Project Dervish - Tile creation

1. Creation of the tiles

1.1 Morph Tile

Starting with an even mesh we sculpt the basic shapes using the Freeform tools in 3ds Max. Sculpting is done only in the middle of the tile, so the beginning and end are not modified. After that the road part is cleaned to avoid unwanted uneven results where the characters would clip through (see Image 1). This is done using a script to automate this step.

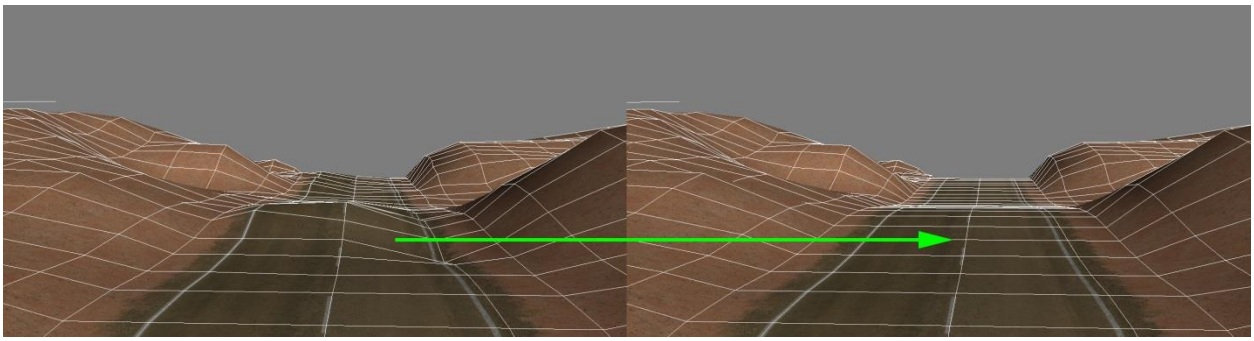

Image 1 Adjusting the road part of the tile.

With this tile finished, it is used as a morph target for the final tiles that will be used in the game and additionally, we set the max scene in a way that two tiles following directly after each other are skin wrapped to the morph tile. This way, we shift the sculpted part to the beginning and end of the tile.

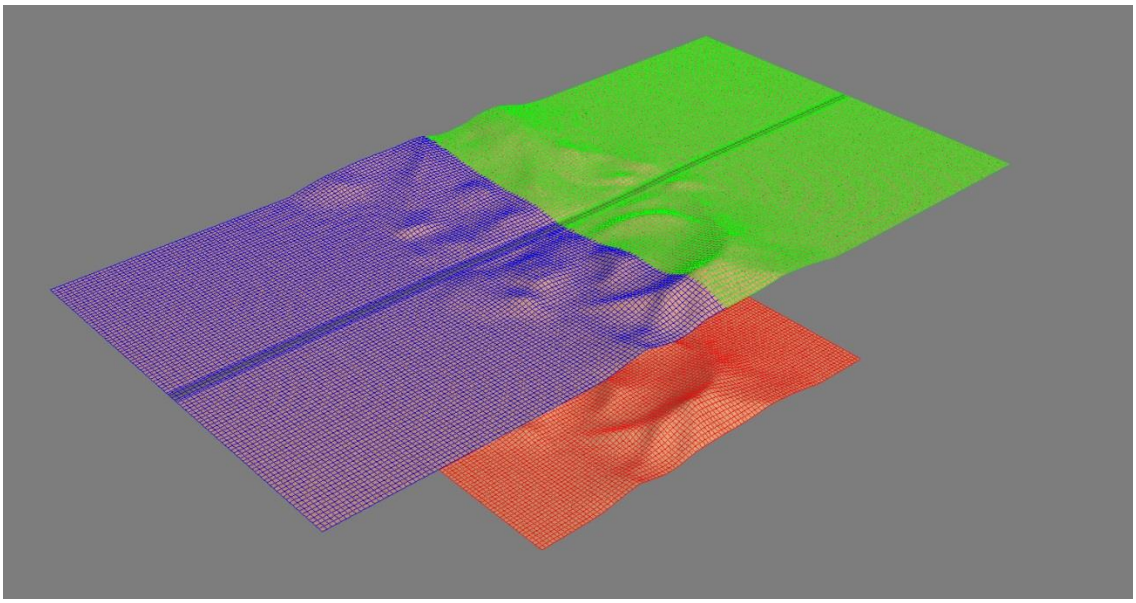

Image 2 Blue and green were skin wrapped to the red one to shift the sculpted part to the end and beginning of the tiles.

The two new generated tiles are again used as targets for the final tiles. This way, one sculpted tile equals three morph targets. Note that with creating one tile you exponentially increase the total amount of possible combinations between the targets (with only three sculpted there are already 64 possibilities).

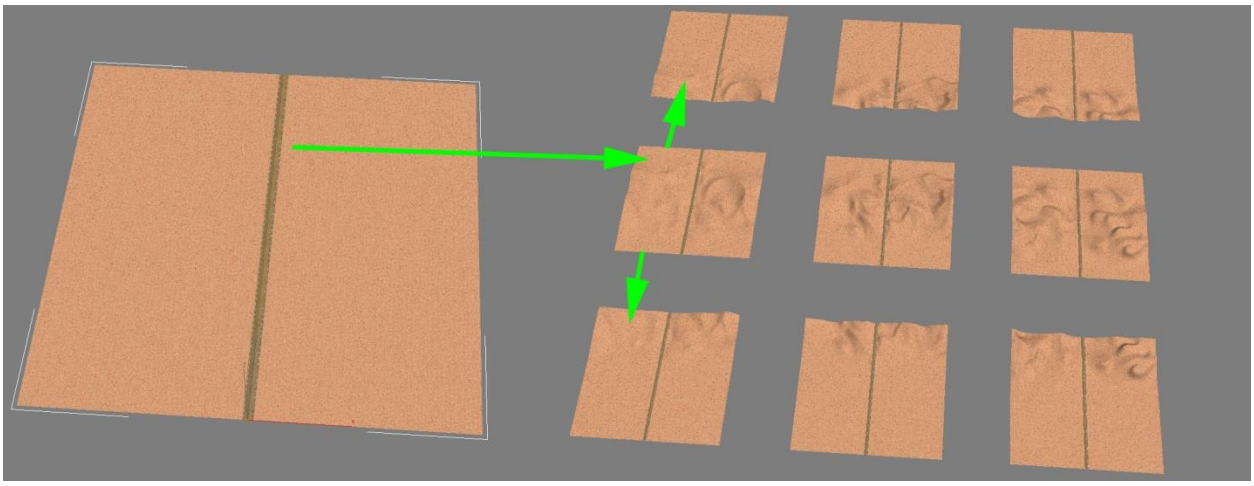

Image 3 Finished target tiles

In order to provide even more variation we apply a path deformer on the top morph tile. This way a tile can have a slight curvature, again creating a vast number of possible combinations.

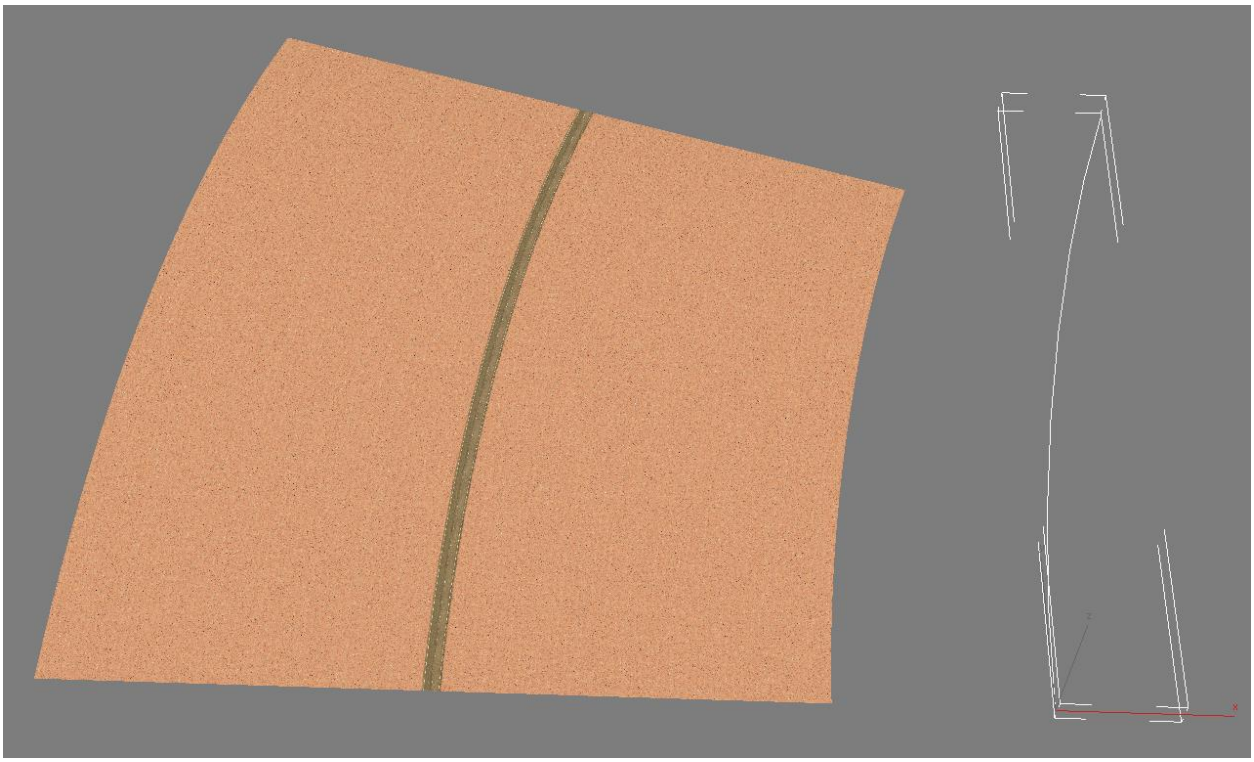

Image 4 One possible curvature for a tile

1.2 Game tile

The game tiles were created based on the morph tile, with reduced and increased polygon density wherever it was needed. In this case, the density decreases the further away from the road you get. Each game tile has two materials assigned to it, a material for the ground where the tiling is later defined in the shader (see 2.1) and a road material that uses a tiling texture for the road and borders of the texture, so that they match exactly to the part of the ground texture. This way no texture seam is visible.

All game tiles (LoD0 – 2) are skin wrapped to the morph tile so when you create a combination all game relevant data is automatically created with it.

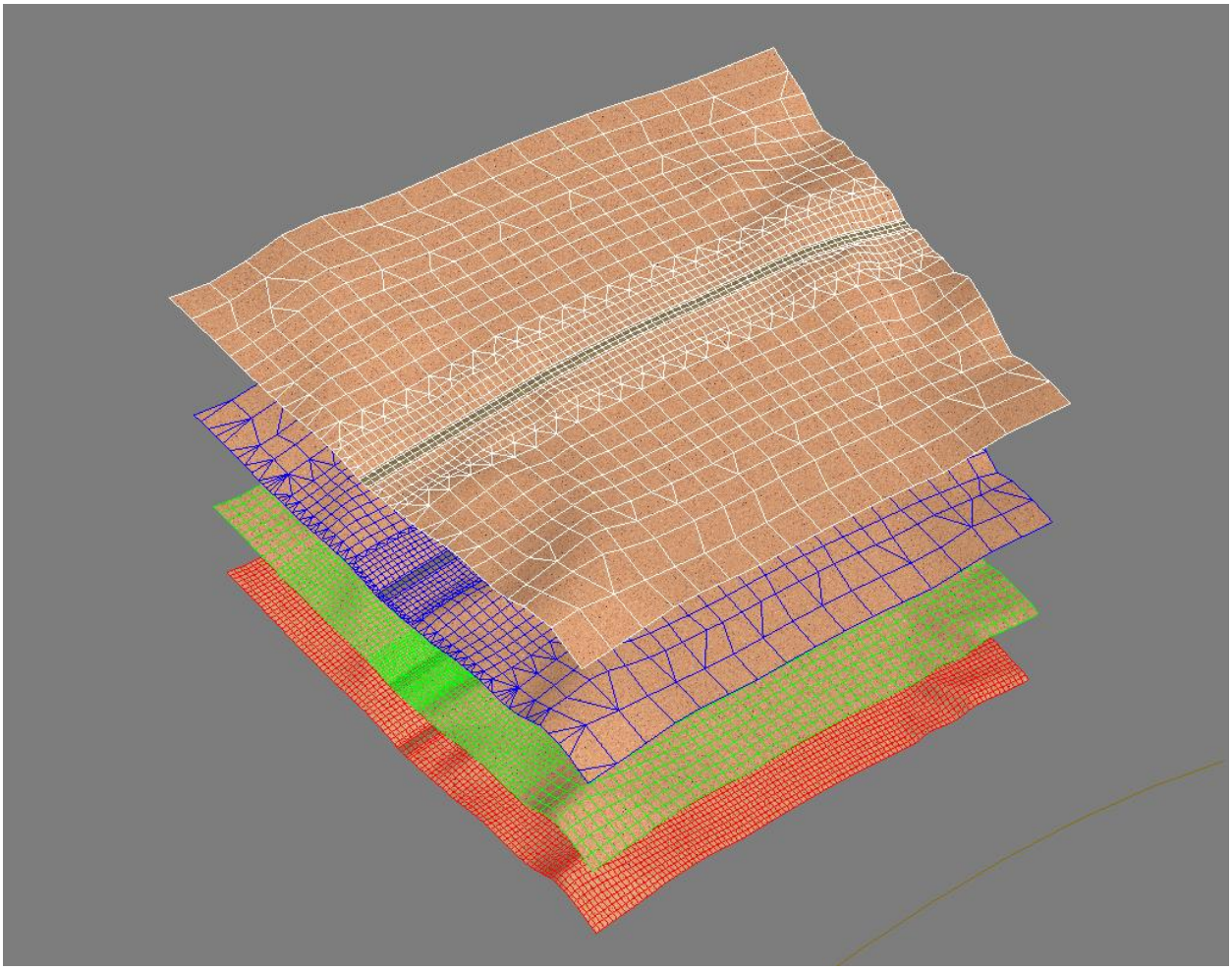

Image 5 Skin wrapped LoDs

As the skin wrap works best with the same geometry, the LoDs tend to have some slight inaccuracy which is not noticeable in the middle of the tile but leads to visible holes between two connectable tiles. Additionally, the normals of each tile must be adjusted so no lighting seam is visible, with the correct tiling applied to the textures no texture seam is visible. To automate this process we rigged the morph

tile in a way that reference tiles can easily be placed and the before mentioned issues fixed, again we wrote a small script to automate this step.

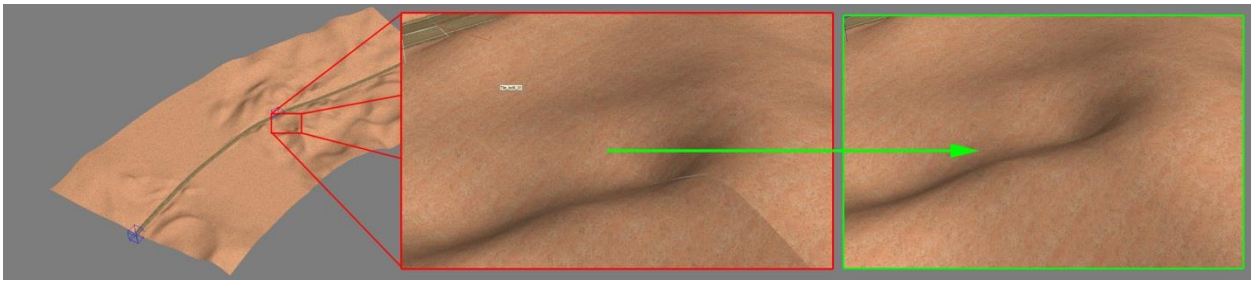

Image 6 Unadjusted and adjusted seams between tiles

To reduce tile stitching glitches the start edges of the tile are extended down and the normals of the new created polys are rotated so they also face upwards.

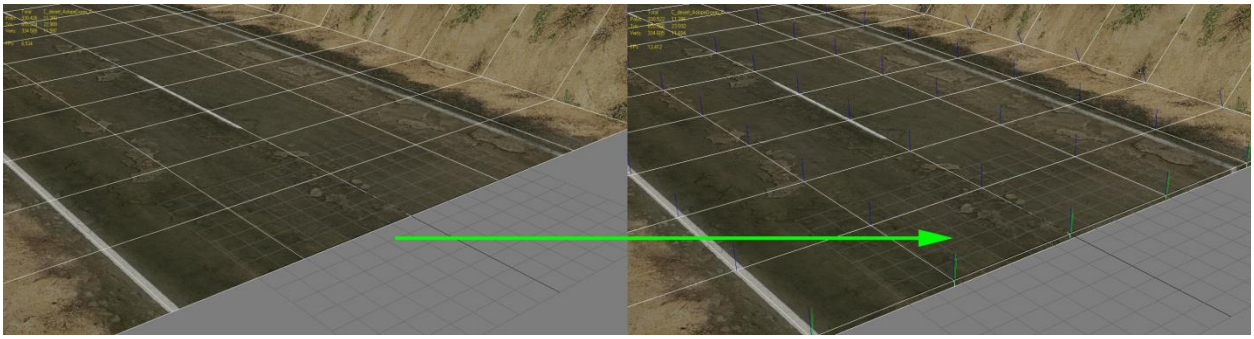

Image 7 Extended polys and adjusted normals

After that the tiles are exported into the editor.

2. Dressing of the tiles

Each tile has its own zone in vForge named exactly like the tile for easier management and for the code to detect what the next tile can be.

2.1 Materials

We assign a special shader that allows us to have up to four different textures, each with different tiling settings blended based on a mask texture (see 2.2.2) to the ground material.
To add variation on the road material we use a different approach (see 2.2.4) so a less expensive shader is assigned to it.

2.2 Object dressing

1. Each tile has its own zone using a special naming convention: <tile name>_<tile description>_zone. This is important as the tile name part also defines which camera dolly is used.

2. A PDAnchorPoint must be placed exactly at the center of the start of the tile and named “Ref” and PDAnchorPoint must be placed exactly at the center of the end of the tile named “Exit”. These points are used to stitch tiles together.
3. The “to be dressed” tile is placed between already finished tiles; this way when the objects for this tile are placed it is easier to avoid “dressing seams”. Starting with the big landmarks like rock structures the tile is made more unique looking. After that step, medium sized objects like houses, water towers, etc. are placed.
4. With these objects placed, the tile has enough definition to paint the mask for the ground material, when the mask is applied in the shader the tile is blending up to four ground textures that obscure the visible tiling of these textures, enriching the look of the tile more.
5. Small generic objects like cacti or small bushes are placed using the “advanced decogroup” (works independent from any speedtree code unlike the normal decogroup) object in vForge. This object allows us to basically paint the placement of the objects, with each click a new instance of the predefined object or even one object out of a group is placed at the clicked position, each time with a different rotation and if desired also different scale. Billboard objects like grass are placed in a similar manner, just by using the Billboard group.
6. As a final step for the object dressing we will hand place small objects like street signs, trash, fences etc.

3. Lighting of the tiles

3.1 Time of Day

As the game has only defined day times, these will be setup once per environment in a separate zone. It includes the sunlight (direction, color, and intensity), the sky box for this environment and additional lighting-relevant objects that define the ambient color, fog (color, near and far distance), Tone mapping etc.

Local dynamic lights were placed per zone with the light emitter object (e.g. Streetlight) and will only be active if the time of day requires it.

3.2 Ambient Lighting

We initially planned to use beast for lightmap generation but we may likely skip this as it due to the camera just the basic ambient color will be enough to provide a nice looking result.