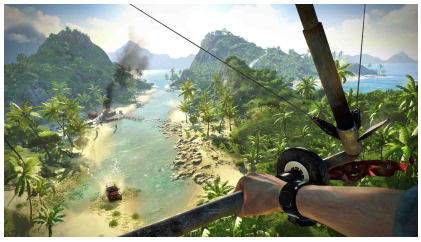
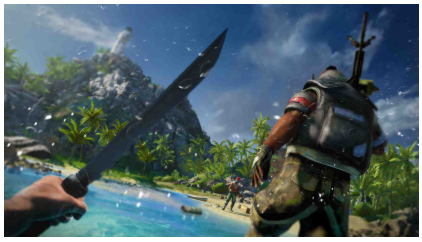
Calibrating Lighting and Materials in *Far Cry 3*

by Stephen McAuley, Ubisoft Montréal

stephen.mcauley@ubisoft.com





Introduction

Far Cry 3 is a first-person shooter due to be released in 2012 on Xbox 360, PlayStation 3 and PC. With indoor and outdoor environments, and a real-time day-night cycle, the game engine must support a huge variety of lighting scenarios. Materials must react correctly under all possible lighting conditions and all possible viewing angles. This made us turn to a physically-based model. By accurately simulating the physical properties of lights and materials, we are able to ensure that every object would react as expected under any lighting condition. This leads to an improved workflow for artists who are no longer trying to overcome lighting problems, and thus also gives a higher quality result.

For materials, we observed that diffuse albedo is a physical property that we can use as in our material representation. We developed a colour correction tool that calculates diffuse albedo from photographs, using a Macbeth ColorChecker® grid as reference. Artists could then build textures using this colour-corrected photographic reference. Not only did this help with balancing material response under all lighting conditions, but it enabled us to have vivid, but not garish, colours required of a tropical island. Other physical properties, such as specular reflectance, were also accurately simulated to ensure the faithful reproduction of both diffuse and specular lighting effects.

To calibrate our lighting, we generated second-order spherical harmonics from the sky dome to use for the ambient light, whereas previously, a separate, single colour ambient light was used. Linking ambient lighting to the sky dome not only ensured consistency but also added a variety of colours to the lighting. Problems came when art direction requested a look which would result from the application of a polarising filter, which blocks the direct sky light, but not the diffuse sky lighting. This was solved with a tailored post-process for the sky.

Finally, we implemented a physically-based shading model in order to accurately use the data we provided. Energy conserving specular was of critical importance to simply our material parameters and to ensure specular highlights were of the correct brightness. As the vast majority of materials in *Far Cry 3* are dielectric, we chose the Torrance-Sparrow microfacet BRDF with Schlick's Fresnel approximation, the Blinn-Phong NDF and the Schlick-Smith visibility term. With performance at a premium, we found a fast approximation to the geometric term that could be combined for free with the normalisation factor of the Blinn-Phong NDF. To correctly filter specular highlights, we selected Toksvig maps over LEAN mapping for memory and performance reasons. These were generated from our normal maps and placed them in a spare channel of our DXT5 normal map textures. Artists could optionally use a single value average Toksvig factor to solve any compression issues.

Figure 1: *Far Cry 3* concept art contains vivid, saturated colours.

Calibrating Albedo

Motivation

Far Cry 3 is set on a tropical island and, as such, art direction required our materials to have bright, vivid and often saturated colours. The real-time day-night cycle and the use of indoor and outdoor environments means that materials must hold up under all conditions, and one cannot afford to tweak materials for particular scenes.

Albedo is the ratio of total reflected energy to incident energy as a function of incident angle. Diffuse albedo is the albedo of the diffuse component of the lighting, which is a constant in a Lambert difuse model. This is best understood by studying a common BRDF used in games, incorporating a Lambert diffuse term and a microfacet BRDF for the specular contribution. The diffuse albedo is represented by $\mathbf{c}_{\text{diff}}$.

$$f(\mathbf{l}, \mathbf{v}) = \frac{\mathbf{c}_{\text{diff}}}{\pi} + \frac{F(\mathbf{l}, \mathbf{h})G(\mathbf{l}, \mathbf{v}, \mathbf{h})D(\mathbf{h})}{4(\underline{\mathbf{n} \cdot \mathbf{l}})(\underline{\mathbf{n} \cdot \mathbf{v}})}$$

Typically in games, the diffuse albedo of a material is created in textures by artists. In the past, this has been viewed as something entirely art-driven and subjective. However, as can be seen from the above equation, diffuse albedo is in fact a physical property of a material, with real consequences if incorrect data is provided, such as the following problems:

- Diffuse albedo textures can be authored to be inadvertently too dark, causing materials to look black in poorly lit environments.
- Diffuse lighting is imbalanced compared to the specular lighting.
- Objects appear out of place, as their diffuse albedo mismatches that of their environment.
- Artists struggle to create life-like saturated colours and instead desaturate their textures.

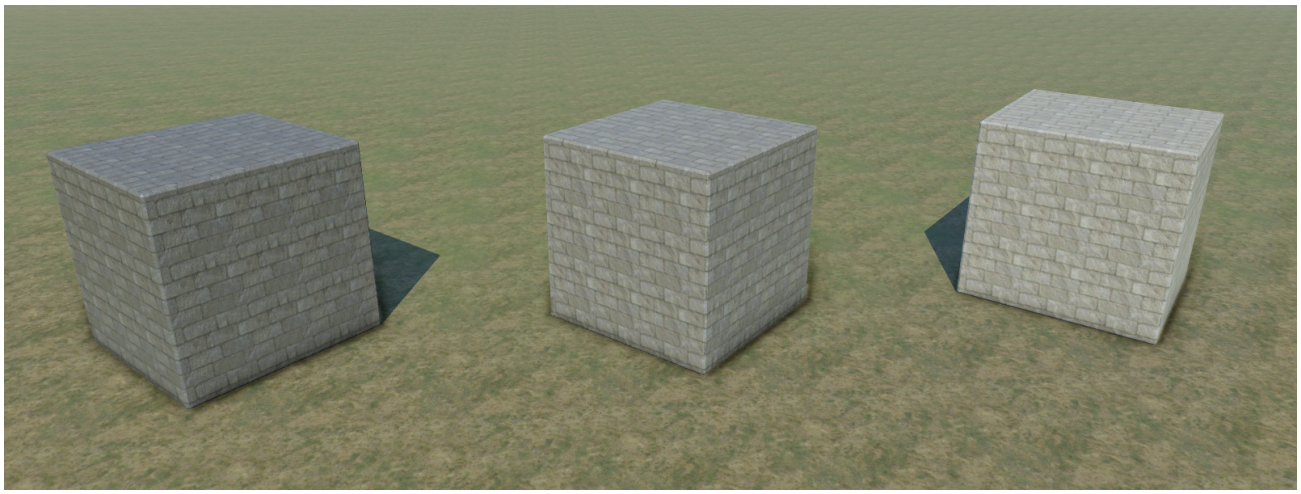


Figure 2: Imbalanced diffuse albedo can make objects appear out of place. Here, the grass is a calibrated texture. Left: diffuse albedo at 50%; Middle: calibrated diffuse albedo; Right: diffuse albedo at 200%.

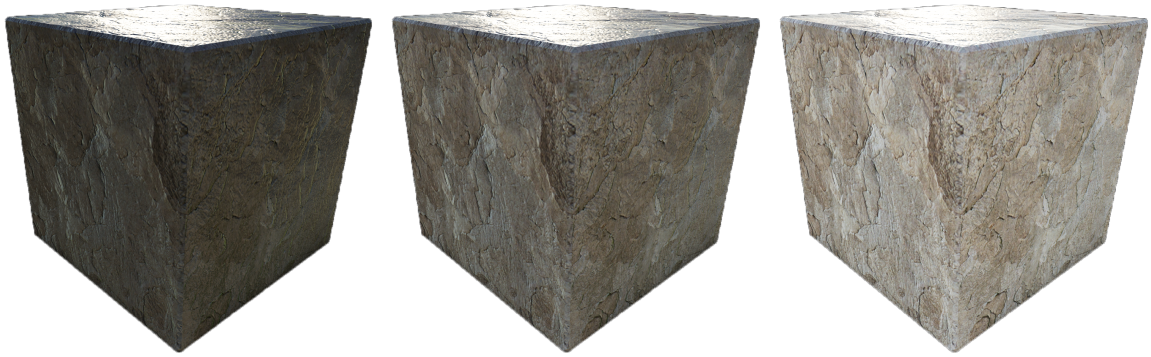


Figure 3: Imbalanced diffuse albedo can make specular appear too bright or too dark, changing the perception of the material. Notice how the material on the left looks metallic. Left: diffuse albedo at 50%; Middle: calibrated diffuse albedo; Right: diffuse albedo at 200%.

The last problem was particularly relevant for *Far Cry 3* since saturated colours were so vital to the art direction of the game. Even worse, desaturation is also used to combat the mismatches in diffuse albedo between materials, compounding the problem. Coupled with the desire for materials to behave under all lighting conditions and for diffuse and specular lighting to be balanced, we noted the importance of calibrating our diffuse albedos to physically-based values.

Photographing Materials

After committing to physically-based diffuse albedo values, we needed to obtain data for every material represented in game. Some diffuse albedo information could be found online, but such freely released data is either unavailable for all the materials we needed to represent, or was provided only in monochrome (as opposed to tri-chromatic) in many cases. Given that most of our texture reference comes from photographs, an ideal approach would be to leverage this representation and capture the albedo data directly from the photographs themselves.

We thus needed a reference object with known diffuse albedo values against which to calibrate our images. The Macbeth ColorChecker® was an ideal choice, consisting of twenty-four patches of colour spread across the visible spectrum, and is widely used for colour balancing in film and photography.

Figure 4: The Macbeth ColorChecker® placed within a photograph. We wish to measure the diffuse albedo of the wall behind the reference chart.

By placing this calibration target into the image alongside the material being captured, we would be able to find a transform that takes the twenty-four colour patches from their photographed colour to their actual sRGB values. Applying this transform to the whole image would yield the diffuse albedo values for the material.

This relies on the lighting being consistent across both the material and the Macbeth ColorChecker®, as well as there being minimal specular reflectance as we assume only diffuse lighting is present. The quality of the photography will greatly impact the final result, thus we decided upon the following set of principles:

- Minimise the direct lighting and specular reflection in the scene. For outdoor photography, overcast conditions are ideal.
- Place the Macbeth ColorChecker® parallel to the surface and the camera plane, ensuring lighting is consistent across the two while minimising specular reflection.
- Adjust the exposure of the camera so all twenty-four patches are adequately exposured, then perform white balancing.

Once a set of photographs had been taken, we turned our attention to finding the aforementioned mapping from the Macbeth patch colours to the sRGB space.

Colour Correction

We now have a set of twenty-four colours from our photograph that correspond to the twenty-four patches in the Macbeth ColorChecker®, and we wish to find a transform from these colours from the photograph to their values in sRGB space. Finding an exact transform is difficult due to the large set of data, leading to computational expense as well as increase difficulty in selecting an appropriate algorithm. Instead, we use an algorithm developed by Paul Malin [11], and consider two types of transforms that would give us a good approximation.

First, an affine transform, which would account for cross-talk between the red, green and blue channels, but as it is only linear it cannot accurately (e.g., non-linearly) adjust levels:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \begin{bmatrix} x_r \\ x_g \\ x_b \\ 1 \end{bmatrix} = \begin{bmatrix} y_r & y_g & y_b \end{bmatrix}$$

Second, a polynomial transform, applied per channel, that would be better at balancing levels whilst ignoring cross-talk:

$$\forall i \in \{r, g, b\}, y_i = \sum_{0}^{m} a_m x_i^m \ .$$

As the strengths and the weaknesses of both types of transform complement each other, using the two sequentially would most likely provide an accurate result.

In order to find the best possible affine and polynomial transforms, we used a linear least squares approach [1]. This states that, given matrices $\mathbf{X}$ and $\mathbf{Y}$, if we wish to find another matrix, $\mathbf{A}$, such that:

$$\mathbf{XA} = \mathbf{Y} \ ,$$

then the best possible approximation to $\mathbf{A}$ (in a least-squares sense) is calculated as follows:

$$(\mathbf{X}^\mathsf{T}\mathbf{X})\hat{\mathbf{A}} = \mathbf{X}^\mathsf{T}\mathbf{Y} \Rightarrow \hat{\mathbf{A}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{Y} \ .$$

This is easy to solve, as the inverse of $(\mathbf{X}^\mathsf{T}\mathbf{X})$ can be calculated using Gauss-Jordan elimination [2]. Thus once we have created such a solver, we merely need to plug in the appropriate values for X and Y in order to find our transforms.

For the affine transform, we are trying to find a matrix A, such that:

$$\begin{bmatrix} x_{0,r} & x_{0,g} & x_{0,b} & 1 \\ x_{1,r} & x_{1,g} & x_{1,b} & 1 \\ \vdots & & & \vdots \\ x_{n,r} & x_{n,g} & x_{n,b} & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix} = \begin{bmatrix} y_{0,r} & y_{0,g} & y_{0,b} \\ y_{1,r} & y_{1,g} & y_{1,b} \\ \vdots & & \vdots \\ y_{n,r} & y_{n,g} & y_{n,b} \end{bmatrix}$$

For the polynomial transform, for each channel i we wish to find the polynomial coefficients a_j, $j \in [0, m]$. These can be found be solving the following equation:

$$\forall i \in \{r, g, b\}, \begin{bmatrix} 1 & x_{0,i} & x_{0,i}^2 & \dots & x_{0,i}^m \\ 1 & x_{1,i} & x_{1,i}^2 & \dots & x_{1,i}^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,i} & x_{n,i}^2 & \dots & x_{n,i}^m \end{bmatrix} \begin{bmatrix} a_{0,i} \\ a_{1,i} \\ a_{2,i} \\ \vdots \\ a_{m,i} \end{bmatrix} = \begin{bmatrix} y_{0,i} \\ y_{1,i} \\ y_{2,i} \\ \vdots \\ y_{n,i} \end{bmatrix}$$

For our final colour correction tool, we experimented with different sequences of transforms. We used the residual sum of squares to estimate the error, which is calculated as followed:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

where y_i are the transformed colours and $\hat{y}_i$ are the actual values of the Macbeth ColourChecker®. Our best results were given with applying the following sequence of transforms: an affine, a cubic

Figure 5: A photograph before and after colour calibration. Left: before calibration; Right: after calibration.

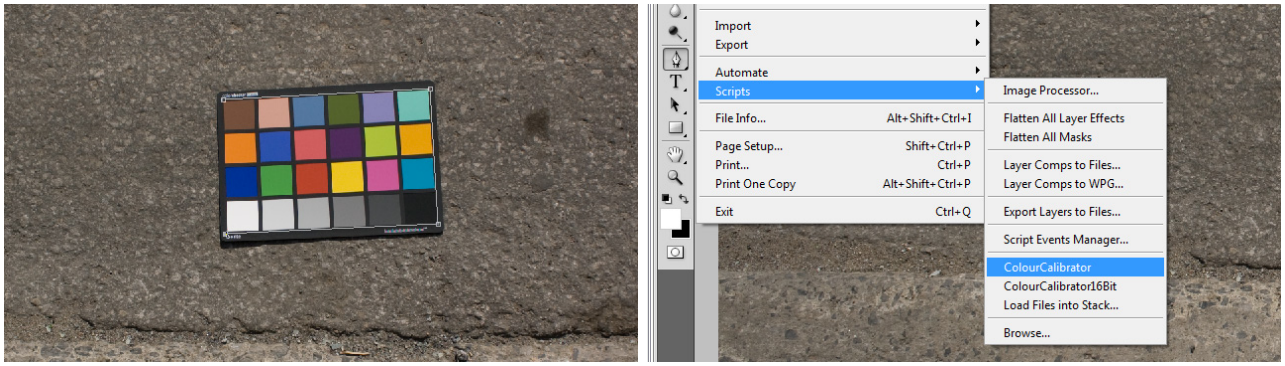


Figure 6: Using the colour calibration tool. 1) Click around the Macbeth ColorChecker® using the pen tool. 2) Run the ColourCalibrator script.

polynomial and a final affine. These were also applied in xyY space instead of sRGB, as we found that it better preserved the white balance of the photographs by separating out luminance from chrominance. Our final errors were in the range 0.03 to 0.1 (with sRGB space colours in the range $[0, 1]$, depending on the photograph. Unfortunately, applying more iterations of the affine and polynomial transforms did not produce improved results, possibly because at that stage, neither of these transforms provided an accurate approximation between the transformed colours and the Macbeth ColourChecker® patches.

Tool

The colour correction is applied using a command line tool, which takes an input photograph and the co-ordinates of the colour checker within said image, and outputs the calibrated image. However, this is obviously difficult for artists to work with, so we made a Photoshop script to invoke the tool. The artist simply clicks around the colour checker in the photograph, then runs the script. This launches the command line tool, then when it is complete the script promptly loads the resulting image into Photoshop for the artist to begin editing.

Problems

Overall, the end results were satisfactory. For our environment we achieved bright, vivid colours that were well-balanced, which met our goals and made calibration an essential part of our art process.

Figure 7: Left: original photograph. Right: after lens and colour correction. Images courtesy of Stephen Hill.

However, a number of problems arose during the project having to do with both workflow and understanding.

First, as we started calibrating fairly late in the project, we were unable to obtain references for all materials that we wished to place in game. Research trips to take photographs would have been ideal, especially to capture the wide variety of tropical vegetation present in *Far Cry 3*. However, good artists can work around this problem as long as they have access to some calibrated materials, and they can estimate semi-correct values.

Even when working directly from calibrated photographs, artists initially experienced difficulties understanding the difference between what was diffuse albedo, and what was baked lighting. Having some form of shadowing baked into textures is almost unavoidable, especially when looking at materials like grass. The calibration process tended to remove such shadows and made the textures look far too flat. Although, ideally, diffuse albedo textures should contain no baked lighting, and any shadowing or ambient occlusion should be captured in additional textures, combining these details with diffuse albedo into one texture is acceptable with current-generation hardware and performance limits. After explaining to artists that they still needed to preserve such details in a texture, even after calibration, they quickly adapted.

The biggest problem was the tinting of diffuse albedo textures in the materials. Artists could set an HDR colour with which to tint the material. The vast majority of the time, this resulted in many materials being far too dark, causing the problems mentioned above with balancing against specular and lighting conditions. In the future, we aim to tightly control colour tinting, favouring hue and saturation adjustments over luminance, and alerting artists whenever they step outside of the Macbeth ColorChecker® gamut.

Future Work

Although functional, the current calibration tool is unwieldy and difficult to use, and does not support batch processing of photographs. Our goal is a one-click solution, thus we need to add automatic detection of the Macbeth ColorChecker® to the tool, which would remove the necessity of human interaction and make processing vast numbers of photographs much quicker.

To improve the quality of the photographs, we are investigating ways of removing camera response and distortion. Raw sensor data can be obtained using `dcraw` [3], which reduces the burden on the colour correction tool to remove the camera response from the image in addition to the lighting. For lens correction, we have experimented with the Adobe Camera Model [**?**]adobecameramodel), fixing artefacts caused by vignetting, fisheye distortion and others. This is especially useful when creating textures directly from the photographs, saving artist time and resulting in a higher quality end product.

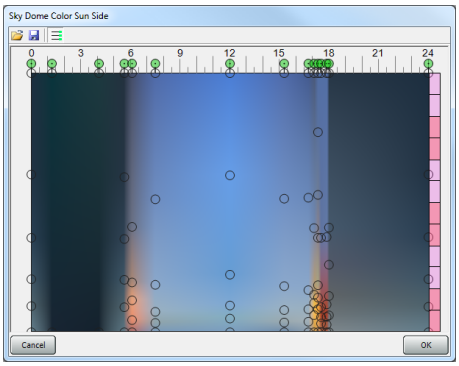


Figure 8: An example of a sky gradient. The x axis is time of day and the y axis is elevation above the horizon.

Finally, better methods are needed for removing any specular reflection from materials when attempting to capture diffuse albedo. One possibility is the use of polarised light. If the material is lit solely by a light of a specific polarisation, then a camera with a polarising filter set at the correct angle would only capture the diffuse response. However, this greatly restricts the material capture process, since a material sample needs to be captured under that particular lighting environment. In addition, the use of the direct lighting will bake self-shadowing into the material data. Yet for select materials this could still produce a higher quality result, and is well worth investigating.

Calibrating Lighting

Motivation

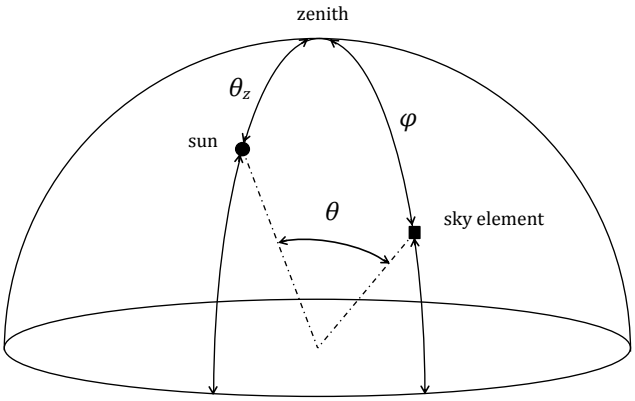
Calibrating lighting is equally as important as calibrating diffuse albedos. Given our outdoor environment, we spent most of our focus on sky lighting. Typically in games, ambient lighting and sky model are decoupled which is incorrect when we think of real life. This causes lighting imbalances to appear in many games, and actually leads to less interesting ambient lighting effects.

Sky Model

At the beginning of the project, the sky was set up by artists using two gradient textures, spanning from the zenith to the horizon. The two gradients were blended based upon azimuth angle, with 0° always aligned with the sun. This gave artists their desired control over colour, however, the gradients only contained LDR values and artists were unable to accurately reproduce the physical variation in sky brightness.

To improve this, we investigated a number of physically-based sky models. However, any accurate scattering model, such as [14], is very difficult for artists to control and we were also nervous about the performance cost and the length of time to implement such a system. This motivated our decision to evaluate the CIE sky model, which models the luminance distribution of a variety of different skies, from overcast to clear. By using this model for luminance and keeping the gradients for hue and saturation, we would give the artists their desired control whilst making our sky physically-based and fully HDR.

In the CIE sky model, the luminance of any sky element is given relative to the luminance of the zenith, and is dependent upon both the angular distance of said element from the zenith and the sun, as well as the angular distance from the sun to the zenith. The standard formula is given in [7]:



	a	b	c	d	e
CIE clear sky model	-1.0	-0.32	-10.0	-3.0	0.45
Far Cry 3 sky model	-1.0	-0.08	-24.0	-3.0	0.30

Table 1: The difference between the CIE clear sky model and the model used in *Far Cry 3*

$$\frac{L_{\theta,\phi}}{L_{\theta_z}} = \frac{(1 + a\exp(\frac{b}{\cos\phi}))(1 + c(\exp(d\theta) - \exp(d\frac{\pi}{2}) + e\cos^2\theta)}{(1 + a\exp(b))(1 + c(\exp(d\theta_z) - \exp(d\pi/2)) + e\cos^2\theta_z)},$$

where:

$L_{\theta,\phi}$ is the luminance in any arbitrary sky element,
L_{θ_z} is the luminance at the zenith,
θ is the angular distance from the sky element to the sun,
ϕ is the angular distance from the sky element to the zenith,
θ_z is the angular distance between the sun and the zenith.

The constants a, b, c, d and e change the properties of the sky from overcast to clear. We began by picking the constants used the CIE clear sky model, which seemed the most appropriate for a tropical island with very few clouds in the sky. We then tweaked these in order to give the sky increased contrast to better fit our visual design specifications.

This sky model enabled us to have a natural, high-dynamic range sky, as well as giving us a much greater deal of variation (and resolution) than could be found in our sky gradients. However, we found that basing the model off the intensity at the zenith (which remains a constant unless scaled by an external value) was too difficult to control, thus instead we made the model relative to the intensity of the sun, allowing the zenith to get naturally darker as the sun approaches the horizon. Our final sky formula was therefore:

$$\frac{L_{\theta,\phi}}{L_{\theta_z}} = \frac{(1 + a\exp(\frac{b}{\cos\phi}))(1 + c(\exp(d\theta) - \exp(d\frac{\pi}{2}) + e\cos^2\theta)}{(1 + a\exp(\frac{b}{\cos\theta_z}))(1 + c(1 - \exp(d\pi/2)) + e)}.$$

Sky Lighting

Many games only support a single colour sky lighting model, that is decoupled from the sky model used for the sky dome. This gives an unnaturally flat sky light, devoid of the range of colours we see in even the indirectly illuminated regions of a scene under sky lighting, particularly at sunrise and sunset. This causes an imbalance between the lighting in the scene and the sky, making the latter look

Figure 9: Sky rendered with our CIE sky luminance model.

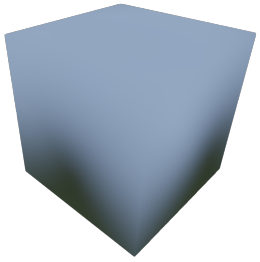


Figure 10: A light probe where a single colour is provided for the sky lighting. We also store sky and sun bounce lighting in the light probe.

either too bright or too dark. As a consequence, artists unnecessarily waste time tweaking both the uncoupled sky light and sky dome.

We instead chose to generate our sky lighting from the sky gradients and the CIE sky model, by using these to create parameterised second-order SH coefficients which are combined with the rest of the ambient lighting in our radiance transfer processing. This added zero GPU cost and only minimal CPU cost, leading to an improved result in terms of colour, with a mixture of reds and blues being present at sunrise and sunset.

However, although the variety of colour in the ambient was a welcome improvement, the balance between ambient lighting and the sky dome proved more problematic. Our art direction targeted a brighter sky and a less saturated ambient. At first, these limitations of the physically-based system were puzzling, until the realisation that the photographic reference used by art direction was captured using polarising filters.

A polarising filter blocks light with a certain polarisation. Light can become polarised in many

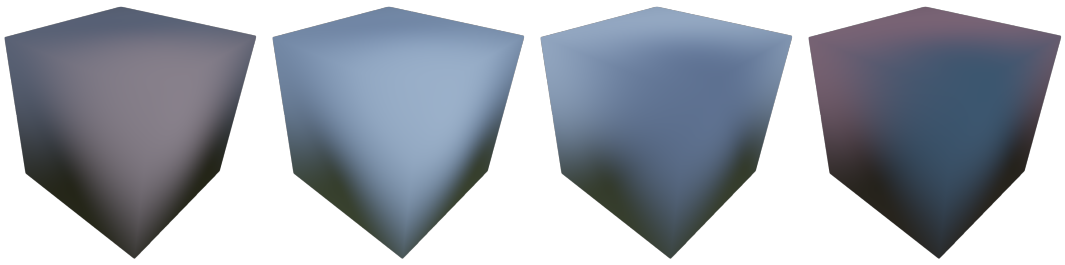


Figure 11: Light probe at various times of day using gradients and the CIE sky model for a non-constant sky color. From left to right: dawn, morning, afternoon and dusk.

Figure 12: Left: single colour sky light. Right: multi-coloured sky light.

Figure 13: Left: a scene without a polarising lens. Right: the same scene with a polarising lens. Attribution: PiccoloNamek at the English language Wikipedia.

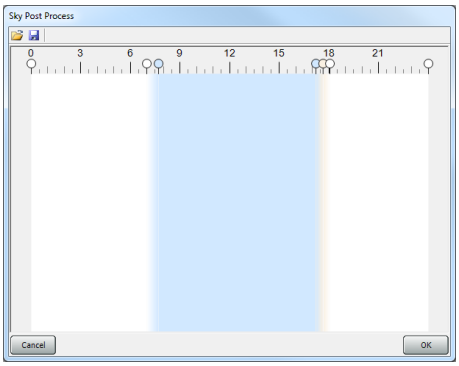


Figure 14: Colours used for a fake polarising filter. The x axis is time of day.

ways, including being reflected by a non-metallic surface (hence the use of polarising filters to remove specular highlights). Some of the light coming from the sky is also polarised. By using a filter set to an appropriate angle, a photograph can block some of the sky light, as well as sky reflection, to create a much deeper blue sky and overall clearer image.

We decided to cheaply simulate this by adding a "fake polarising filter" to our sky dome model. This is a simple scaling factor represented by an artist-specified colour, dependent on time of day, that is applied in the sky dome shader itself. This has the advantage that, when in-game cube maps are captured, the sky is already a deeper blue. This additionally simulates direct reflection filtering via polarisation.

We also allow artists to decrease the saturation of the ambient lighting to prevent it from becoming too blue. Together, these enabled us to achieve our visual target, but they certainly did not achieve our goal of facilitating the artist-driven balancing of ambient against the sky easier. We gained a richer ambient, but at the expense of artistic intuition and speed of development. However, we are planning to improve artist workflow in order to ideally eliminate these compromises.

Shading

Motivation

The shading model used at the start of *Far Cry 3* was not energy-conserving and lacked a Fresnel term. Both specular power and specular intensity from the Blinn-Phong equation were exposed to the artists, which was not only unintuitive, but they also had limited ranges greatly restricting the amount of materials that could be represented. This meant that our specular reflection was far too dark, with specular intensity often unable to go bright enough, and grazing angles suffered even more without a Fresnel term. Switching to deferred rendering also gave us an additional motivation to reduce the number of parameters and place them in intuitive, 0 to 1, ranges.

The Microfacet BRDF

We decided to use the Torrance-Sparrow microfacet BRDF [4],

$$f(\mathbf{l},\mathbf{v}) = \frac{F(\mathbf{v},\mathbf{h})G(\mathbf{l},\mathbf{v},\mathbf{h})D(\mathbf{h})}{4(\underline{\mathbf{n}\cdot\mathbf{l}})(\underline{\mathbf{n}\cdot\mathbf{v}})},$$

where l is the light direction, v is the view direction, h is the half-vector, n is the normal, F is the Fresnel term, G the geometry factor for shadowing and masking, and D is the normal distribution

function (NDF). Often the geometry term is combined with the terms on the denominator of this BRDF, and referred to as the *visibility* function [10]:

$$V(\mathbf{l}, \mathbf{v}, \mathbf{h}) = \frac{G(\mathbf{l}, \mathbf{v}, \mathbf{h})}{(\underline{\mathbf{n} \cdot \mathbf{l}})(\underline{\mathbf{n} \cdot \mathbf{v}})} \ .$$

The advantage of using Torrance-Sparrow is that you can choose which functions to use for the Fresnel, visibility and distribution functions. On current generation hardware, we searched for the fastest possible versions of each function, whilst still maintaining our desired physical basis.

For the distribution function, we chose Blinn-Phong [4] for performance:

$$\frac{1}{4} D(\mathbf{h}) = \frac{m+2}{8\pi} (\underline{\mathbf{n} \cdot \mathbf{v}})^m \ .$$

Where, following [10], we calculate specular power from a gloss value $g \in [0, 1]$, such that $m = 2^{13g}$.

We began with Schlick's approximation for the Fresnel term [4], but switched to the optimisation found in [9], where a spherical gaussian approximation to Schlick's approach was used instead, saving an additional shader instruction:

$$F_{Schlick}(\mathbf{v}, \mathbf{h}) = \mathbf{c}_{\text{spec}} + (1 - \mathbf{c}_{\text{spec}})(1 - \underline{\mathbf{h} \cdot \mathbf{v}})^5$$
$$F_{SG}(\mathbf{v}, \mathbf{h}) = \mathbf{c}_{\text{spec}} + (1 - \mathbf{c}_{\text{spec}}) e^{-6\underline{\mathbf{h} \cdot \mathbf{v}}}$$

For the visibility function, we chose Schlick-Smith, as it was the only feasible visibility function that is dependent on l, v and surface roughness, the latter being typically neglected in many other common visibility terms [4] [10]:

$$V(\mathbf{l}, \mathbf{v}, \mathbf{h}) = \frac{1}{((\underline{\mathbf{n} \cdot \mathbf{l}})(1-a) + a)((\underline{\mathbf{n} \cdot \mathbf{v}})(1-a) + a)} \ .$$

where $a = k\sqrt{\frac{2}{\pi}}$, with k being the Beckmann distribution roughness parameter, and thus can be calculated from the Blinn-Phong specular power m as follows [10]:

$$a = \frac{1}{\sqrt{\frac{\pi}{4} m + \frac{\pi}{2}}} \ .$$

Optimising the Visibility Term

Unfortunately, using the full Schlick-Smith visibility term was too expensive for our game. However, we wondered if we could approximate its effects with an average value, dependent only on surface roughness, by integrating and normalising the result over the unit hemisphere. Given that the surface area of the unit hemisphere is 2π, we can evaluate this integral as follows:

$$\begin{aligned}
&\frac{1}{2\pi} \int_{\Omega} \frac{1}{((\underline{\mathbf{n} \cdot \mathbf{l}})(1-a) + a)((\underline{\mathbf{n} \cdot \mathbf{v}})(1-a) + a)} \mathrm{d}\Omega \\
&= \left[\int_0^{\frac{\pi}{2}} \frac{\sin(\theta)}{(\cos(\theta)(1-a) + a)} \mathrm{d}\theta \right]^2 \\
&= \left[\frac{-\log(a)}{1-a} \right]^2 \ .
\end{aligned}$$

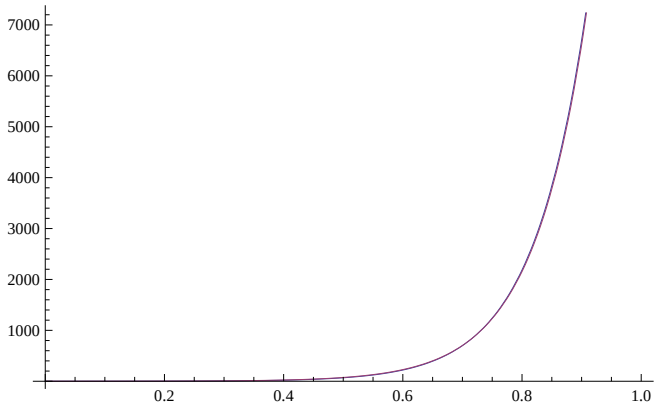


Figure 15: Comparison of E (blue) and E' (red) between 0 and 1.

Unfortunately, this term costs the same number as cycles as the original Schlick-Smith. However, given this term is now entirely dependent on roughness, we are able to combine it with the normalisation factor of our NDF and come up with an approximation of the two together. We decide to base our combined function on our gloss factor, as opposed to the Beckmann roughness or specular power, as this gives the most linear distribution of highlight sizes.

Recalling that specular power $m = 2^{13g}$, our combined function is:

$$E(g) = \frac{2^{13g}+2}{8}\left[\frac{-\log(\sqrt{\frac{1}{\frac{\pi}{4}2^{13g}+\frac{\pi}{2}}})}{1-\sqrt{\frac{1}{\frac{\pi}{4}2^{13g}+\frac{\pi}{2}}}}\right]^2 .$$

A quick exercise in curve fitting gives us an extremely accurate approximation costing only a single instruction more than the original Blinn-Phong normalisation factor:

$$E(g) \simeq E'(g) = 2^{-1.85689+16.1624g} .$$

However, this approximation did not produce a good visual result. As an average factor, it both overestimates and underestimates the intensity of specular reflection, and situations where the function overestimated looked particularly jarring. This is compounded by the fact that the Schlick-Smith visibility term is known to over-brighten specular reflection compared with experimental data [6]. Instead, we decided to look at functions of the form $E(g) = \frac{\alpha m+\beta}{8}$, where $\alpha = 1$ and $\beta = 2$ is the standard Blinn-Phong normalisation factor (divided by the 4 present in the microfacet BRDF). This would have the advantage of adding no additional cost to our shader.

Notice that we chose constants that would keep $E(0) = \frac{3}{8}$, so at the lowest glossiness specular highlights would keep the same intensity. This was for artistic reasons, since this is where unwanted over-brightening would be most noticeable.

We selected $E(g) = \frac{2m+1}{8}$ as our final combined normalisation and visibility function. As can be seen from the graphs, it is never greater than the average Schlick-Smith, which fits our requirement of always underestimating, but also importantly makes our specular highlights brighter than they were previously. Thus we are able to have a physically-based brighter specular reflection without the added cost of a visibility term.

Specular Filtering

Switching to a microfacet BRDF yielded many improvements, however, it also exposed the certain key limitations in our physically-based model, most notably the undersampling of our specular highlights,

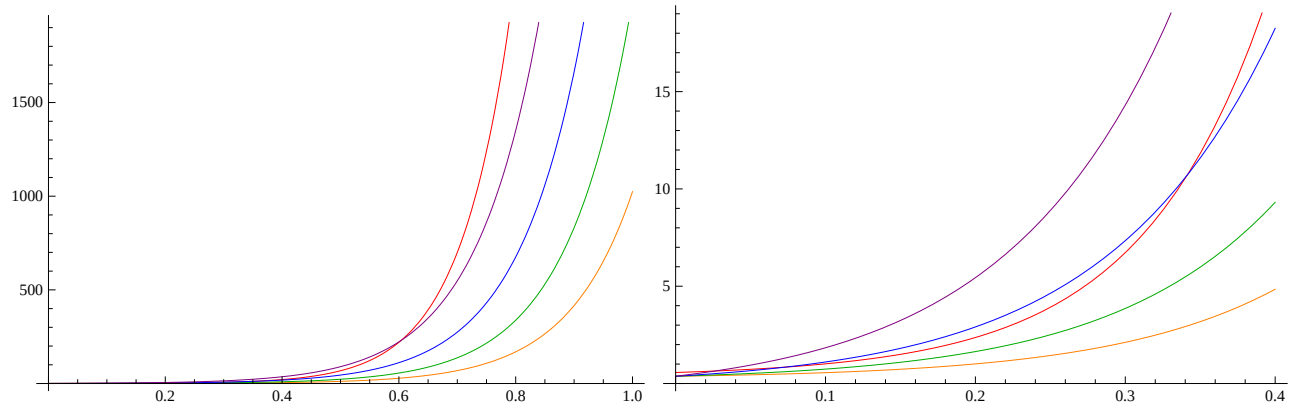


Figure 16: Comparison of approximations to the average Schlick-Smith function. Red: original function. Purple: $\frac{8m-5}{8}$. Blue: $\frac{4m-1}{8}$. Green: $\frac{2m+1}{8}$. Yellow: $\frac{m+2}{8}$.

causing shimmering artefacts and a change in appearance in the distance. The major reason for these problems is that we do not factor the variance of the normal map into our specular roughness. As we move into the distance, we sample lower mip-levels of the normal map, which are increasingly smooth and give the material the appearance of increased gloss. Instead, this variance that is lost in our normal map should be fed into our specular roughness, ensuring that appearance is preserved. Even at close distances, using the variance from just a bilinear filtered normal will help to prevent shimmering artefacts from occurring.

There has already been a lot of research on this topic, including LEAN Mapping [12] [5] and its cheaper variant CLEAN mapping [5]. The former even produces anisotropic highlights, but both involved too much memory and calculation to be feasible in *Far Cry 3*. Instead, we decided to follow the conclusions of [8] who, drawing on the work of [13] suggested creating a Toksvig map from gloss and normal textures to solve this problem.

A Toksvig map uses the variance of the normal map to estimate surface roughness, and therefore gloss. Given a specular power m, and a filtered normal $\mathbf{n}_a$, a new specular power can be calculated as follows:

$$m' = \frac{\|\mathbf{n}_a\| m}{\|\mathbf{n}_a\| + m(1 - \|\mathbf{n}_a\|)} \ .$$

As we sample increasingly lower mip levels of the normal map, normal variance increases and the specular power decreases. Thus given a gloss map and a normal map, we can actually create a new gloss map from this modified specular power which will take into account normal variance and correctly preserve appearance when anti-aliasing specular reflection in the distance.

Unfortunately, at this point in the project we were not using specular gloss maps, and the cost of adding an extra texture would have been far too high (for performance but especially for memory). We decided to let artists paint a gloss map into the alpha channel of the normal map in Photoshop. Our texture exporter and compressor then used these values to generate the new gloss map as described above, and the result was placed into the red channel of our DXT5 normal map (since only the green and alpha channels are used, for y and x normal components respectively), which could then be read for free in the shader along with the normal itself.

As might be expected, this had adverse affects on the compression of the Y channel that were often unacceptable. In these cases, the only option within our performance and memory constraints was to scrap the gloss map altogether, unfortunately losing the benefits of Toksvig mapping at the same time. However, we managed to find an acceptable compromise with the realisation that we could pack a

Figure 17: Left: scene with no specular filtering. Right: scene with average Toksvig factors.

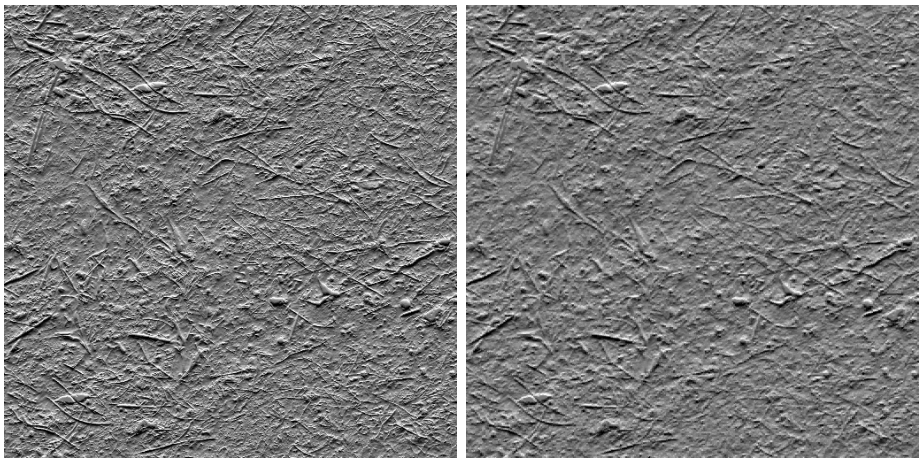


Figure 18: Placing a gloss or Toksvig map in the red channel of a DXT5 normal map negatively affects the compression of the normal Y component. Left: without a gloss map; Right: with a gloss map.

single colour into the red channel that obviously would not affect compression. We decided to calculate an *average Toksvig factor* by generating a gloss map from normal map variance using Toksvig, and averaging the result. This gave us some degree of appearance preservation and anti-aliasing without having to sacrifice the quality of our normal maps.

In the final game, the artists had the choice of using a gloss map or not, balancing higher quality specular against lower quality normals. However, even if they chose not to use a normal map, they still had the average Toksvig factor helping the appearance of their specular highlights.

Conclusion

Although it was introduced relatively late into the project, physically-based lighting, shading and materials have made a huge difference to *Far Cry 3*. Both the visual appearance of the game and the artist workflow improved, despite the problems that arose when weaknesses in the simulation were exposed. Morever, our physically-based approach had little to no performance impact, often in fact improving performance by removing now unnecessary hacks.

On the visual side, calibrating albedo textures has enabled us to have scenes with vivid, saturated colours without a continual need balancing on the art side. Changing to a physically-based BRDF has given us brighter specular reflection with reduced shimmering, and the new sky model not only enhances the appearance of the sky itself, but also increases colour variation in the ambient lighting.

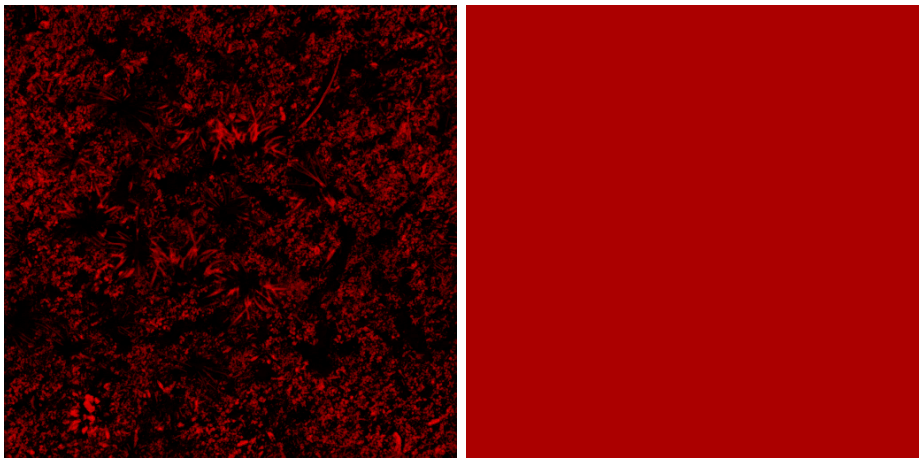


Figure 19: The two options for gloss maps. Left: gloss map adjusted by the Toksvig factor; Right: single colour averaged Toksvig factor.

Artists also liked the new workflow. Increased simplicity made lighting and materials easier to author, which was worth the trade off for the small number of cases where certain materials or certain lighting conditions became more difficult to represent. However, where problems did arise, they showed the need for better tools and more education, as initially authoring specular roughness textures feels counter intuitive, especially when there is no immediate feedback given to the artist.

Despite the benefits, numerous difficulties were encountered. We continually discovered areas that we were not simulating, from the ability to apply a polarising filter to the sky, to simply inaccuracies in our BRDF. Although we found solutions for many of these problems, we had to be careful that these solutions were not reintroducing complexity that had been removed with the shift to physically-based models. Again, this demonstrated to us the need for improved tools, always giving our artists the solid starting point of physically-based shading, but allowing them to tweak beyond the boundaries where necessary.

Acknowledgements

For their assistance on the calibration of diffuse albedo textures, many thanks go to Naty Hoffman of 2K Games for teaching me the theory and Paul Malin of Activision Central Tech for sharing his algorithm. Thanks to everyone on *Far Cry 3* at Ubisoft, especially Mickael Gilabert, Philippe Gagnon, Jean-Alexis Doyon, Vincent Jean and Minjie Wu. For proof-reading, thanks to Derek Nowrouzezahrai at Université de Montréal. Final thanks to Stephen Hill, for assistance in many of the ideas developed in these notes.

Bibliography

[1] http://en.wikipedia.org/wiki/Linear_least_squares_%28mathematics%29.

[2] http://en.wikipedia.org/wiki/Gauss-Jordan.

[3] http://cybercom.net/~dcoffin/dcraw/.

[4] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-Time Rendering*. AK Peters, third edition, 2008.

[5] Dan Baker. Spectacular Specular - LEAN and CLEAN Specular Highlights. In *GDC*, 2011. http://www.gdcvault.com/play/1014558/Spectacular-Specular-LEAN-and-CLEAN+lean+and+mean+specular.

[6] Brent Burley. Physically-Based Shading at Disney. In *SIGGRAPH*, 2012. http://blog.selfshadow.com/publications/s2012-shading-course/.

[7] Stanislav Darula and Richard Kittler. CIE General Sky Standard Defining Luminance Distributions. In *eSim*, 2002.

[8] Stephen Hill. Specular Showdown in the Wild-West, 2011. http://blog.selfshadow.com/2011/07/22/specular-showdown/.

[9] Sébastien Lagarde. Adopting a Physically Based Shading Model, 2011. http://seblagarde.wordpress.com/2011/08/17/hello-world/.

[10] Dimitar Lazarov. Physically Based Lighting in Call of Duty: Black Ops. In *SIGGRAPH*, 2011. http://advances.realtimerendering.com/s2011/Lazarov-Physically-Based-Lighting-in-Black-Ops%20(Siggraph%202011%20Advances%20in%20Real-Time%20Rendering%20Course).pptx.

[11] Paul Malin. Personal communication.

[12] Mark Olano and Dan Baker. LEAN Mapping. In *I3D*, 2010. http://www.cs.umbc.edu/%7Eolano/papers/lean/.

[13] Michael Toksvig. Mipmapping normal maps. Technical report, 2004. http://developer.nvidia.com/content/mipmapping-normal-maps.

[14] Carsten Wenzel. Real-time Atmospheric Effects in Games Revisted. In *GDC*, 2007. http://www.crytek.com/download/GDC2007_RealtimeAtmoFxInGamesRev.ppt.