Use Case Study: Journey - Effects and Terrain

Presented by Madis Janno

Use Case Study: Journey - ~~Effects and Terrain~~

Presented by Madis Janno

Use Case Study: Journey - Sand

Presented by Madis Janno

Journey

Released in 2012

Journey

Released in 2012

Playstation exclusive (until this year)

Journey

Released in 2012

Playstation exclusive (until this year)

Fastest selling game on PlayStation Store on release in North America and Europe

Journey

Released in 2012

Playstation exclusive (until this year)

Fastest selling game on PlayStation Store on release in North America and Europe

Looks really pretty


JOURNEY

This seminar based on:

Really short talk (22 min) by
John Edwards

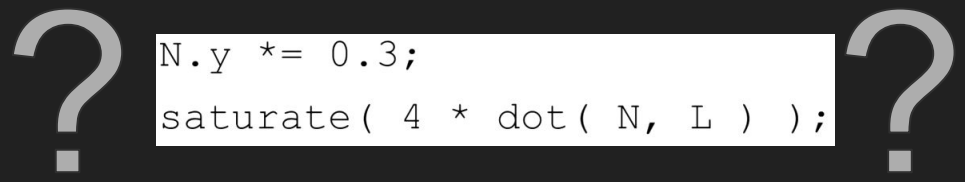
Overview of how their sand rendering system works

Mentions a lot of things

I will try to explain in more detail, make some observations, point out stuff that might have other uses

Why talk about Journey?

Stylized graphics means more “weirdness”



Why talk about Journey?

Stylized graphics means more “weirdness”

Thought an open-world game in a desert is a cool idea, so a decent excuse to consider how to make a desert environment look interesting

Why talk about Journey?

Stylized graphics means more “weirdness”

Thought an open-world game in a desert is a cool idea, so a decent excuse to consider how to make a desert environment look interesting

Example of how you can do some strange “hacks” to get good looking results

Why talk about Journey?

Stylized graphics means more “weirdness”

Thought an open-world game in a desert is a cool idea, so a decent excuse to consider how to make a desert environment look interesting

Example of how you can do some strange “hacks” to get good looking results

“To be a graphics programmer, you need neither intelligence nor competence.”

-John Edwards

Some things I will talk about in Journey

Making a desert without using reference photos

Mipmaps: “sharp” mipmaps

Specular shader that ignores light source location (is it technically still a specular?)

Anisotropic filtering: when it still isn’t good enough

Having two specular shaders at the same time

Weird diffuse shader

Terrain geometry

Making a desert without using reference photos

(they did use them at the end and added more detail because of it)

Started making the game by having an actual trip to a desert

Making a desert without using reference photos

(they did use them at the end and added more detail because of it)

Started making the game by having an actual trip to a desert

Made the shaders and rendering system based on the impression that left on them

Making a desert without using reference photos

(they did use them at the end and added more detail because of it)

Started making the game by having an actual trip to a desert

Made the shaders and rendering system based on the impression that left on them

Not something you could do with more realistic graphics

Making a desert without using reference photos

(they did use them at the end and added more detail because of it)

Started making the game by having an actual trip to a desert

Made the shaders and rendering system based on the impression that left on them

Not something you could do with more realistic graphics

Were more concerned with making it feel like a desert than actually looking like one

≠

Graininess and sharp mipmaps

Mipmaps

What are mipmaps?

Mipmaps

What are mipmaps? Why are mipmaps?

If the texture is bigger than the polygon being drawn, you end up with aliasing and flickering, because it keeps switching between different parts of the texture that correspond to the pixel

(https://www.shadertoy.com/view/3dVSDR)

Mipmaps

What are mipmaps? Why are mipmaps?

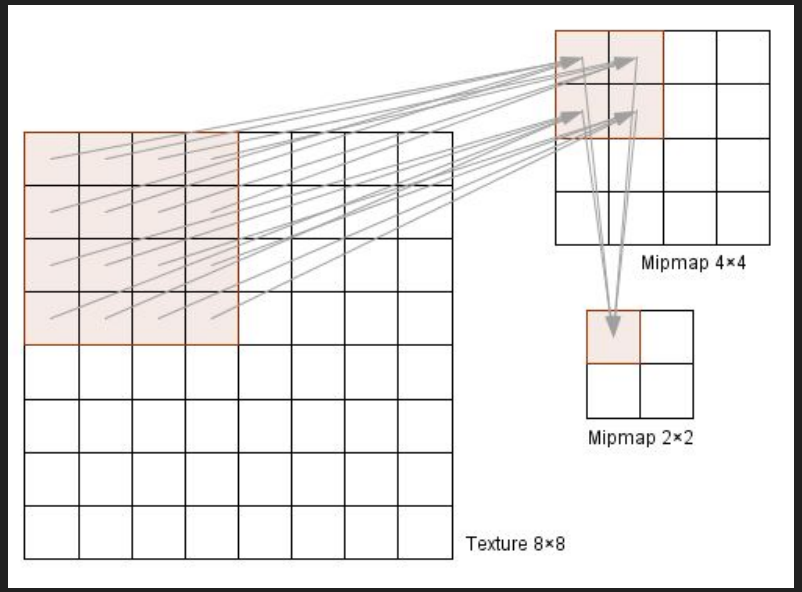
If the texture is bigger than the polygon being drawn, you end up with aliasing and flickering, because it keeps switching between different parts of the texture that correspond to the pixel

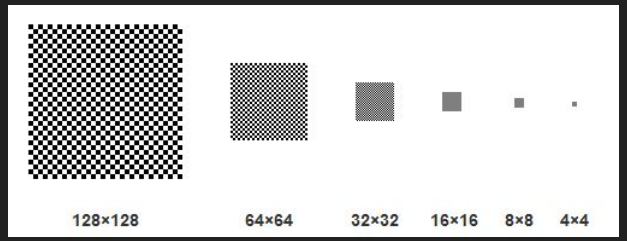
(https://www.shadertoy.com/view/3dVSDR)

Solution is multisampling. If several texels of the texture are in a pixel, just sample and average them all. This is very slow.

Mipmaps

This is very slow. Instead we precalculate this as a mipmap by averaging groups of 4 at a time.



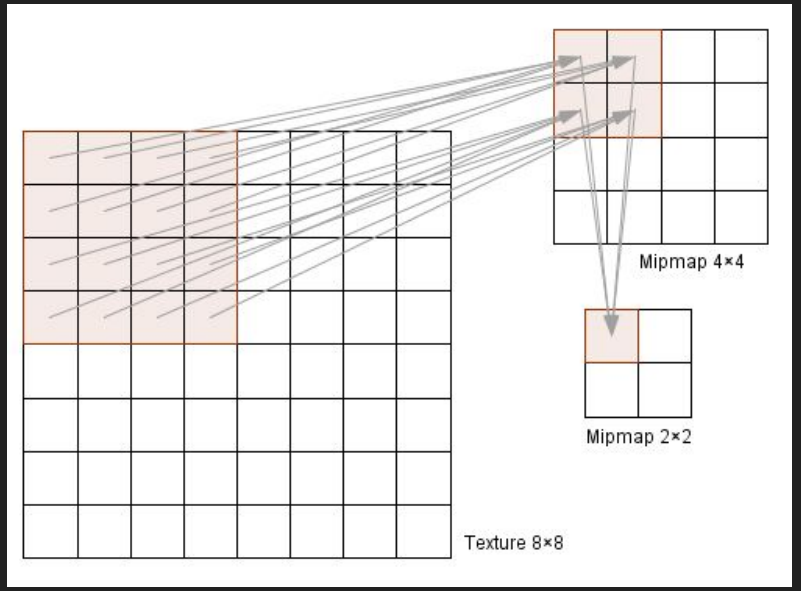


Images from **MTAT.03.015 Computer Graphics** materials

Mipmaps

This is very slow. Instead we precalculate this as a mipmap by averaging groups of 4 at a time.

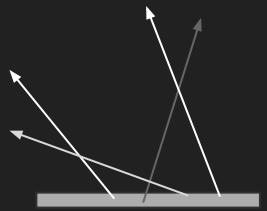
This is in essence a box blur. Which means each step is slightly more towards the mean (pure white + pure black -> gray)

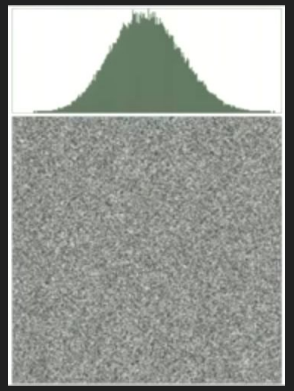


Graininess and sharp mipmaps

Journey uses noisy bump/normal maps on sand to add graininess

Essentially normals will be pointing in fairly random directions causing it look like it’s composed of individual particles



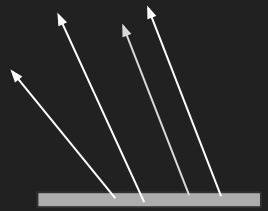


Graininess and sharp mipmaps

Journey uses noisy bump/normal maps on sand to add graininess

Essentially normals will be pointing in fairly random directions
causing it look like it’s composed of individual particles

Effect breaks down with terrain further from the camera because of mipmaps removing details and causing areas where each pixel has a similar normal



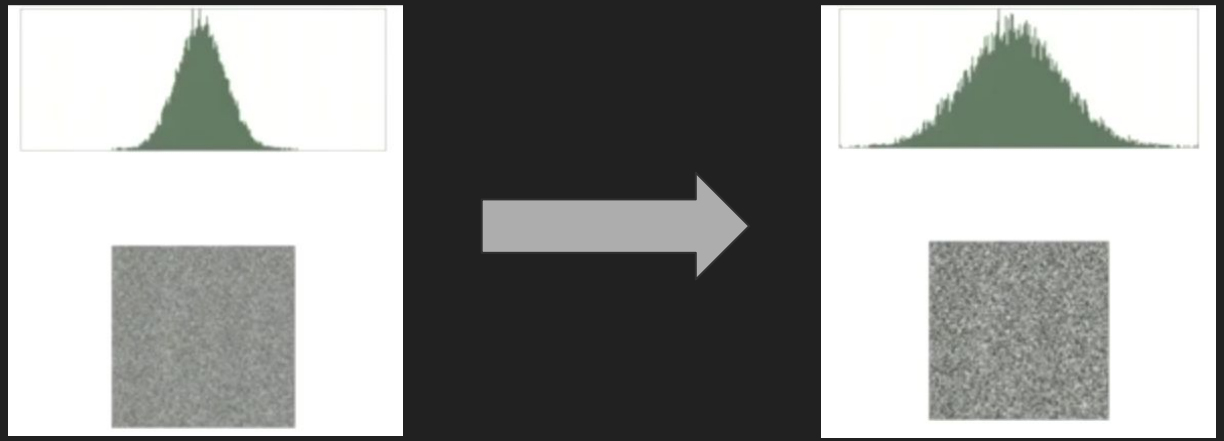
Aliasing, mipmaps and blur

Can’t just disable mipmaps, because of aliasing and flickering.

Aliasing, mipmaps and blur

Can’t just disable mipmaps, because of aliasing and flickering.

Solution is a less uniform mipmap. Just multiply the mipmap by 2 to exaggerate differences and break up blobs



Sharp

Regular

Glittery sand -> glitter specular

Sand has a glittery quality in sunlight, some sand particles reflect light at just the right angle and seem to shine

Glittery sand -> glitter specular

Sand has a glittery quality in sunlight, some sand particles reflect light at just the right angle and seem to shine

Pretty much just a specular. Normals already noisy, so use that

Glittery sand -> glitter specular

Sand has a glittery quality in sunlight, some sand particles reflect light at just the right angle and seem to shine

Pretty much just a specular. Normals already noisy, so use that

But disaster, sand now too sparkly. As you move the camera, lots of flashing pixels

Glittery sand -> glitter specular

Sand has a glittery quality in sunlight, some sand particles reflect light at just the right angle and seem to shine

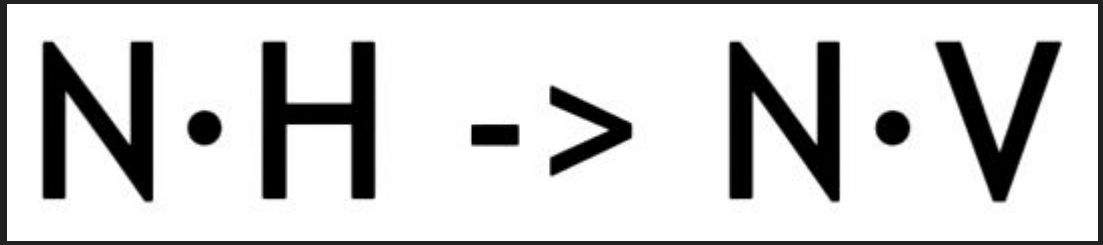
Pretty much just a specular. Normals already noisy, so use that

But disaster, sand now too sparkly. As you move the camera, lots of flashing pixels

Weird solution, use a slightly different formula

$$N \cdot H \rightarrow N \cdot V$$

Glittery sand -> glitter specular



(H is the half-way vector between light source and viewer, V is the viewer)

Was kinda confused by this, until I realized, yeah, they completely removed the light source portion from the calculation

Glittery sand -> glitter specular conclusions

Apparently the normals are noisy enough that this works

Apparently this causes less flashing

Really confused why they even tried that

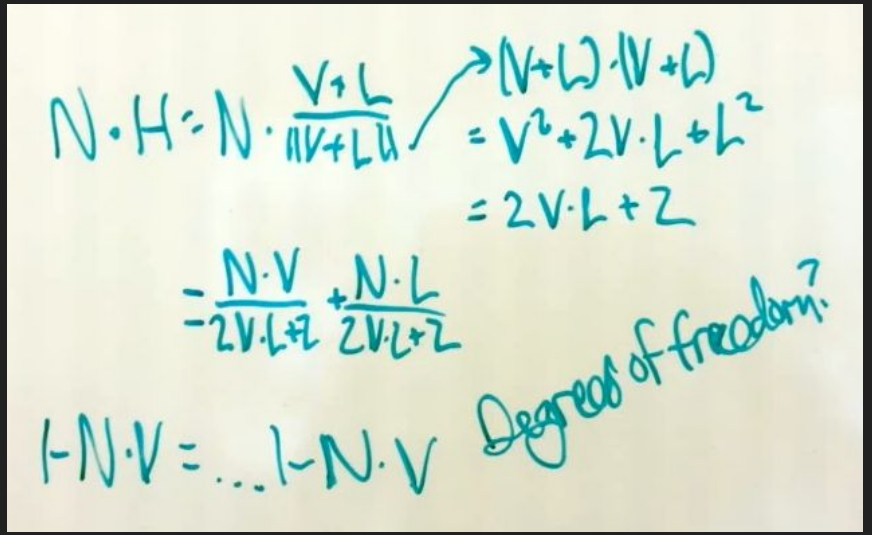
Glittery sand -> glitter specular conclusions

Apparently the normals are noisy enough that this works

Apparently this causes less flashing

Really confused why they even tried that

Guy giving talk does not know why this works better. Image on right might or might not explain it.



Glitter specular

Tried out a bit with shadertoy. Flat surface with normals pointing up but should be okay for testing. Green channel is N*H, red is N*V. Just random (3d) locations for light source and viewer. Green quite regularly dominated red, and also responded a lot more to changes in relative locations.

Too much glittery goop AKA failure of texture filtering

Problem now: mipmaps not giving good enough results

Too much glittery goop AKA failure of texture filtering

Problem now: mipmaps not giving good enough results

Worse problem: Anisotropic filtering also does not help

Too much glittery goop AKA failure of texture filtering

Essentially graininess breaks down in particular spots, causing large patches of pixels which glitter and look more like goop than individual particles

Too much glittery goop AKA failure of texture filtering

Essentially graininess breaks down in particular spots, causing large patches of pixels which glitter and look more like goop than individual particles

Simple solution: Just turn the glitter off where it breaks

Too little glitter

New problem: Have to turn glitter off on most of the screen.

Too little glitter

New solution: More glitter

Too little glitter

New solution: More glitter

New specular highlights taken from shaders generally used for water

Diffuse

Regular Lambert:

```
saturate( dot( N, L ) );
```

Diffuse

Regular Lambert:

```
saturate( dot( N, L ) );
```

Their Lambert:

```
N.y *= 0.3;
saturate( 4 * dot( N, L ) );
```

Diffuse

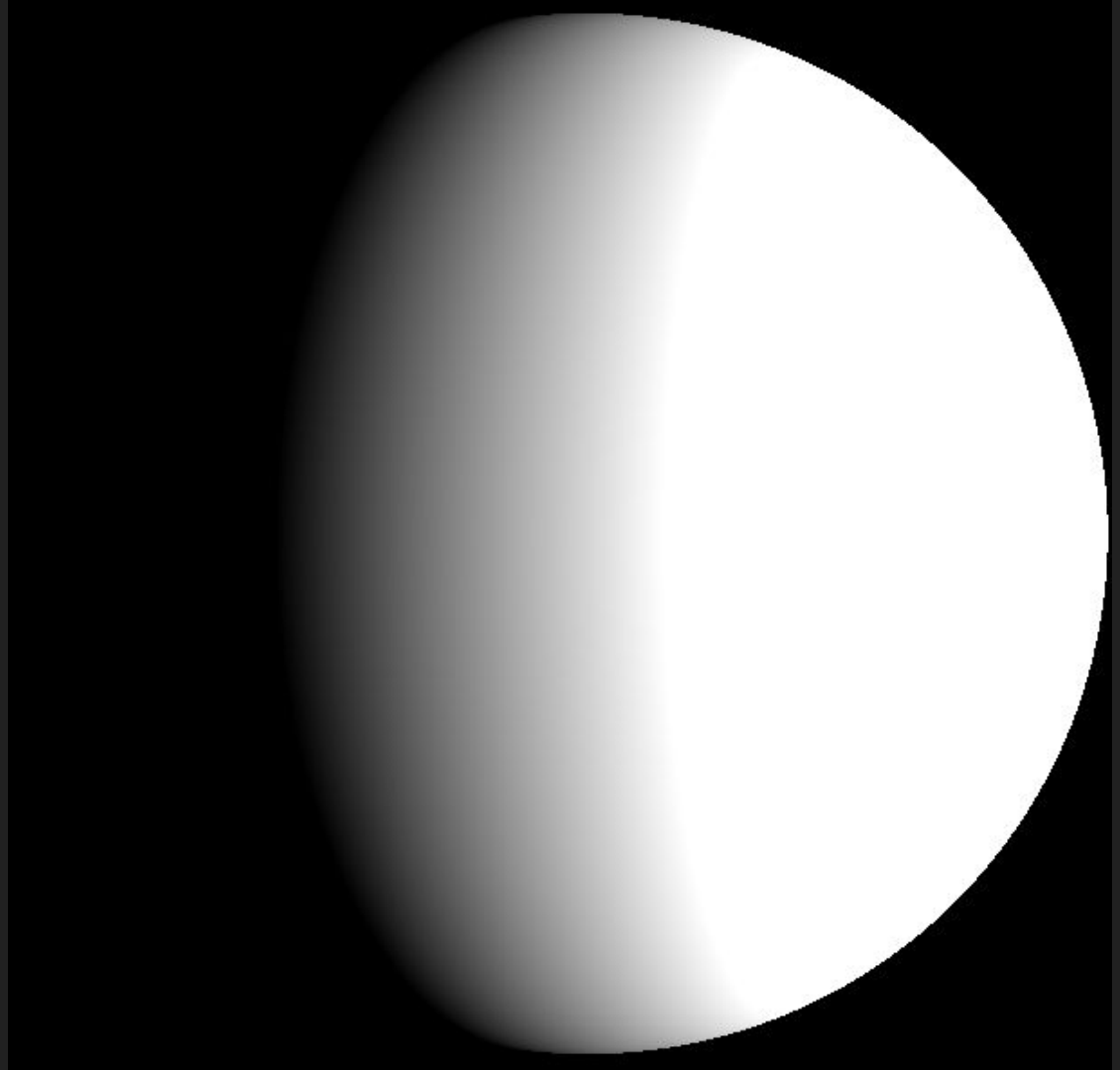
Regular Lambert:

```
saturate( dot( N, L ) );
```

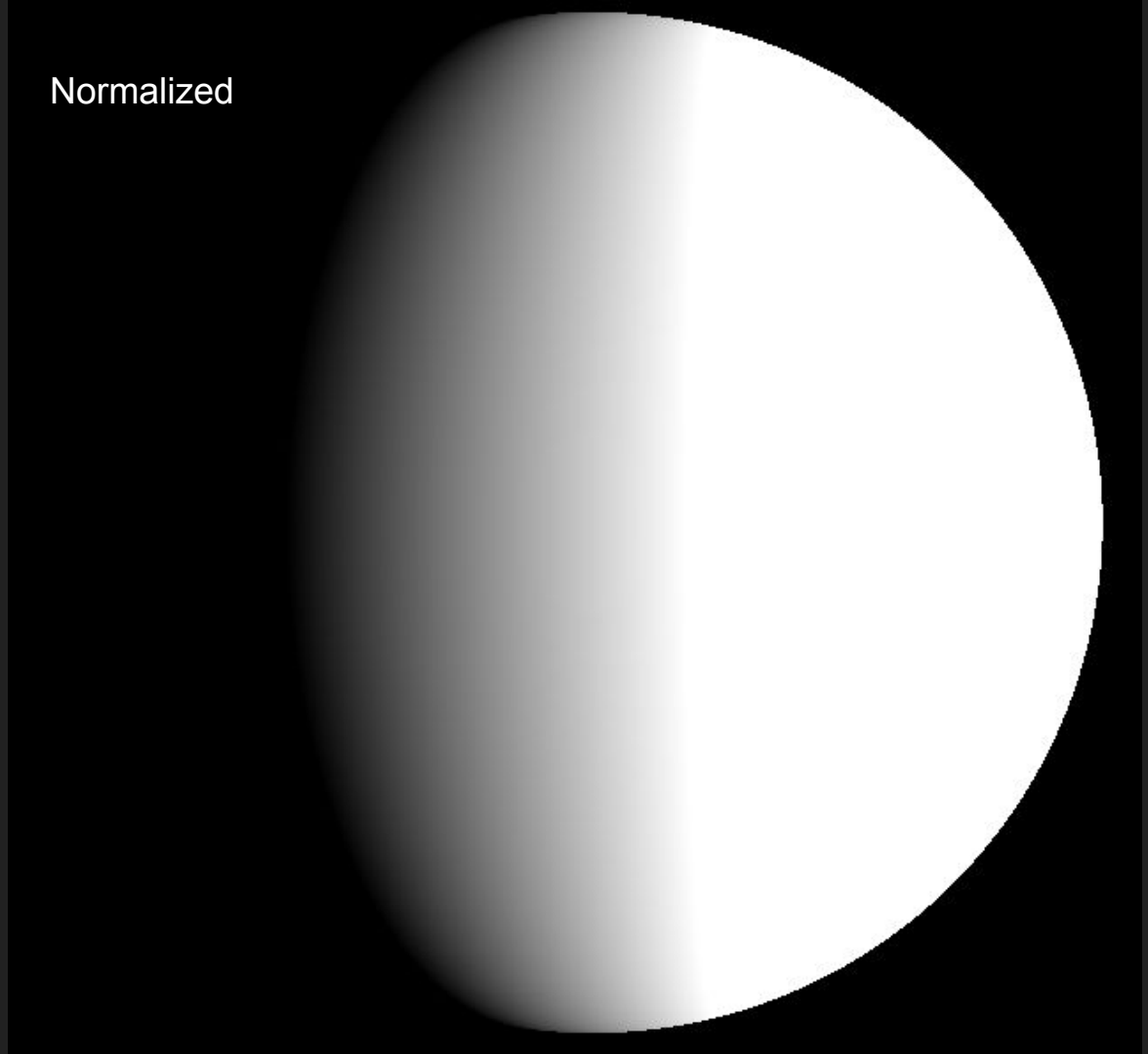
Their Lambert:

```
N.y *= 0.3;
saturate( 4 * dot( N, L ) );
```

Essentially: 4 times as bright, vertical component 3.33 times less important.
Apparently higher contrast.



Normalized



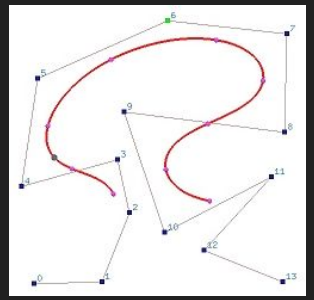
Terrain geometry

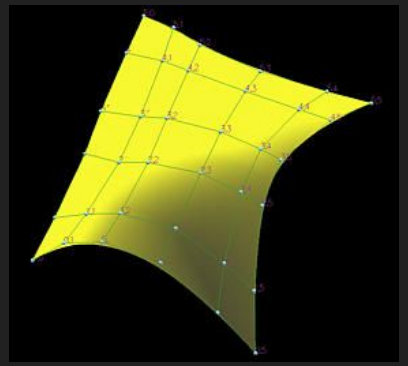
Actually a low resolution 512x256 texture

Terrain geometry

Actually a low resolution 512x256 texture

Turned to actual geometry through b-splines, really really smooth. 2nd derivative continuous.



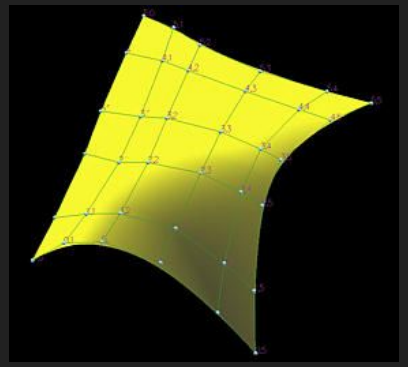


Terrain geometry

Actually a low resolution 512x256 texture

Turned to actual geometry through b-splines, really really smooth. 2nd derivative continuous.

Interestingly 2nd derivative can be used to estimate ambient occlusion.



Ambient occlusion

Not all places get exactly as much ambient light. For example corners and small spaces

Ambient occlusion

Not all spots get exactly as much ambient light. For example corners and small spaces

This depends on how much the surface is “occluded”, meaning how much stuff is in the way of light reaching the spots

Ambient occlusion

Not all spots get exactly as much ambient light. For example corners and small spaces

This depends on how much the surface is “occluded”, meaning how much stuff is in the way of light reaching the spots

Generally done with screen-space ambient occlusion

Ambient occlusion

Not all spots get exactly as much ambient light. For example corners and small spaces

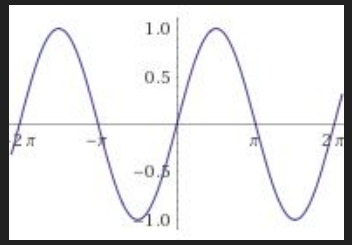
This depends on how much the surface is “occluded”, meaning how much stuff is in the way of light reaching the spots

Generally done with screen-space ambient occlusion

Can also be estimated if you know the second derivative of a surface.

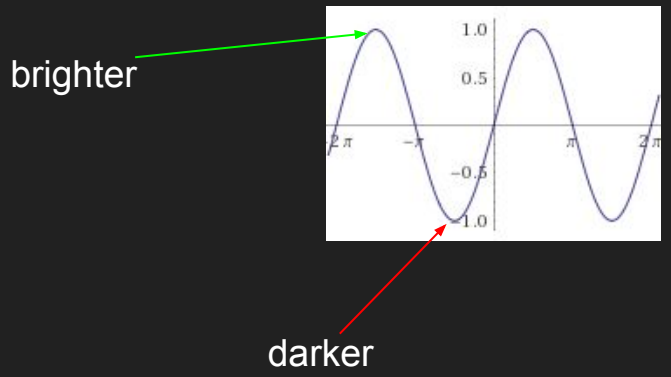
Ambient occlusion

Can also be estimated if you know the second derivative of a surface.



Ambient occlusion

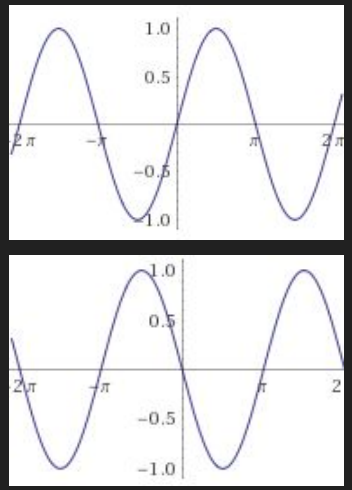
Can also be estimated if you know the second derivative of a surface.



Ambient occlusion

Can also be estimated if you know the second derivative of a surface.

“Simple” way is that you have an imaginary half-sphere where light is coming from, and based on the second derivative you can calculate how much of that sphere is covered up by the curve



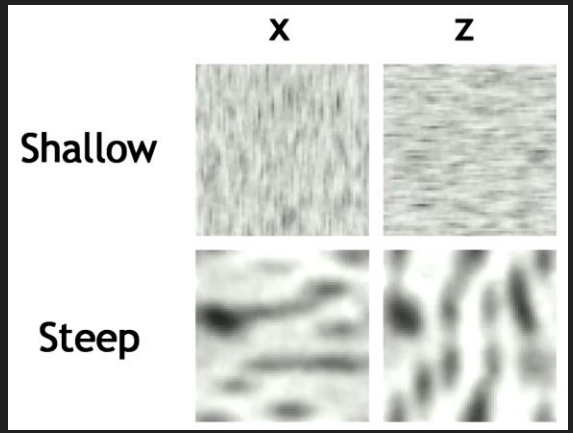
Back to geometry

Journey makes heavy use of tessellation and height-maps.

Back to geometry

Journey makes heavy use of tessellation and height-maps.

The detailed heights are based on tiling textures applied depending on the angle of the surface

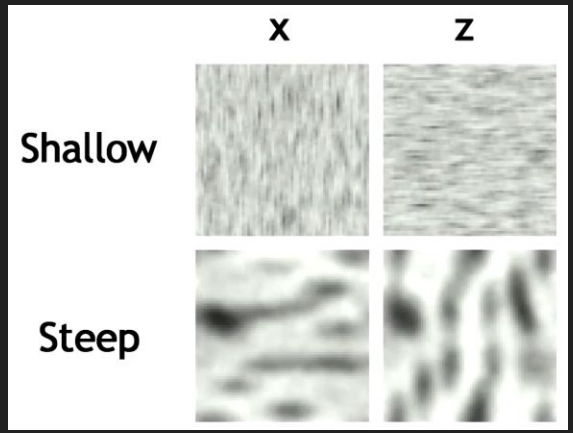


Back to geometry

Journey makes heavy use of tessellation and height-maps.

The detailed heights are based on tiling textures applied depending on the angle of the surface

Lots of triangles to make it look good



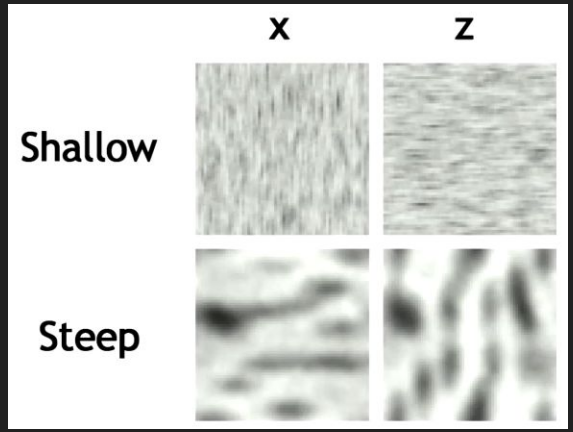
Back to geometry

Journey makes heavy use of tessellation and height-maps.

The detailed heights are based on tiling textures applied depending on the angle of the surface

Lots of triangles to make it look good

Task well suited to the PS3 SPU’s



Sand flow

There is actually a particle system in the game for the sand which the grain texture is based on.

Sand flow

There is actually a particle system in the game for the sand which the grain texture is based on.

~10000 32x32 particles with a sand texture. With a physics simulation run on the SPU’s

Sand flow

There is actually a particle system in the game for the sand which the grain texture is based on.

~10000 32x32 particles with a sand texture. With a physics simulation run on the SPU’s

Does not elaborate on this further.

That’s it for now. Any questions/thoughts?

Thanks for listening!