by tuning parameters (e.g. attack damage, gravity or jetpack acceleration). To simplify this task and in order to allow fast testing of different parameters, we implemented a game console which shows the state of all active entities on the screen and enables the user to directly manipulate them:



An in-game console which allows the modification of parameters and attributes of all the present entities.

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 has 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

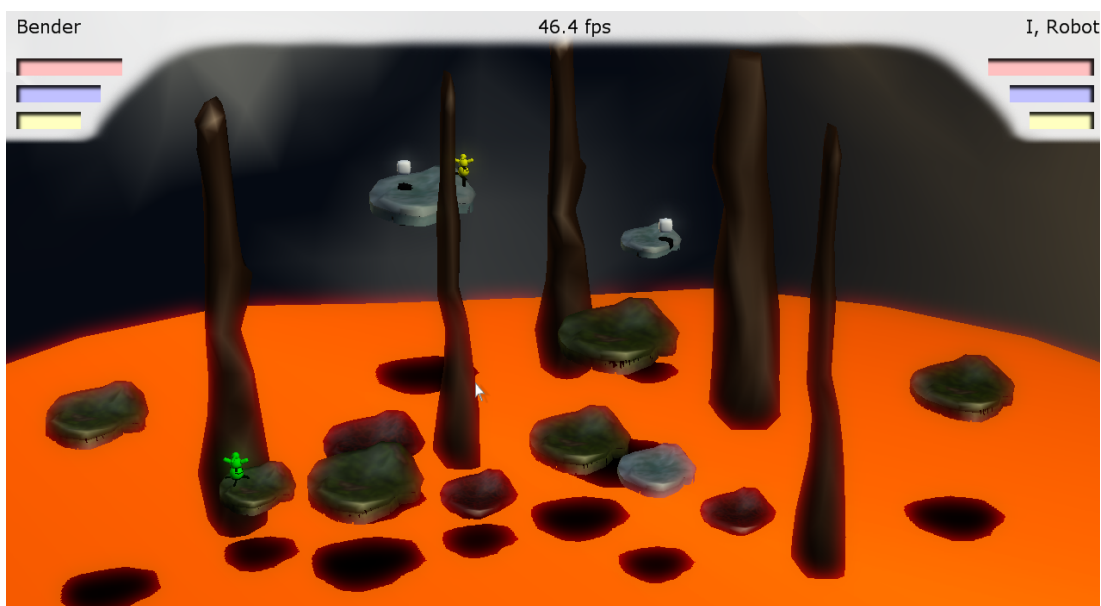
Some problems still remain, such as:

- 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.

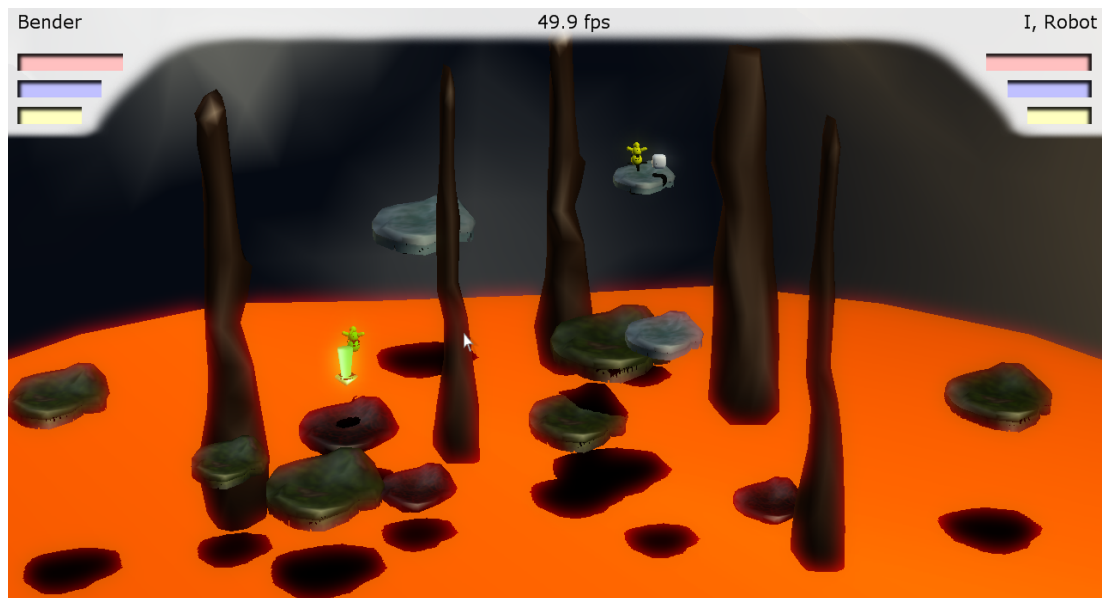
- Islands don't collide with the cave at the back, nor are they stopped from leaving the screen to the left, right or bottom.
- On island attraction, some collision response is not correct: Islands can sometimes go through pillars.

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 and 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.

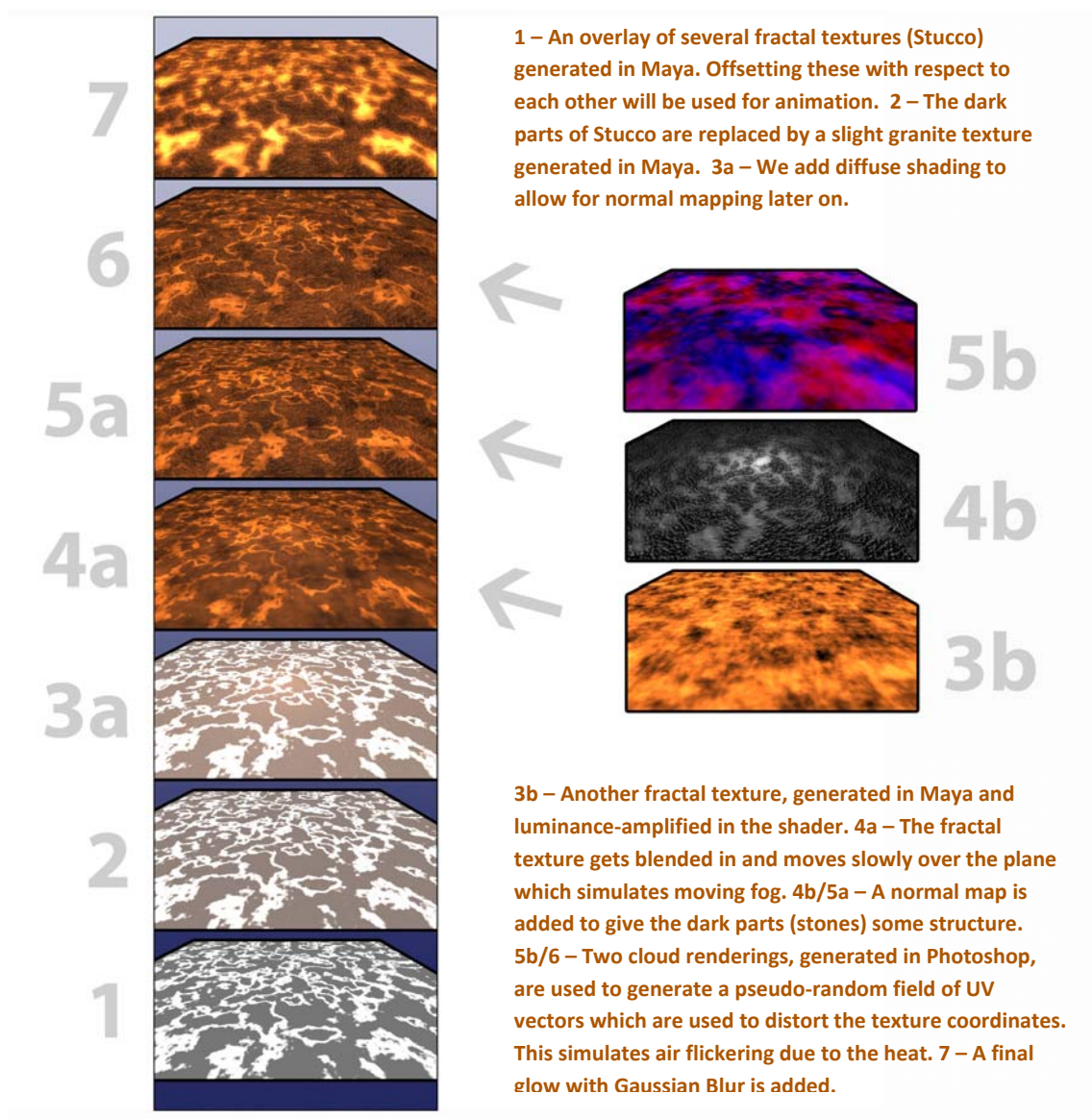
The content pipeline creates all three collision volumes for each triangle mesh. The level designer then chooses which bounding volume is assigned to a given entity.

PRODUCTION EXAMPLE - CREATING LAVA SURFACES

FIRST APPROACH: LAVA PLANES

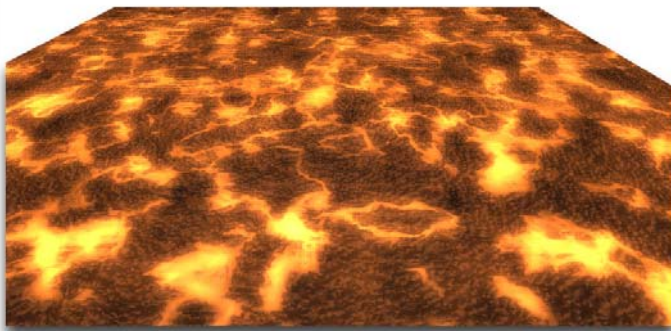
We would like to show an example of a graphical element which we consider to be crucial for a credible ambience of our game. This serves both as a documentation for our own reference and for a work report for the lab.

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.9icg.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.



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

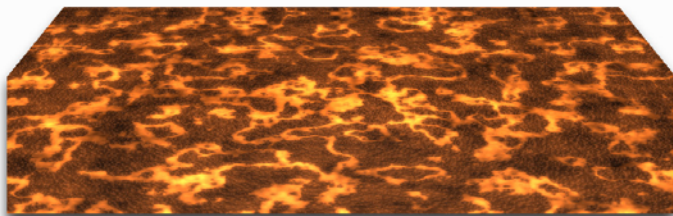


The effect of the shader described above looks already quite pretty when seen from a perspective projection like the one in the picture to the left.

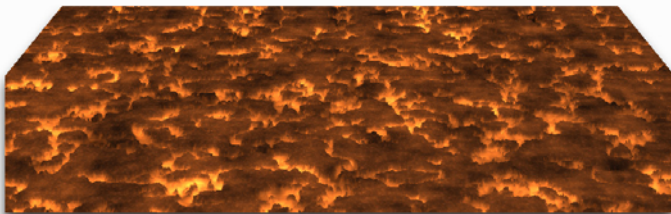
However, we had to find out that the effect owes much of its dramaticism to the wide-

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



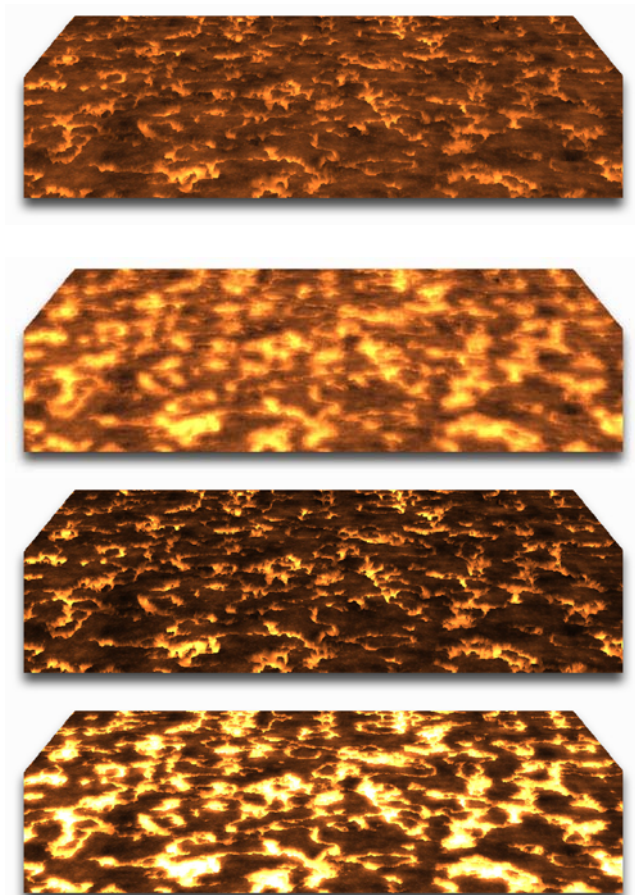
the angle between the camera and the ground plane is relatively flat in our setting, we thought that it would be nice to have some actual geometric structure in the lava instead of just plain normal mapping. To find out if this would help, we took an implementation of Parallax Occlusion Mapping and included 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.

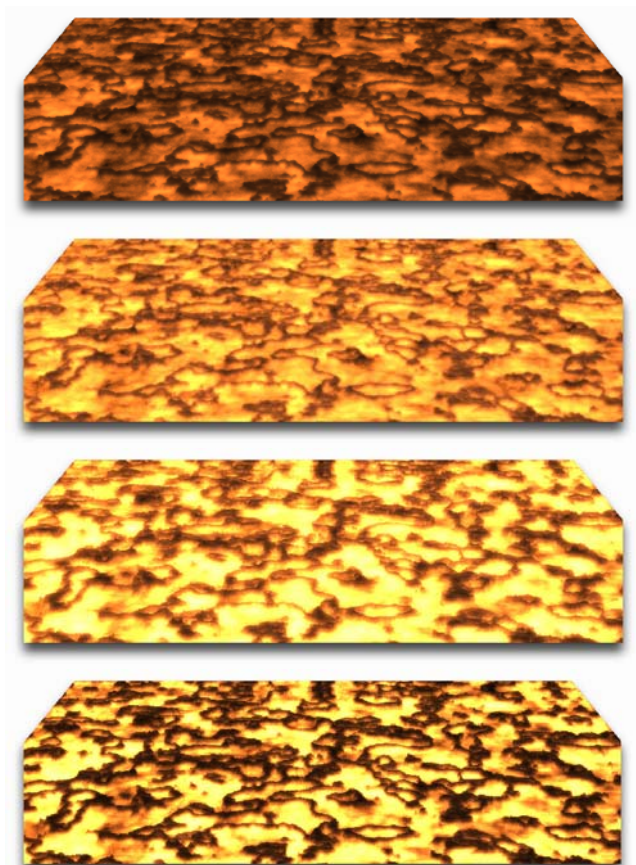
In this set, we inverted the depth effect of PO mapping by negatively scaling the occurring gradient term. The Stucco map which combines the textures (see earlier) is still unchanged, though.

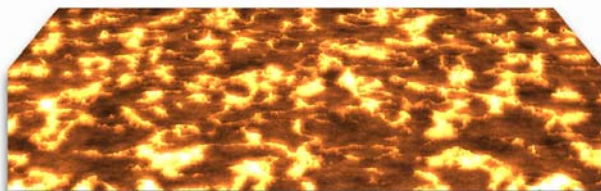
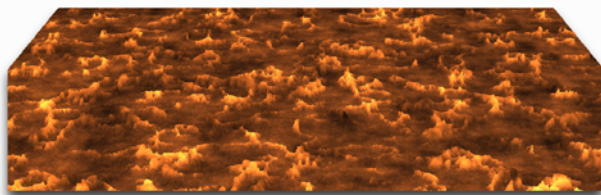
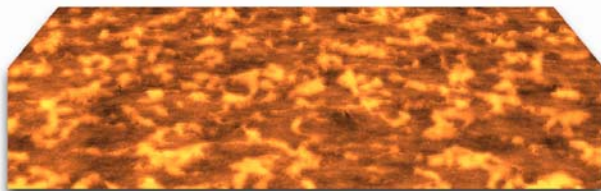
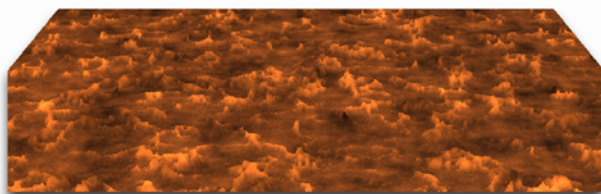
First – low glow radius and strength, linear tone mapping.

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.

