Some Graphic Secrets Behind

WRATH OF THE CAVE TROLL

BY
GENNARO ZAZO

Contents

- Physically Based Shading
  - Skin
  - Metal
  - Cloth
  - Eyes
  - Environment
  - Water
- Energy Conservation
- Image Based Lighting
  - Diffuse
  - Specular
  - Planar Reflections and Parallax Correction
- Layered Material
- LightMap & Shadow
- Wind Effects
- Water Droplets on Camera
- PostProcess Effects
- Extra Details
- Future Works

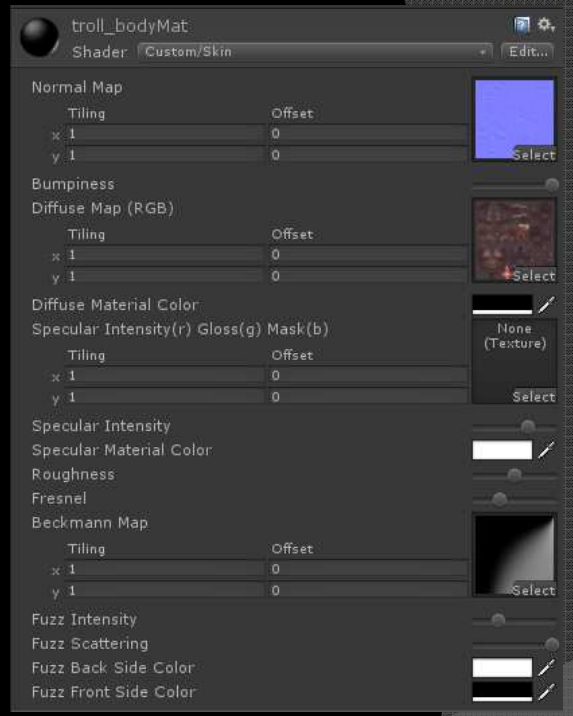
Skin

- Normal map
- Lambert model for diffuse
- BRDF with Beckman distribution function for specular
- Specular intensity map
- Gloss intensity map
- Roughness
- Fresnel term
- Fuzz to simulate back face lights scattering

Running in forward rendering

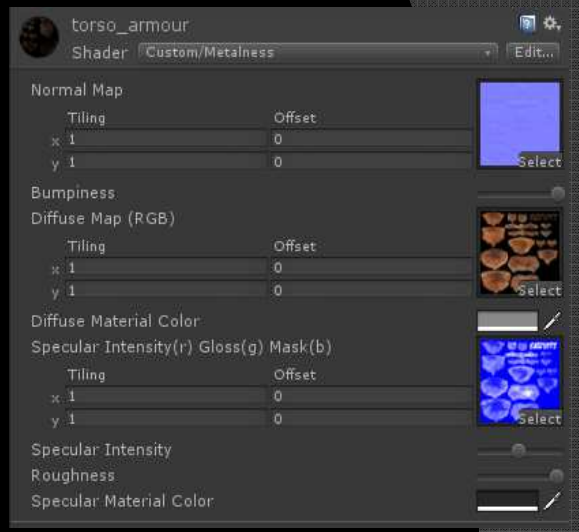
References:

- http://blog.selfshadow.com/publications/s2013-shading-course/



Metal

- Normal map
- Lambert model for diffuse
- BRDF with GGX micro facet distribution function for specular
- Specular intensity map
- Gloss intensity map
- Roughness
- For Fresnel index reflectance we use 0.98112



Running in forward rendering

References:

- http://blog.selfshadow.com/publications/s2013-shading-course/

Cloth

- Normal map
- Lambert model for diffuse
- BRDF with Ashikhmin distribution function for specular
- Specular intensity map
- Gloss intensity map
- Roughness
- Fresnel term to simulate different cloth

Running in forward rendering

References:

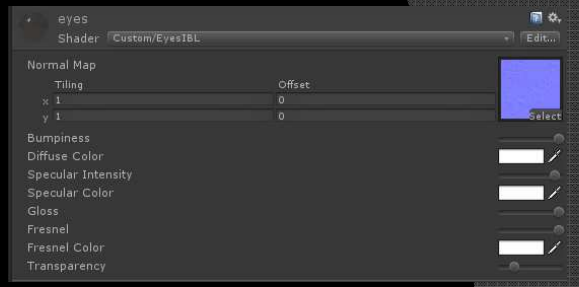
- http://blog.selfshadow.com/publications/s2013-shading-course/
- http://www.cs.utah.edu/~michael/brdfs/facets.pdf
- http://www.cs.utah.edu/~premoze/dbrdf/dBRDF.pdf

Eyes

To simulate the eyes depth, we use two models

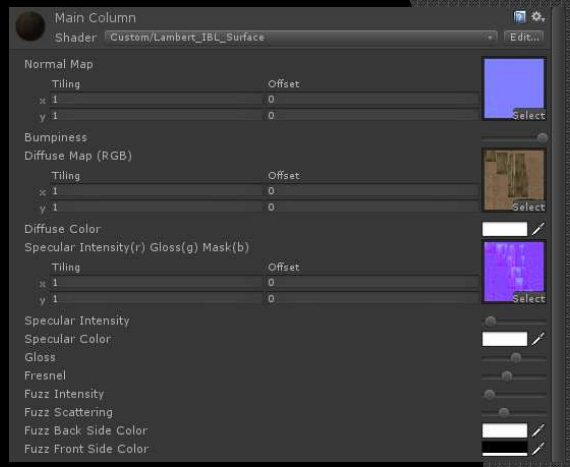
- Iris with skin shader
- Glass with eye ibl shader,
  - Blinn Phong model for specular
  - Fresnel

Running in forward rendering



Enviroment Shading

- Normal map
- Lambert model for diffuse
- Blinn Phong for specular
- Specular intensity map
- Gloss intensity map
- Fresnel term
- Fuzz to simulate back face lights scattering



For Grass and Plants we use a different version with alpha cutout

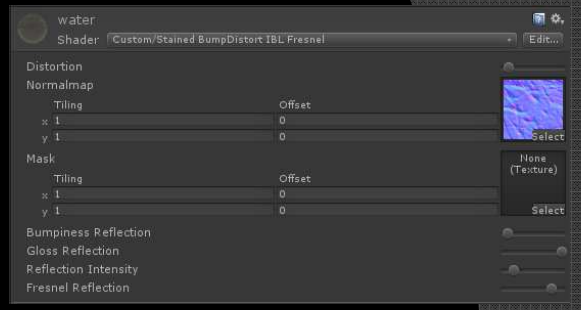
Running in deferred rendering

Water Shading

To simulate water and puddles we use a simple distortion shader with animated normal map and mask

- Distortion amount
- Bumpiness for IBL reflection
- Gloss Reflection
- Fresnel Reflection
- Intensity Reflection

Running in forward rendering



Energy Conservation

All shaders are affected by energy conservation

- For Diffuse Term
- For Specular Term
- For Diffuse+Specular

References:

- http://www.rorydriscoll.com/2009/01/25/energy-conservation-in-games/

Image Based Lighting Diffuse & Specular

All shaders are affected by IBL

- A well known technique
- We build an offline specular cube map of the scene
- Instead of applying convolution to captured cube map for the diffuse component, we prefer to use outdoor cube map to preserve some extra contributes of the outdoor lighting

References:

- http://ict.usc.edu/pubs/Image-Based%20Lighting.pdf/

Planar Reflections with Parallax Correction

To prevent some reflection artifacts (Reflected objects are not at the right position) we use parallax correction

References:

- http://seblagarde.wordpress.com/2012/09/29/image-based-lighting-approaches-and-parallax-corrected-cubemap/

Layered Material

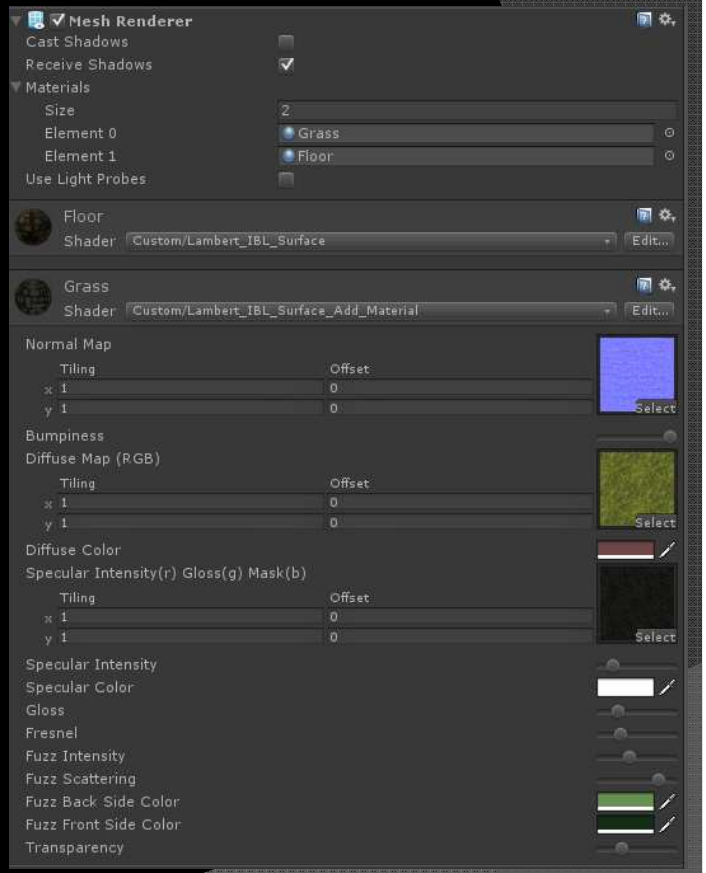
In some situations we want more materials on a single object

- Base Material
- Add Material

To achieve the effect we force unity to render additional material with some tricks in the additional shader material.

The additional material is in alpha blending driven by a mask and a transparency amount, to achieve different results with same mask.

We can add infinites additional materials.



LightMap & Shadow

We use (Unity built-in) SingleLightMap to achieve

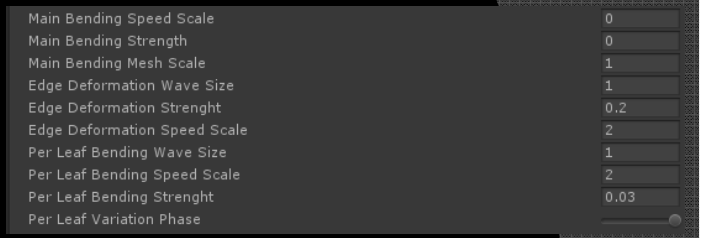
- GI effects
- Ambient Occlusion

All shadows are in real time

- Moon Light
- Cauldron Light

Wind Effects

A more realistic wind effect, with more control like in UDK based on vertex colour. We can easily simulate grass, plants and tree with the same vertex shader.



- Main Bending, xz deformation
- Leaf Edge Deformation, red channel, local wave deformation
- Per-Leaf Variation, green channel, phase variation
- Per-Leaf Bending, blue channel, y deformation

References:

- http://minifloppy.it/tutorials/udk-wind-vertex-shader/

Water Droplets on Camera

We want water droplets on camera. There are a lot of techniques some based on fake simulation (ugly) and other based on fluids (expensive).

We choose an alternative way. To achieve water droplets on camera we simulate two systems

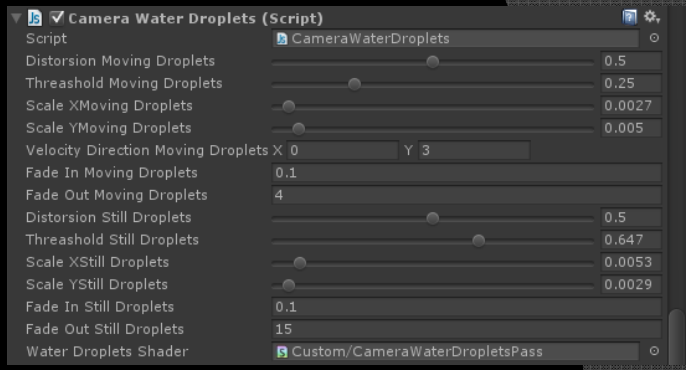
- Still droplets
- Moving Droplets

Both systems are simulated by a 3d Perlin Noise as post process effect.

Our implementation is a simplex noise without arrays or textures, very cheap and fast.

References:

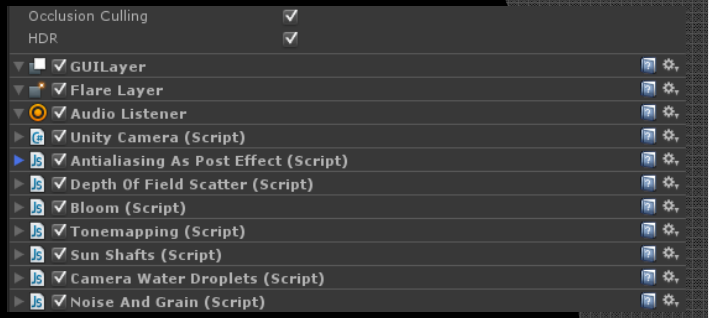
- https://github.com/ashima/webgl-noise



PostProcess Effects

To achieve the final image we use some post process effects Unity built-in (great results and good implementations)

- Almost all running in deferred rendering then we choose antialiasing as post process, FXAA3Console(cheap and good)
- DOF for Dx11 with bokeh
- Bloom
- ToneMapping
- Sun Shafts
- Noise and Grain (because we love it ;))



Extra Details

- The demo is created in Unity 4 Pro
- It should be running at least at 40 fps on GTX 460 1GB
- It is DX11 only
- It supports only 16/9 and 16/10 resolutions
- All the textures are in 4K

Future Works

Time is running out but we'd like to add some extra works

- Screen Space Sub Surface Scattering, Troll is fantastic even without but ….
- Screen Space Local Reflection(very expensive) with planar specular reflection works very well. https://www.youtube.com/watch?v=pKpwi4GgaOg
- Some performance optimizations
- Fix some random spikes, very strange we don't where they come from.

Future Works

The time is running out but we'd like to add some extra work

- Screen Space Sub Surface Scattering, the Troll looks great even without but ….
- Screen Space Local Reflection(very expensive) with planar specular reflection works very well. https://www.youtube.com/watch?v=pKpwi4GgaOg
- Some performance optimizations
- Fix some very strange random spikes, we don't know where they come from.