Uncharted 2: Character Lighting and Shading

John Hable
Naughty Dog
Focus: Characters
You play as Drake, the loveable rogue. Check out this link for more character development: http://www.penny-arcade.com/comic/2009/10/19/.
Elena
Lazaravic
Shafer
Todo List:

- Skin
- Hair
- Cloth
- 5 Secrets
The best implementation of skin in realtime that I’ve seen is the NVIDIA Human Head demo.

Skin

- Efficient Rendering of Human Skin, Eugene d'Eon, David Luebke, and Eric Enderton, Eurographics 2007
- NVIDIA Human Head Demo
- Compare with real shots.
When you look at the render on the left vs. an actual picture of Doug Jones, there is a certain fleshiness that is hard to explain. That’s what I like about the NVIDIA technique. On some level, it just “feels” like skin.

Obviously, movies can go much farther than we can in realtime. We’ll catch up eventually...I hope.
Let’s Test!

- Lazaravic
- RenderMonkey

This next section will show lots of pictures using Lazarovic from U2. Note that these tests are in Rendermonkey, not the game engine.
Compare 5-ish Approaches

1. Standard Diffuse
2. Nvidia Approach
3a. 12-Tap Separate
    • Do 12-Tap blur as separate shader.
3b. 12-Tap Combined
    • Do 12-Tap blur in final render pass.
4. Bent Normals
5. Blended Normals
1) Standard Diffuse

Lazarovic with standard dot(N,L) lighting.
Lighting-Only
The NVIDIA technique divides light into diffuse light that gets absorbed and immediately retransmitted, versus light that bounces around inside the skin for a while before exiting. Not that the SSS light is more red-ish.
2) NVIDIA Human Head Demo

- Efficient Rendering of Human Skin, Eugene d'Eon, David Luebke, and Eric Enderton, Eurographics 2007
2) NVIDIA Approach

- Combine Blurs

```plaintext
diffColor = direct*float3(.233,.455,.649);
diffColor += lm1*float3(.100,.336,.344);
diffColor += lm2*float3(.118,.198,0);
diffColor += lm3*float3(.113,.007,.007);
diffColor += lm4*float3(.358,.004,0);
diffColor += lm5*float3(.078,0,0);
```

Their approach uses essentially 6 layers of blur.
The first layer is essentially no blur. This simulates light that gets absorbed and immediately retransmitted.

2) NVIDIA Approach

- Combine Blurs

```c
diffColor = direct*float3(.233,.455,.649);
diffColor += Im1*float3(.100,.336,.344);
diffColor += Im2*float3(.118,.198,.0);
diffColor += Im3*float3(.113,.007,.007);
diffColor += Im4*float3(.358,.004,.0);
diffColor += Im5*float3(.078,0,0);
```
2) NVIDIA Approach

- Combine Blurs

```
diffColor = direct*float3(.233, .455, .649);
diffColor += lm1*float3(.100, .336, .344);
diffColor += lm2*float3(.118, .198, 0);
diffColor += lm3*float3(.113, .007, .007);
diffColor += lm4*float3(.358, .004, 0);
diffColor += lm5*float3(.078, 0, 0);
```

The use weights for all 5 lightmaps to simulate the light that bounces inside the skin.
2) NVIDIA Approach

- Combine Blurs

```csharp
diffColor = direct*float3(.233,.455,.649);
diffColor += Im1*float3(.100,.336,.344);
diffColor += Im2*float3(.118,.198,.0);
diffColor += Im3*float3(.113,.007,.007);
diffColor += Im4*float3(.358,.004,.0);
diffColor += Im5*float3(.078,0,0);
```
2) NVIDIA Approach
Lighting-Only

- Lambertian vs. NVIDIA

The left side is a standard dot(N,L) and the right is with the NVIDIA skin shading.
Why it works?

- Red bleeds
- Blue/Green not so much
- Towards Light: Cyan-ish
- Away From Light: Red-ish

Key point: The normals that point towards the light tend to look more cyan-ish and the normals that point away tend to be more red-ish.
Lighting-Only

- Lambertian vs. NVIDIA

Comparison of pure dot(N,L) diffuse to the NVIDIA SSS technique.
Their approach looks great, but is very expensive in both memory.
3) 12-Tap Approximation

NVIDIA
Instead of using 5 gaussian blurs, we’ll try to approximate that with a single 12-tap blur. It’s not as good, but much, much cheaper.
3) 12-Tap Approximation

- From ShaderX7 (Hable, Borshukov, and Hejl)
- Do a single 12-tap blur

```cpp
float3 blurJitteredWeights[13] =
{
    { 0.220441, 0.437000, 0.635000 },
    { 0.076356, 0.064487, 0.039097 },
    { 0.116515, 0.103222, 0.064912 },
    { 0.064844, 0.086388, 0.062272 },
    { 0.131798, 0.151695, 0.103676 },
    { 0.025690, 0.042728, 0.033003 },
    { 0.048593, 0.064740, 0.046131 },
    { 0.048092, 0.003042, 0.000400 },
    { 0.048845, 0.005406, 0.001222 },
    { 0.051322, 0.006034, 0.001420 },
    { 0.061428, 0.009152, 0.002511 },
    { 0.030936, 0.002868, 0.000652 },
    { 0.073580, 0.023239, 0.009703 };
};

float2 blurJitteredSamples[13] =
{
    { 0.000000, 0.000000 },
    { 1.633992, 0.036795 },
    { 0.177801, 1.717593 },
    { -0.194906, 0.091094 },
    { -0.239277, -0.220217 },
    { -0.003530, -0.118219 },
    { 1.320107, -0.181542 },
    { 5.970690, 0.253378 },
    { -1.089250, 4.958349 },
    { -4.015465, 4.156699 },
    { -4.063099, -4.110150 },
    { -0.638605, -6.297663 },
    { 2.542348, -3.245901 };
};
```

Check the ShaderX7 chapter for more detail.
3) 12-Tap Approximation

float3 blurjitteredWeights[13] =
{
    { 0.220441, 0.437000, 0.635000 },
    { 0.076356, 0.064487, 0.039097 },
    { 0.116515, 0.103222, 0.064912 },
    { 0.064844, 0.086388, 0.062272 },
    { 0.131798, 0.151695, 0.103676 },
    { 0.025690, 0.042728, 0.033003 },
    { 0.048593, 0.064740, 0.046131 },
    { 0.048092, 0.003042, 0.000400 },
    { 0.048845, 0.005406, 0.001222 },
    { 0.051322, 0.006034, 0.001420 },
    { 0.061428, 0.009152, 0.002511 },
    { 0.030936, 0.002868, 0.000652 },
    { 0.073580, 0.023239, 0.009703 },
}
3) 12-Tap Approximation

float3 blurJitteredWeights[13] =
{
    { 0.220441, 0.437000, 0.635000 },
    { 0.076356, 0.064487, 0.039097 },
    { 0.116515, 0.103222, 0.064912 },
    { 0.064844, 0.086388, 0.062272 },
    { 0.131798, 0.151695, 0.103676 },
    { 0.025690, 0.042728, 0.033003 },
    { 0.048593, 0.064740, 0.046131 },
    { 0.048092, 0.003042, 0.000400 },
    { 0.048845, 0.005406, 0.001222 },
    { 0.051322, 0.006034, 0.001420 },
    { 0.061428, 0.009152, 0.002511 },
    { 0.030936, 0.002868, 0.000652 },
    { 0.073580, 0.023239, 0.009703 },
}
3) 12-Tap Approximation

float3 blurJitteredWeights[13] =
{
    { 0.220441, 0.437000, 0.635000 },
    { 0.076356, 0.064487, 0.039097 },
    { 0.116515, 0.103222, 0.064912 },
    { 0.064844, 0.086388, 0.062272 },
    { 0.131798, 0.151695, 0.103676 },
    { 0.025690, 0.042728, 0.033003 },
    { 0.048593, 0.064740, 0.046131 },
    { 0.048092, 0.003042, 0.000400 },
    { 0.048845, 0.005406, 0.001222 },
    { 0.051322, 0.006034, 0.001420 },
    { 0.061428, 0.009152, 0.002511 },
    { 0.030936, 0.002868, 0.000652 },
    { 0.073580, 0.023239, 0.009703 },
};
3) 12-Tap Approach

- A: Separate Blur
3) 12-Tap Approach

- B: Combined Blur
3) 12-Tap Approximation

- *Separate* Blur
  - Render to Lightmap
  - Blur Lightmap with 12 taps
  - Render Final Scene

- *Combined* Blur
  - Render to Lightmap
  - Render Final Scene, with 12 taps from lightmap

- Both are fine
12-Tap Separate
12-Tap Combined
4) Bent Normals (?)

- Pretend R/G/B come from different Normals
- R closer to Geometry, GB closer to Normal Map
- Diffuse Calculation 3 Times
4) Bent Normals
Notice how using different normals for R/G/B seems to cause some blue spottiness.
Why so blue?

- Dot Products
- At extreme angles, you can have cases where $\text{diffuseR} = 0$ and $\text{diffuseB} = 1$
  - Or vice-versa

That’s because you have cases where the blue diffuse is near 1 and the red diffuse is near 0.
Another approach is to do a diffuse calculation for the Geometry and Normal Mapped normals, and lerp between them taking more red from the Geometry normal and more Green/Blue from the normal mapped normal.
5) Blended Normals
Still Kinda Blue
Recap

1. Standard Diffuse
2. Nvidia Approach
3. 12-Tap Combined
   • Do 12-Tap blur as separate shader.
3a. 12-Tap Merged
   • Do 12-Tap blur in final render pass.
4. Bent Normals
5. Blended Normals
1) Straight Diffuse
2) NVIDIA Approach
3a) 12-Tap Separate
3b) 12-Tap Combined
4) Bent Normals
5) Blended Normals
1) Standard Diffuse
2) NVIDIA Approach
3a) 12-Tap Separate
3b) 12-Tap Combined
4) Bent Normals
5) Blended Normals
What did we do?

• Cutscenes:
  – 3b) 12-Tap Combined

• In Game
  – 5) Blended Normals
  – 12-Tap Combined cost .4ms per head
  – Difference didn’t balance the cost
Cutscenes
Cutscenes

Sometimes, in cutscenes, the shots get pretty close, so we need the quality.
Normal Gameplay

You spend most of normal gameplay staring at the back of Drake’s neck, so a separate pass for SSS was not worth the cost.
Limitations

- Resolution
- Seams
Conclusion

- SSS really helps
- Do what makes sense for you
- Do better than Blinn-Phong
Hair
Hair
Hair

- Stolen from Thorsten Scheuermann
  - [http://developer.amd.com/media/gpu_assets/Scheuermann_HairSketchSlides.pdf](http://developer.amd.com/media/gpu_assets/Scheuermann_HairSketchSlides.pdf)
- Kajiya-Kay
Specular Off
Combined
Notes

- Slight Wraparound Diffuse
- Kajiya-Kay Specular
- No self-shadowing
  - Looks into largest cascade w/extra bias
- Diffuse Map as Specular Mask
  - Partially Desaturated
- Deferred lights use Blinn-Phong
Conclusions

- Anisotropic
- Do better than Blinn-Phong
Cloth
Cloth
Diffuse-Only, Right?
Specular

• Fresnel?
Cloth

- “Examples of completely diffuse materials include Cloth and Cardboard”
  - Nope!
  - Sidenote: Cardboard is actually really shiny
- Cloth has some Fresnel too
U2 Cloth

- Rim Lobe, Inner Lobe, Remaining Diffuse
U2 Cloth

• Final
OurCloth()
{
    VdotN = saturate( dot( V, N ) );
    Rim = RimScale * pow( VdotN, RimExp );
    Inner = InnerScale * pow( 1-VdotN, InnerExp );
    Lambert = LambertScale;

    ClothMultiplier = Rim + Inner + Lambert;
    FinalDiffuseLight *= ClothMultiplier;
}
Cloth

- Not based on light direction
- Looks better than nothing
- Hanno: “Doesn’t look that great, but it’s hard to screw up.”
Conclusions

- Do something, anything
5 Secrets
The left shot is one of the first released screenshots before a lot of the tech got in. The one on the right is what we shipped. The one on the left has lots of hacks that we eventually took out, such as that orange glow around Drake’s skin.
Hacks Don’t Work!

- Hacks look great in still images.
- Don’t work when you move the light/camera.
- Photoshop tricks only work in stills
Secret #2

- Avoid Wraparound Lighting Models
- A few exceptions (I.e. Hair)
- $\text{diff} = 0.5 + 0.5 \times \text{dot}(\text{N}, \text{L})$
  - As opposed to: $\text{diff} = \text{saturate}(\text{dot}(\text{N}, \text{L}))$
- VFX/Photography: Ignore this point
Secret #2

- Lambert.
- Looks too crunchy.
Secret #2

- Wraparound
- Do you want your key light to look like this?
- Looks ok for ambient.
Secret #2

- You want this.
Secret #2

- With SSS
- Looks fine.
- Disclaimer for Film / Photography
Harsh Falloff

• Harsh Falloff is your friend
  – If you use it right
• Don’t wrap light to make skin softer
• Instead change your shading model
• Doesn’t apply to VFX and Photography
Notice that the NVIDIA demo has a harsh falloff. It looks great if you do everything else right, which is why that demo is the gold standard for skin in realtime.
Secret #3

- Avoid Blurry Maps on Faces
- Have detail in Diffuse Map
- Have crazy detail in Normal Map
- It should look terrible in Maya
Since he looks fleshy, you would think that he has blurry maps.
NVIDIA Human Head Maps

- Diffuse Maps
NVIDIA Human Head Maps

- Normal Maps (world space?)
NVIDIA Human Head Maps

- Don’t paint soft maps
- Paint very crunchy detailed maps
- Let lighting model soften it

In the demo, they have extremely detailed maps and they use the lighting model to soften it.
IMO what makes skin look right is how light bleeds around the normals. If you paint soft maps with no detail in the normals, it just looks flat.
Maps

• Crank the detail!

Notice that there is more detail in the shot on the right. For U2, we really cranked the detail in the maps and the strength of the normals and then let the lighting model soften it.
Secret #4

- Don’t bake too much lighting into diffuse maps
- Especially AO
- Becomes Unlightable
Drake’s Shirt

- Looks fine in Sunlight
- Very boring in Shadows
Drake’s Shirt

- Create Light Rig
- Ambient Light
AO into Diffuse

- Apply to Diffuse
- Better Ambient
- The Catch?
The Unlightables

- \((151/255)^{2.2}=0.315\)
- \((35/255)^{2.2}=0.0126\)
- 25x!!!
- Full histogram in ambient

Don’t bake too much AO into your diffuse maps.
Solution

- Leave AO map separate
- Can be lower-res
- direct=diffuseMap*diffuseLight
- ambient=diffuseMap*aoMap*ambientLight
- diffuseColor=direct+ambient
You get HDR lighting from having high dynamic range in your LIGHTING. I see a lot of games that have all the tech for HDR lighting, but it still looks flat. The reason 99% of the time is that they have too much black in their textures. For an example of a game doing a really good job of managing their textures, check out Mirror’s Edge.
Secret #5

- Make sure your AO and Diffuse match
- We screwed this up
- Don’t play telephone
Chloe-Intro

- White Direct
- Yellow-ish Ambient
Chloe

- Direct Too High (White)
- Ambient Too Low (Yellow)
Drake

- Direct Too Low (White)
- Ambient Too High (Yellow)
Go through the cutscenes again and look for Drake’s tongue. You’ll see what I mean.
Conclusions

• Custom Shading Models
• If you tried these things in the past
  – Take a second look if your lighting has improved
• Linear-Space Lighting

I’m a big fan of custom lighting models (i.e. beyond Blinn-Phong). Btw, doing proper Linear-Space Lighting is more important than everything in this presentation combined.
References

- Practical Real-Time Hair Rendering and Shading, Thorsten Scheuermann, Siggraph Sketch, 2004
- Efficient Rendering of Human Skin, Eugene d’Eon, David Luebke, and Eric Enderton, Eurographics 2007
- Fast Skin Shading, John Hable, George Borshukov, and Jim Hejl, ShaderX7, 2008
Done!