A Planet in Ashes - Escaping a Nightmare

Group 42
Bernhard Pointner **01527081**
Dominik Scholz **01527434**

Story

After an attack by foreigners the planet in our story lies in ashes. The destroyed environment is shown. It seems like everyone died and there is no possibility to survive on this planet anymore. The sun sets. There is still one intact spaceship left in the colony. The path to it is dangerous (cringey overload) but our hero manages to reach the ship and escape the desolate place.

Implemented

- ✓ Illumination
- ✓ Model/Texture loading
- ✓ Free moveable camera
- ✓ Advanced scene
- ✓ Scene settings loaded via parameters of config file
- ✓ Effects
- ✓ Audio
- ✓ Font/Text Rendering

Keyboard keys

KEY P	PLAY/PAUSE	Starts/stops the automatic camera movement through all shots (Video)
KEY RIGHT	SCENE SWITCH	Switch free-moveable camera to next scene shot
SPACE	CAM SWITCH	Tabs through all available cameras. The first one is the simplified path following camera and the second one the free moveable camera (debug cam).
KEY R	RELOAD SHADER	Reloads all relevant shaders
KEY C	RELOAD CONFIG	Reloads the main config file and therefore the whole scene (kind of restart)
KEY N	RELOAD SHOT	Reload current active camera shot (scene)
KEY X	RELOAD LAYOUT	Reload current active layout used by a camera shot

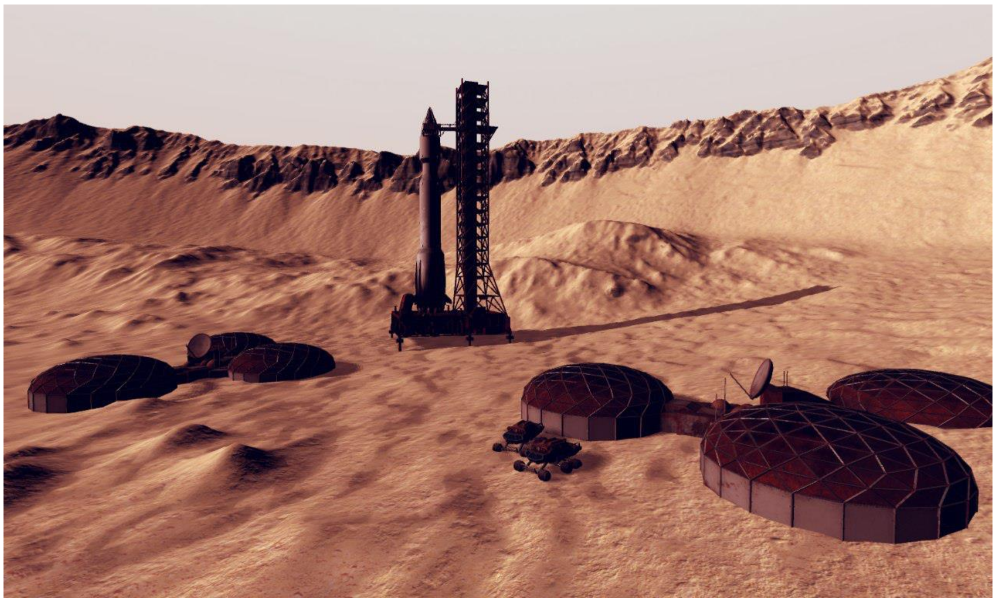
Config settings (Assets/config.json)

width	1920	Set video width
height	1080	Set video height
shadow-resolution	8192	Set shadow map size (higher => more accurate)
brightness-adjust	0.0	Reloads all relevant shaders
fullscreen	true	Enable/Disable fullscreen mode
auto-start	true	Enable/Disable auto start of video
audio-enabled	true	Enable/Disable audio

Inspiration

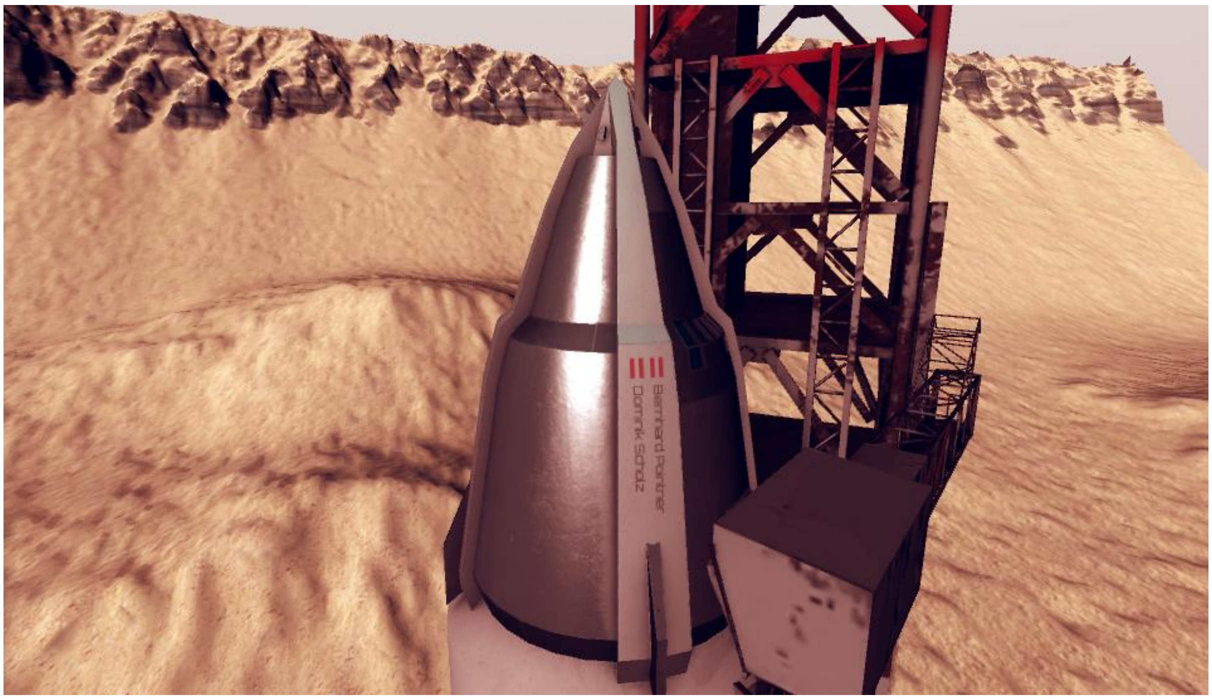
Source: https://www.artstation.com/artwork/nyaLr

Result



EFFECTS

1) PBR/PBS/BRDF



Description:
We implemented a PBR shader that is inspired by the Disney BRDF shader. We reduced the parameter to allow for real time usage (metallic, roughness), for that we closely followed the implementation of the real-time PBR shader from epic games (Unreal Engine). We implemented the cook-torrance illumination model for the specular term. We also used the skybox for the reflection of the metallic term. Additionally we baked ambient occlusion maps in substance that are applied to the shading afterwards. (see main.frag)

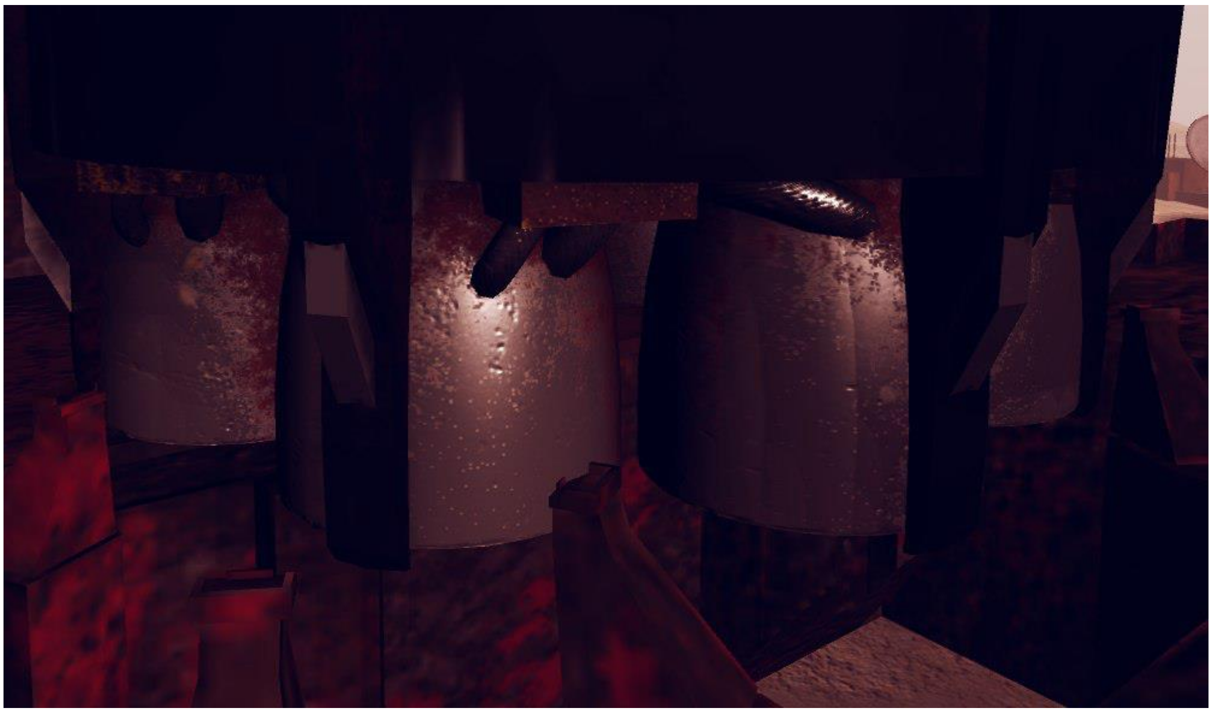
Source:
https://disney-animation.s3.amazonaws.com/library/s2012_pbs_disney_brdf_notes_v2.pdf
https://blog.selfshadow.com/publications/s2013-shading-course/karis/s2013_pbs_epic_notes_v2.pdf

2) MSAA (Multisampling)

Description:
Just as little improvement we used a multisample-framebuffer for the main rendering that gets blitted to a child buffer (used because our multsample framebuffer is not the backbuffer)

Source:
https://learnopengl.com/Advanced-OpenGL/Anti-Aliasing

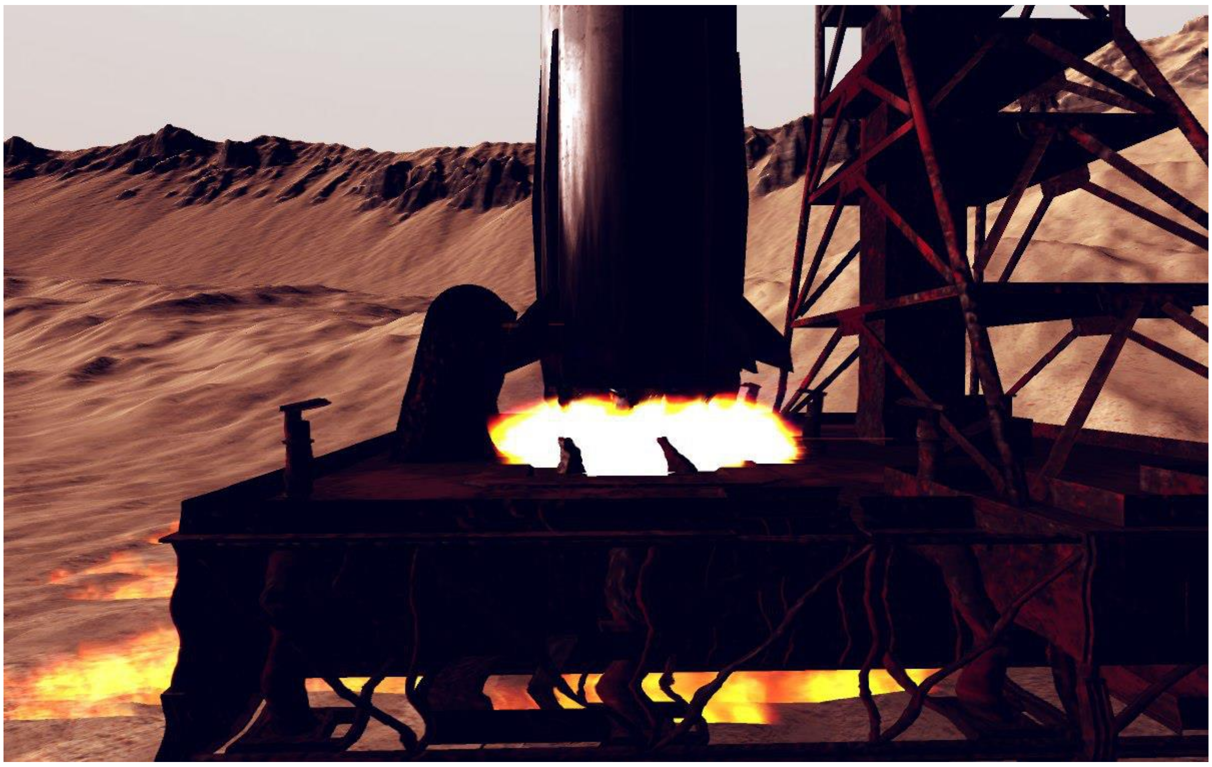
3) Normal Mapping



Description:
The normal mapping is used for displaying bumps on the terrain. Those bumps reduced the needed complexity of the geometry of the terrain model => speed increase

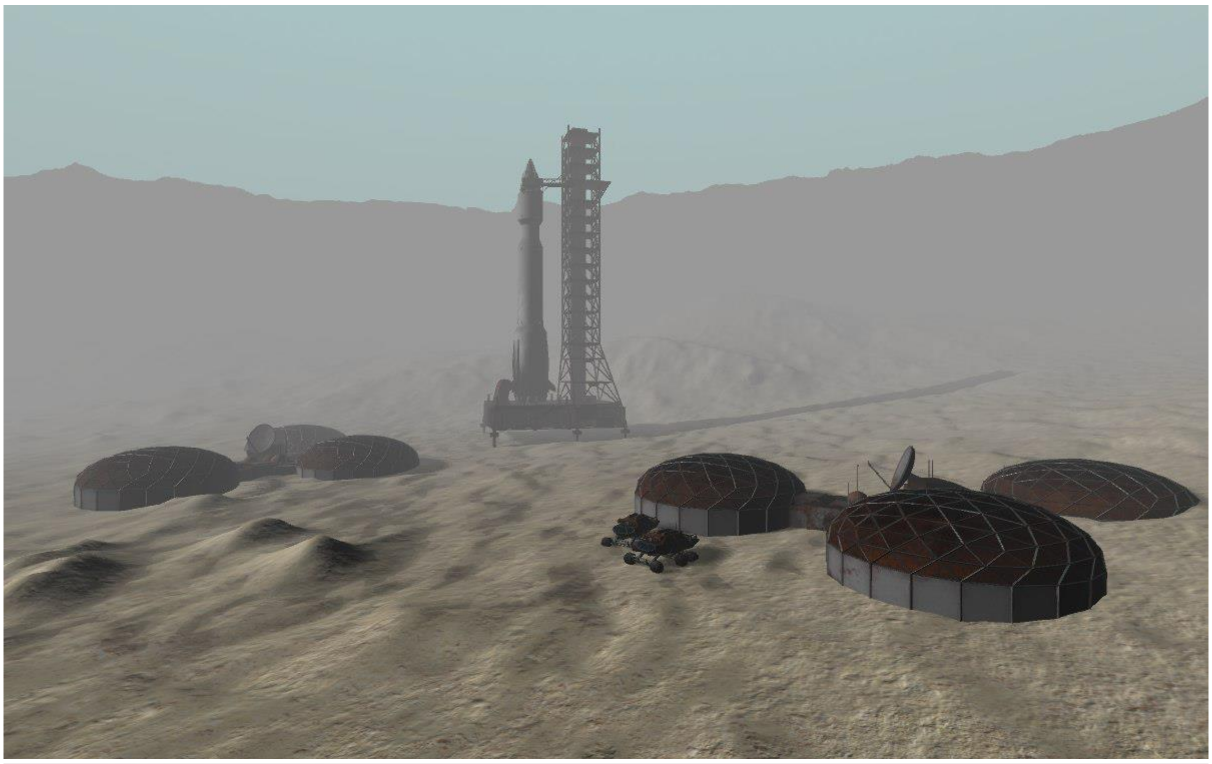
Source:
https://learnopengl.com/Advanced-Lighting/Normal-Mapping

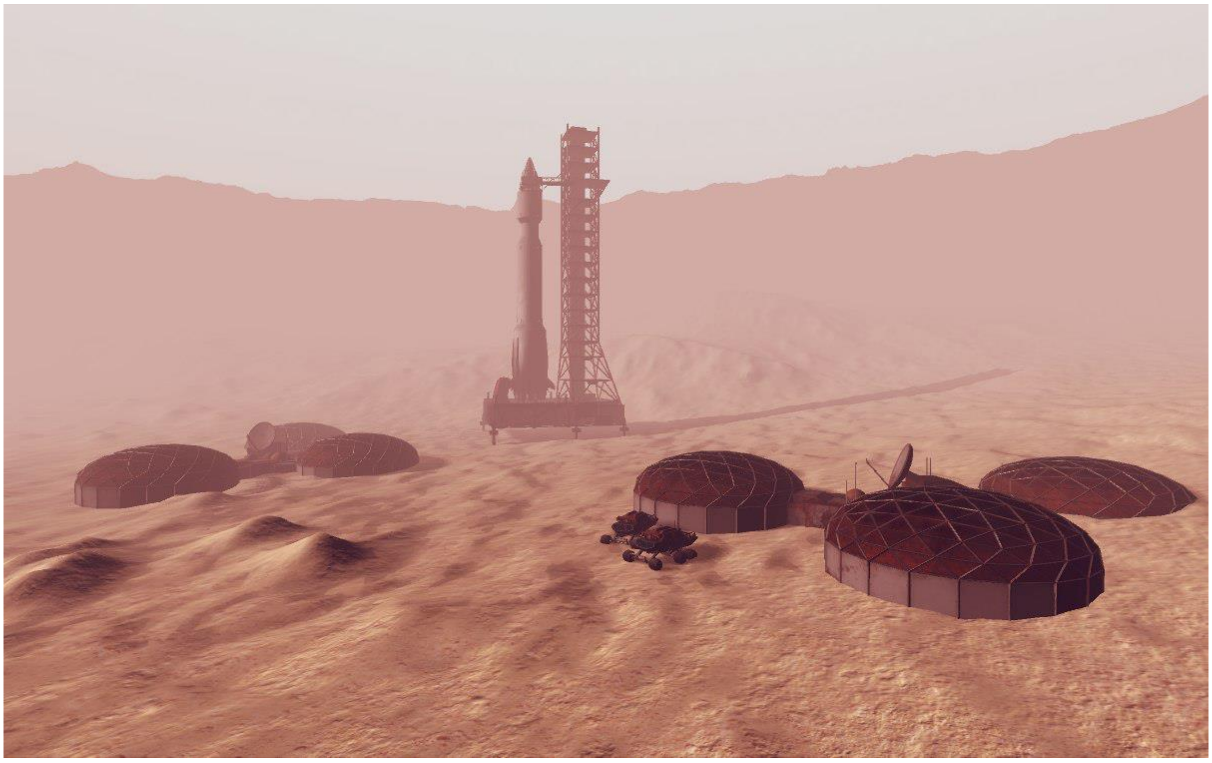
4) Heat Distorsion

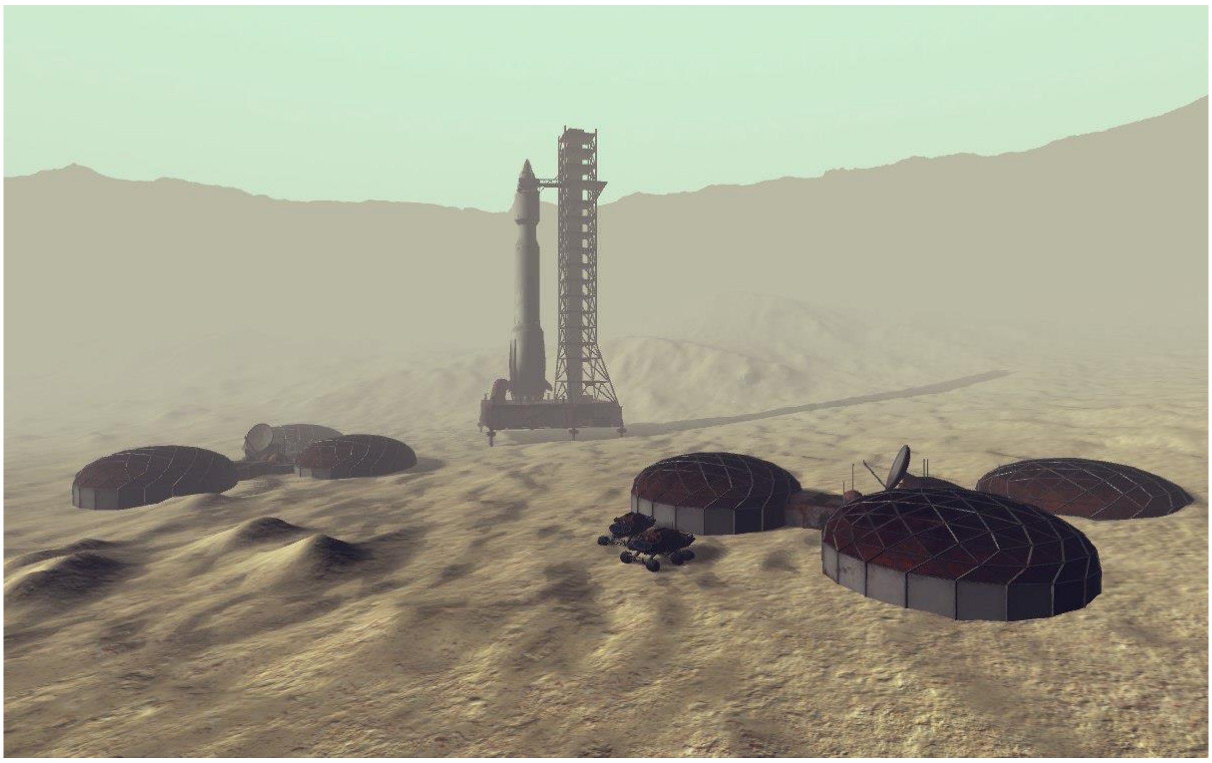


Description:
We implemented a simple heat distortion affect by drawing proxy geometry (called “disorters”) into a framebuffer attachment. These disorters are used as mask for a displaced lookup from the main framebuffer. The lookup is displaced by sine curves and a noise texture.
Source:
https://tympanus.net/codrops/2016/05/03/animated-heat-distortion-effects-webgl/

5) Color Grading

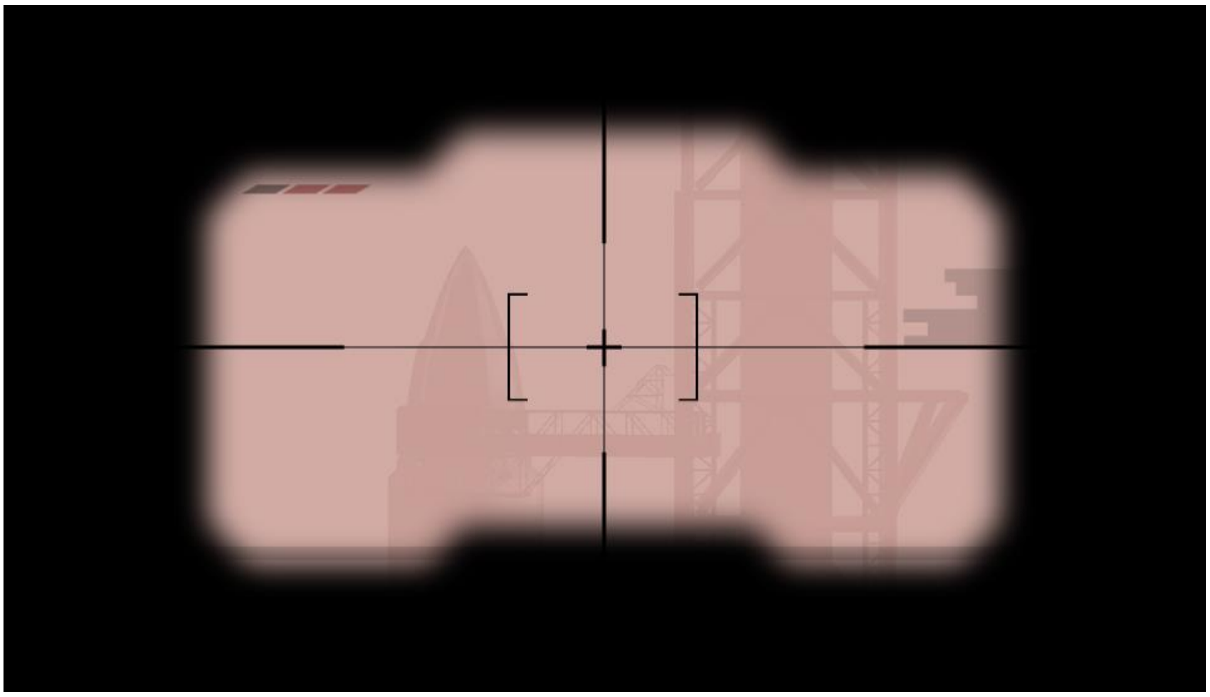






Description:
We implemented color grading with look-up-tables (LUTs). We used 16x16x16 LUTs like used in the unreal engine (in contrast to NVidias 32x32x32 because that is used for offline rendering/video processing). 16x16x16 seemed a good choice to approximate color grading transfer functions (also because game engines use this size). We used the improved way of uploading a 16x256 pixel 2D-texture as 3D texture which makes it easy to configure the LUT. The workflow to generate a LUT looks as follows: make a screenshot of the application using a default LUT (identity) -> apply filter in a image editing program -> apply the same filters on the LUT image -> upload new LUT image. Our shot engine also supports interpolation of LUT as two LUT are uploaded to the shader (ss.frag).
Source:
https://developer.nvidia.com/gpugems/GPUGems2/gpugems2_chapter24.html

6) Depth of Field (currently still under development)

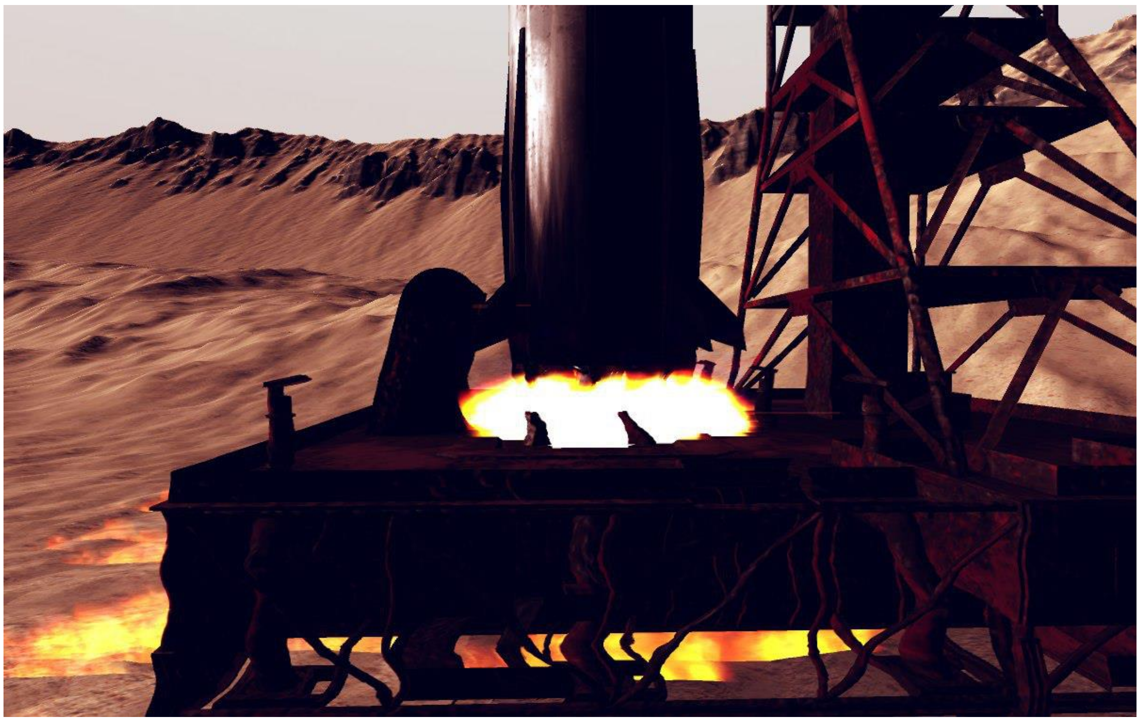


Description:
Depth of Field should have been used for an intro shot where the actor (third) stands far apart from the space ramp. Therefore, he uses a telescope to take a closer look of the space ramp and the shuttle. To get a realistic feeling, the effect depth of field should be used to blur the near environment of the actor and to focus (sharp image) the space ramp from a far distance.

Currently, we implemented all steps until 4.3 of the dof source but we were not able to finish the effect until the submission deadline. However, anyway this effect was listed as an optional effect in the effect list of the initial submission document.

Source:
https://catlikecoding.com/unity/tutorials/advanced-rendering/depth-of-field

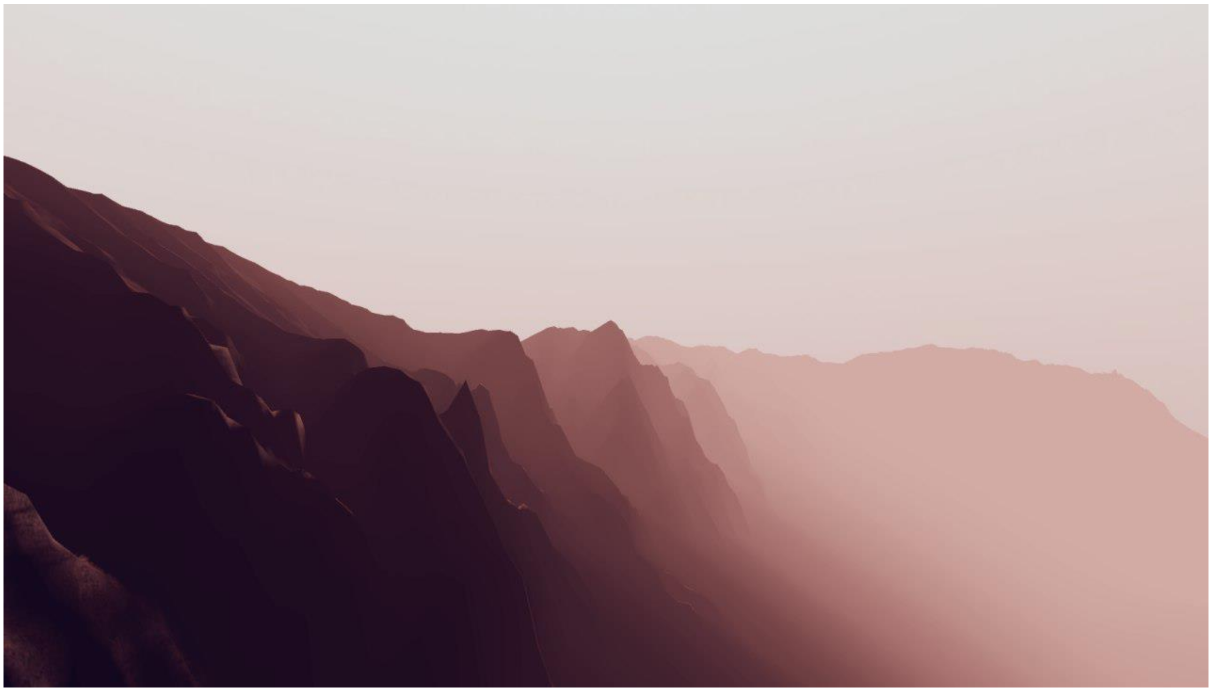
7) GPU Fire particles



Description:
The fire effect is implemented as gpu particles. The geometry of the particles is generated in the geometry shader. The physic of each particle is defined by the compute shader which also calculates the particle generation. The fragment shader uses a 2D fire sprites texture with many different small sprites of the fire status in each program sequence.
This effect is a replacement for the motion blur effect mentioned in the initial submission. We think that this effect fits better to the story respectively is more relevant for realistic feeling. However, due to the complexity of the effect (GPU, geometry shader, compute shader) a replacement, we think, this effect is fine as a replacement.

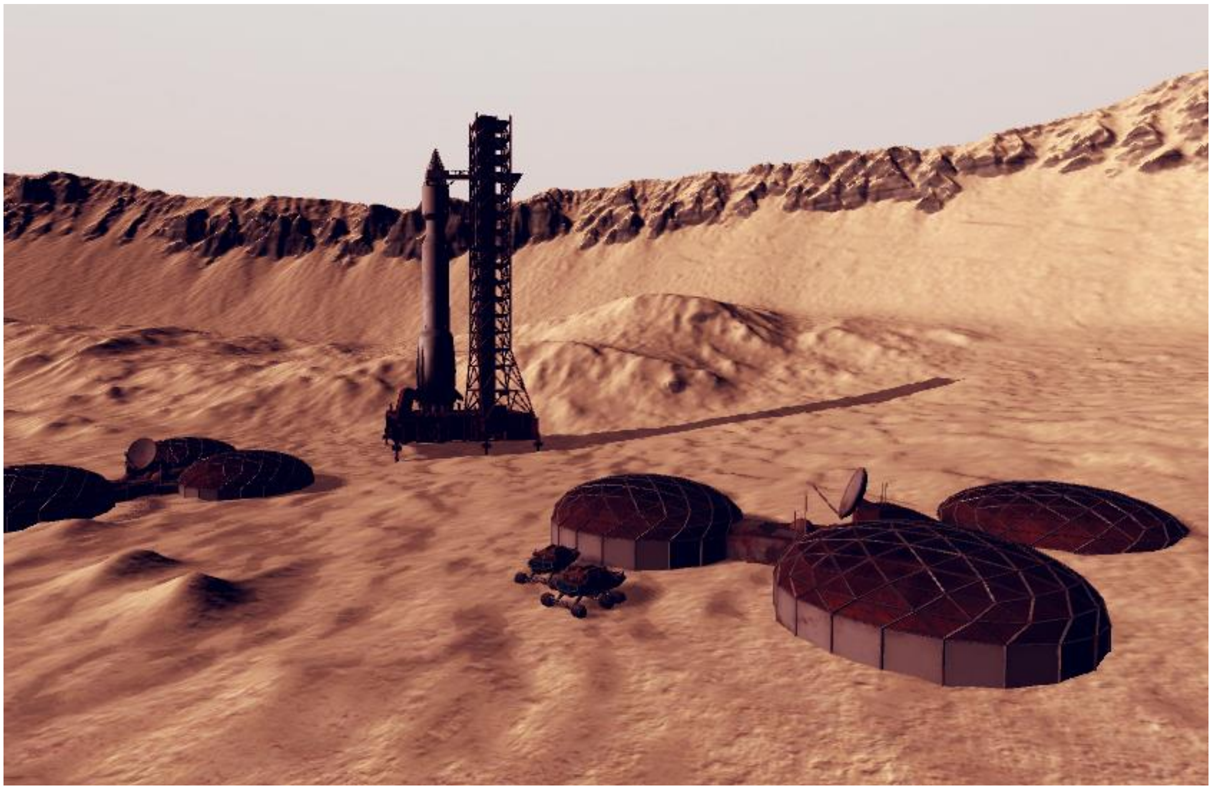
Source:
https://learnopengl.com/In-Practice/2D-Game/Particles
+ lecture slides for compute shader integration

8) Range-based Fog



Description:
The used fog is implemented with a squared increase of density dependent on the distance between the camera position and each vertex.
Source:
http://in2gpu.com/2014/07/22/create-fog-shader

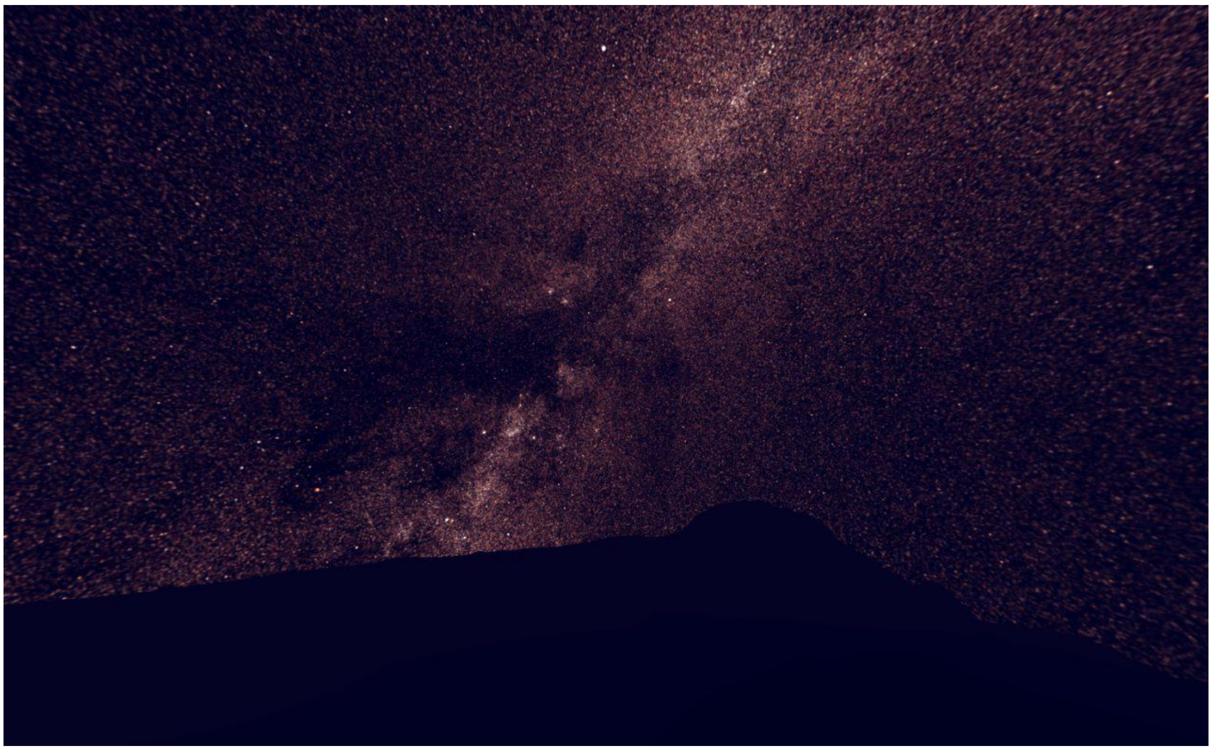
9) Shadow Mapping (directional light)



Description:
Because our scene is outside any room, directional-based light shadows are used instead of point shadows. The shadow map is generated in the main framebuffer which draws all objects each scene.

Source:
https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping

10) Skybox



Description:
The skybox is geometry-based and uses a cubemap planes for each face of the cube. Which skybox each is used by each scene is defined in the configs of layout. Because the layouts changes during the video, also the skyboxes changes (night -> daylight)

Source:
https://learnopengl.com/Advanced-OpenGL/Cubemaps

Implementation

Scene/Shot Engine

We implemented a data driven approach to organizing our scene in shots (units of camera movements) and layouts (model arrangements). This allowed for quick debug cycles. The shots also allow to set shader parameters and even interpolate them. This is used in the scene, to transition camera positions, camera settings, transistion LUT, rotate skyboxes, change effects, reposition effects, place disorters,….. everything can be configured on runt-time by adding/configuring new shit files.

Used tools

Visual Studio 2017, Blender (Models), Substance (Textures)

Audio

FMOD SoundClass

- bensound-birthofahero.mp3
  Source: https://www.bensound.com/royalty-free-music/track/birth-of-a-hero

Texture

- cam.png
- default_lut.png
- lut.png
- dirt.jpg
- dirt_normal.jpg
- houses_Base_Color.png
- houses_Metallic.png
- houses_Mixed_AO.png
- houses_Normal.png
- houses_Roughness.png
- mobilelauncher_railings_mat_Base_Color.png
- mobilelauncher_railings_mat_Metallic.png
- mobilelauncher_railings_mat_Mixed_AO.png
- mobilelauncher_railings_mat_Normal.png
- mobilelauncher_railings_mat_Roughness.png
- rocket_merged_body_Base_Color.png
- rocket_merged_body_Metallic.png
- rocket_merged_body_Mixed_AO.png
- rocket_merged_body_Normal.png
- rocket_merged_body_Roughness.png
- rocket_merged_bottom_Base_Color.png
- rocket_merged_bottom_Metallic.png
- rocket_merged_bottom_Mixed_AO.png
- rocket_merged_bottom_Normal.png
- rocket_merged_bottom_Roughness.png
- rocket_merged_spaceship_Base_Color.png
- rocket_merged_spaceship_Metallic.png
- rocket_merged_spaceship_Mixed_AO.png
- rocket_merged_spaceship_Roughness.png
- rocket_merged_spaceship_Normal.png
- rover_merged_Adds_Base_Color.png
- rover_merged_Adds_Metallic.png
- rover_merged_Adds_Mixed_AO.png

- rover_merged_Adds_Normal.png
- rover_merged_Adds_Roughness.png
- rover_merged_body_Base_Color.png
- rover_merged_Body_Metallic.png
- rover_merged_body_Mixed_AO.png
- rover_merged_Body_Normal.png
- rover_merged_Body_Roughness.png
- terrain_side.png
- terrain_normal.png
- terrain_top.png
- cubemap
- cubemap1 – NASA images

Models

- hirise1.fbx (Terrain) – NASA HiRise data
- hirise2.fbx (Terrain) – NASA HiRise data
- houses.fbx (Space houses) – Modelled & Textured by Dominik Scholz
- ramp.fbx (Space ramp) – NASA public model, textured by Dominik Scholz
- rocket_merged.fbx (Space rocket) – Modelled & Textured by Dominik Scholz
- rover_merged.fbx (Space rover) - Modelled & Textured by Dominik Scholz

Libraries and sources

[1] Assimp, http://assimp.sourceforge.net/
[2] OpenGL, https://www.opengl.org
[3] GLFW, http://www.glfw.org
[4] GLEW, http://glew.sourceforge.net
[5] DeviL, http://openil.sourceforge.net
[6] FreeType, https://www.freetype.org
[7] Sound FMOD, https://cuboidzone.wordpress.com/2013/07/26/tutorial-implementing-fmod/