Cinematic Character Lighting in

Ben Cloward
Senior Technical Artist
BioWare

Aaron Otstott
Senior Software Engineer
BioWare

Character lighting is a crucial part of the experience in Star Wars: The Old Republic. From in-game cinematics, to combat, to exploration, all of the game's major components rely on lighting to make the characters look heroic, distinguish them from the environment, and achieve the game's unique visual style. In designing the character lighting system, the team at BioWare faced several unique challenges.

CHALLENGES

- Achieve the Artistic Vision
- Make Characters Stand Out
- Automate Quality Lighting in Cinematics

ACHIEVE THE ARTISTIC VISION

Early on, the art team decided we wanted to create a distinct look for our game that would differentiate it from other games and make it unique. We also noticed that games that were created to be as realistic- looking as possible tended to look dated after a few years of graphics advancements.

With that in mind, our concept artists began experimenting with various visual styles and developed a look that we call “stylized realism.” In many games, the concept art simply serves as inspiration for the in-game art, but in our game, the art director challenged us to make the in-game art match the concept art as closely as possible. The goal was to be able to put a concept image next to a screen shot from the game and not be able to tell the difference. While some of the responsibility of achieving this goal belonged to the 3D artists, the tech artists and graphics programmers also had a big challenge to develop shaders that produced the same visual quality as the painted artwork created by out concept artists.

MAKE CHARACTERS STAND OUT

Second, we noticed that our characters had a tendency to blend into the background and get lost within the detailed environments, when lit with standard scene lighting.

This was a problem because we wanted our players to be able to notice other characters in the world, such as enemies, quest givers, and other players.

In our cinematics and conversations where the focus was supposed to be on the actors in the conversation, the business in the background often pushed itself forward, demanding the player's attention and drawing focus away from the actors.

AUTOMATE QUALITY LIGHTING

Finally, the lighting placed to illuminate our environments worked really well for the general mood of the area and to define the shapes of structures, but it was often less than ideal for characters or cut scenes. A cinematic conversation may take place in a dark cantina or in an outdoor area where light comes from a poor direction.

In film, each individual shot is very carefully lit to convey the meaning and mood that the director is looking for. Often, lighting artists will spend hours setting up the lighting for a single shot that may last only a few seconds. Film lighting is designed to look natural, but really isn't anything like the actual light found in the environment.

For our game, we wanted to achieve cinematic quality in lighting as though it was carefully hand-placed for each shot. At first, the art director wanted to have the artists place lights individually for each cinematic. However, this idea was quickly withdrawn when we estimated that it would take several months worth of crunch time, with every artist we had, to complete the job. We needed another solution that produced cinematic-quality lighting, but without requiring any manual light setup.

OUR SOLUTION

- Light Color Gradient
- Stylized Rim Lighting in Shaders
- Controlled Light Direction
- Sky Specular
- Depth of Field

We met these three challenges by introducing several key elements into the lighting system. We used a color gradient to delineate the color of the light sources. We added code in the shader to produce stylized rim lighting. We adjusted the directional light vector so that characters are always lit from a pleasing angle. And finally we added a broad specular highlight from the sky and depth-of-field in cinematic shots.

TALK OUTLINE

- Discuss each element - Light Color Gradient, Rim Light, Controlled Light Direction, Sky Spec, DOF
  - Mechanics
  - Evolution of the Idea
  - Results
- Show how all elements fit together to meet the challenges

In this talk, we're going to go in-depth to discuss each of these additional lighting elements. We'll explain the mechanics of each element, talk about how the initial idea evolved into the final implementation, and show the results. Finally, we'll show how all of the elements work together and compliment each other to produce cinematic lighting, achieve our goals, and overcome the challenges we faced.

BioWare
A DIVISION OF EA

LIGHT COLOR GRADIENT

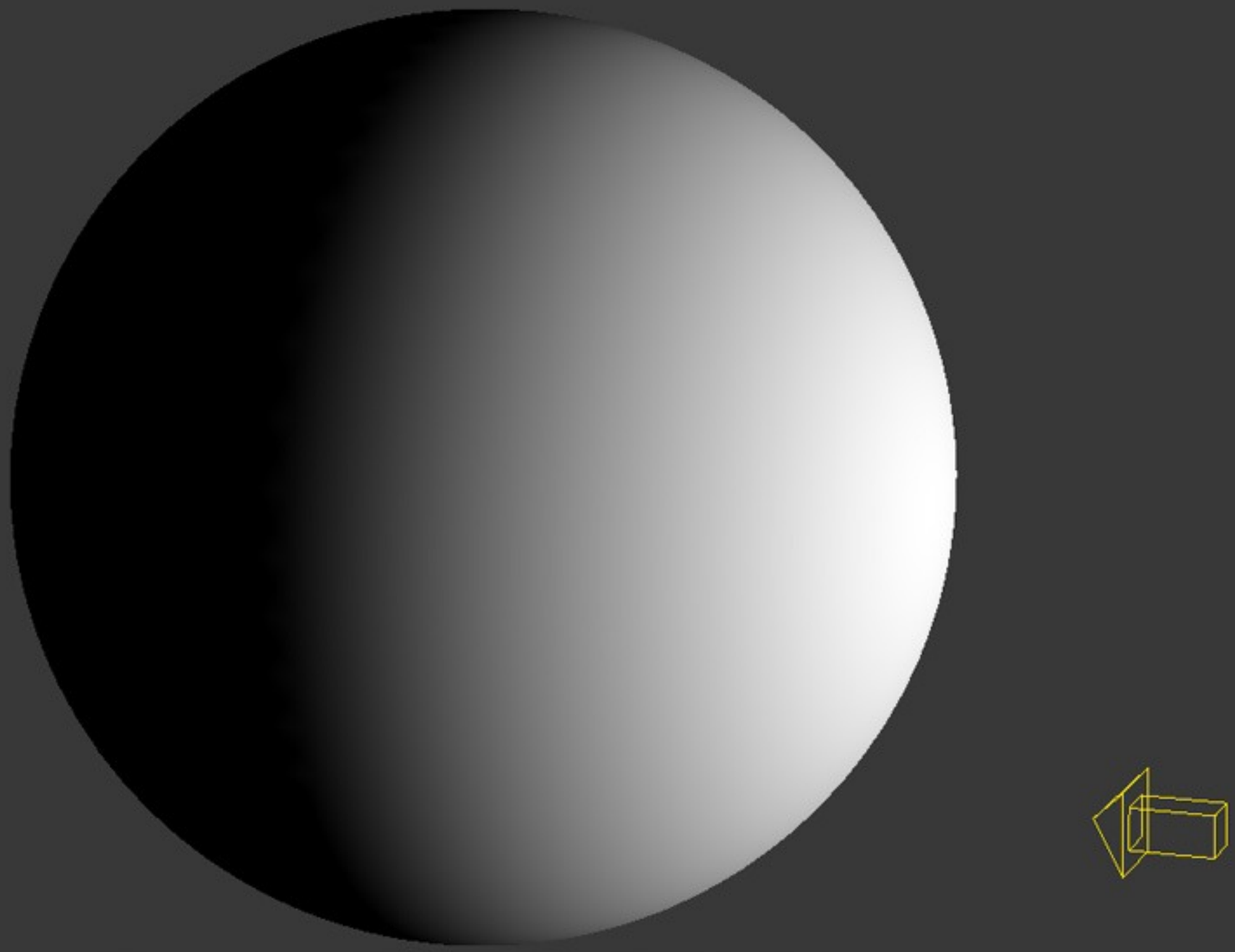
LIGHT COLOR GRADIENT

- A color gradient to represent the range of color from the light side to the dark side
- Uses Half-Lambert Function (N dot L * 0.5 - 0.5)
- Gives the artists precise control over light direct color, terminator color, and bounced light

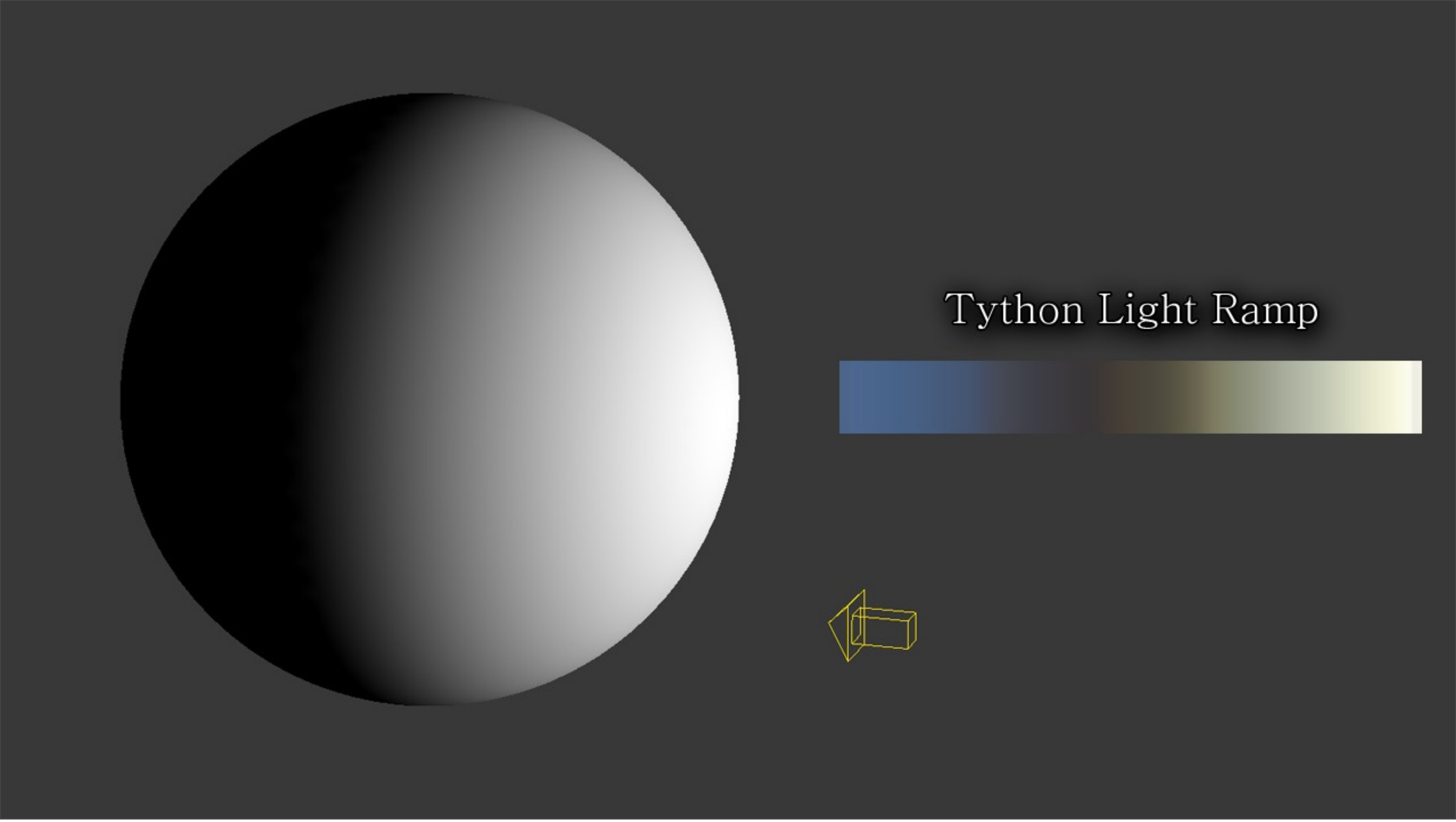
The foundation of our cinematic lighting system is something we call “Gradient Lighting.” Our artists create a color gradient for each light that represents the range of color from an object’s light side (facing the light) to the dark side (facing away from the light).

The coordinates for the gradient look-up are generated with a simple Half Lambert Function (N dot L * 0.5 - 0.5) which makes the gradient wrap all the way around the model. This is similar to the lighting technique used in Team Fortress 2.

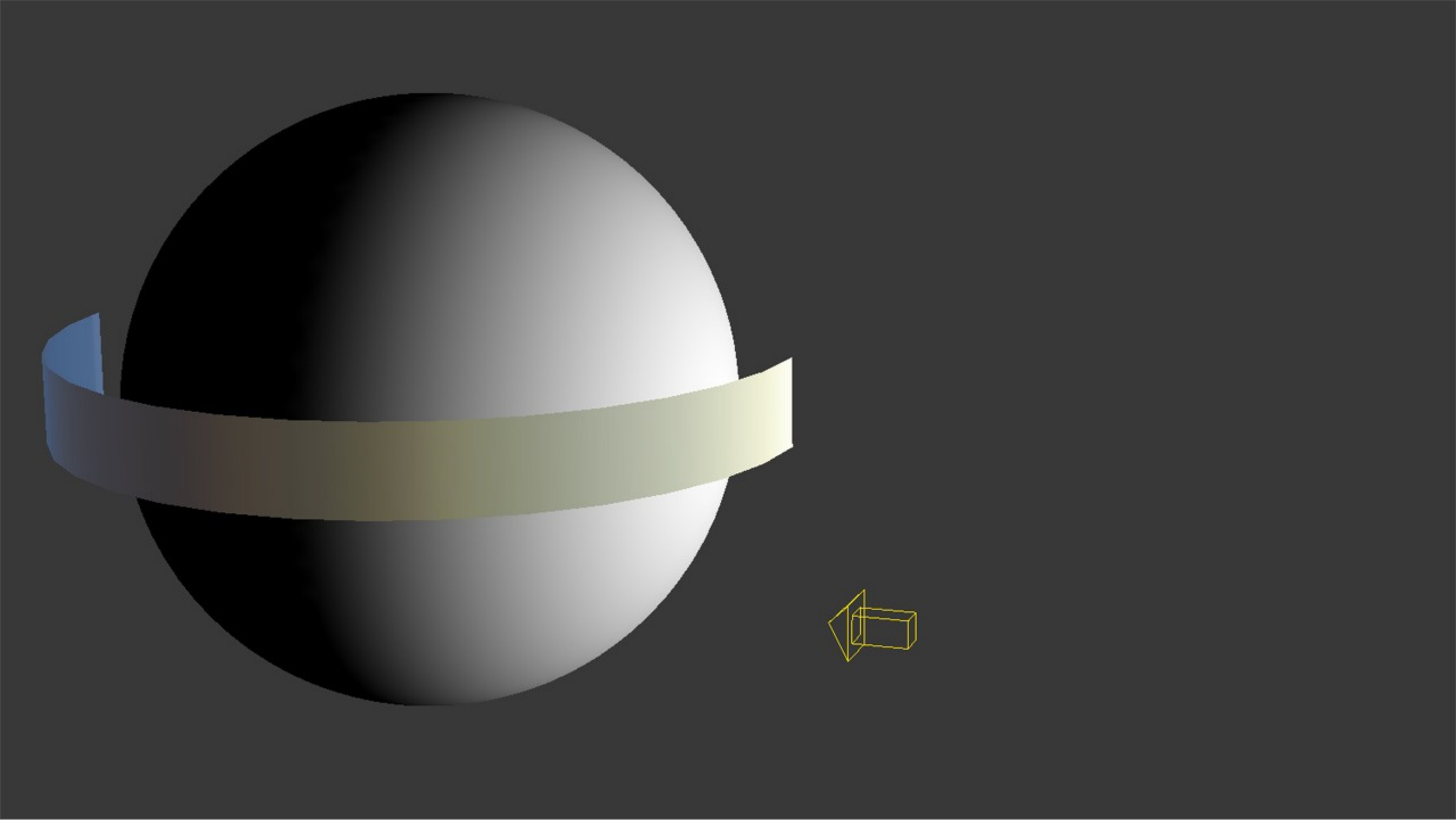
This gives the artists precise control over the light’s direct color, the color of the terminator, and even the color of the bounced light. With this technique our color gradient is not limited to a blend between light color and darkness.



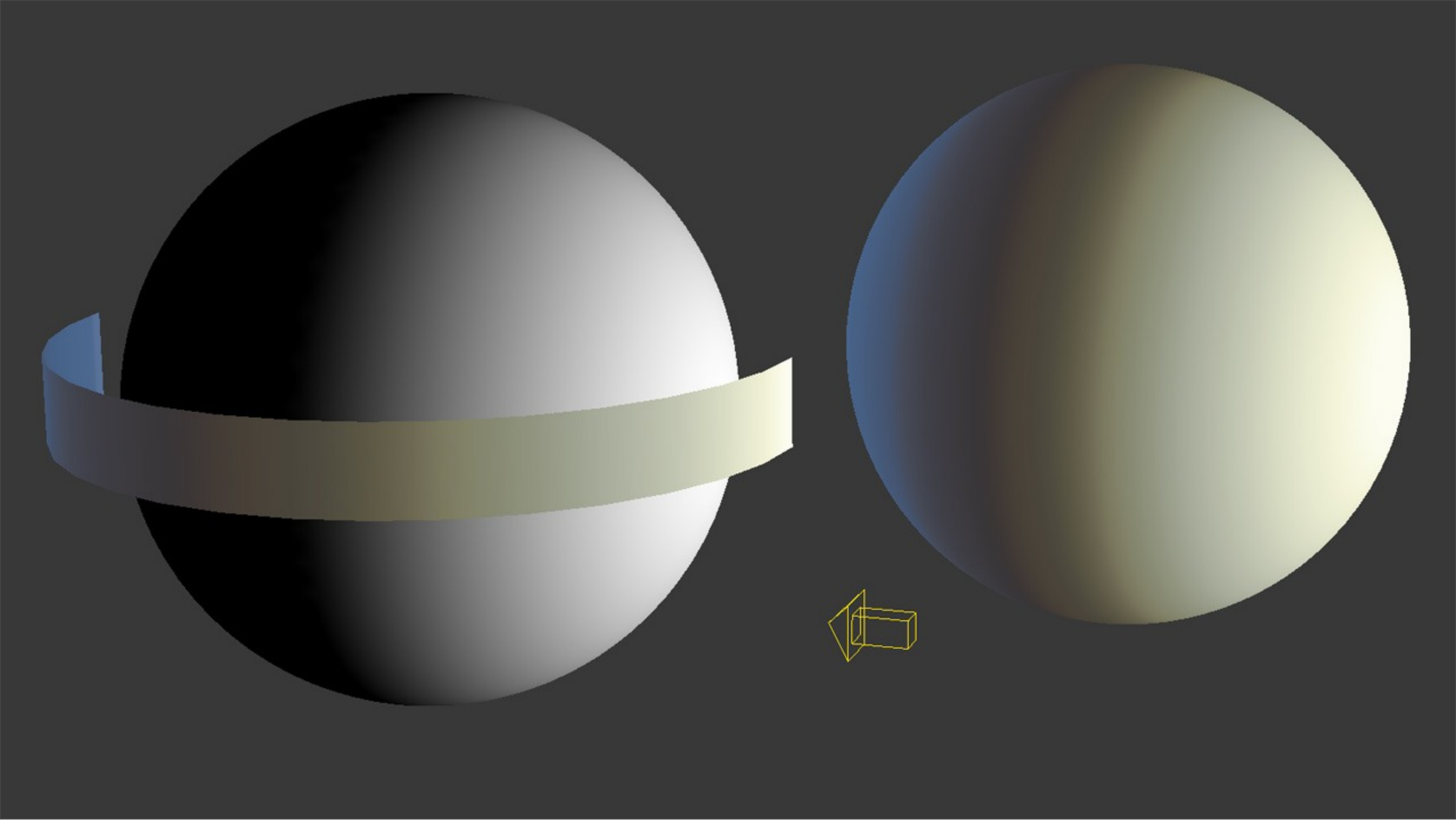
Here's a simple sphere lit with standard diffuse lighting. It looks pretty flat and with this type of lighting, the artist can only define a single color to control the appearance.



Instead, we give the artists the ability to paint a color gradient. Here's an example - the light gradient that the artists painted for the planet Tython. Notice that it's transitioning from a light cream color through a subtle green, to a brown and then to a blue on the left. The green represents light getting diffused through tree leaves because Tython is a forest planet. The blue represents indirect light coming from the bright blue sky. With a gradient, the lighting can provide all of this additional information.



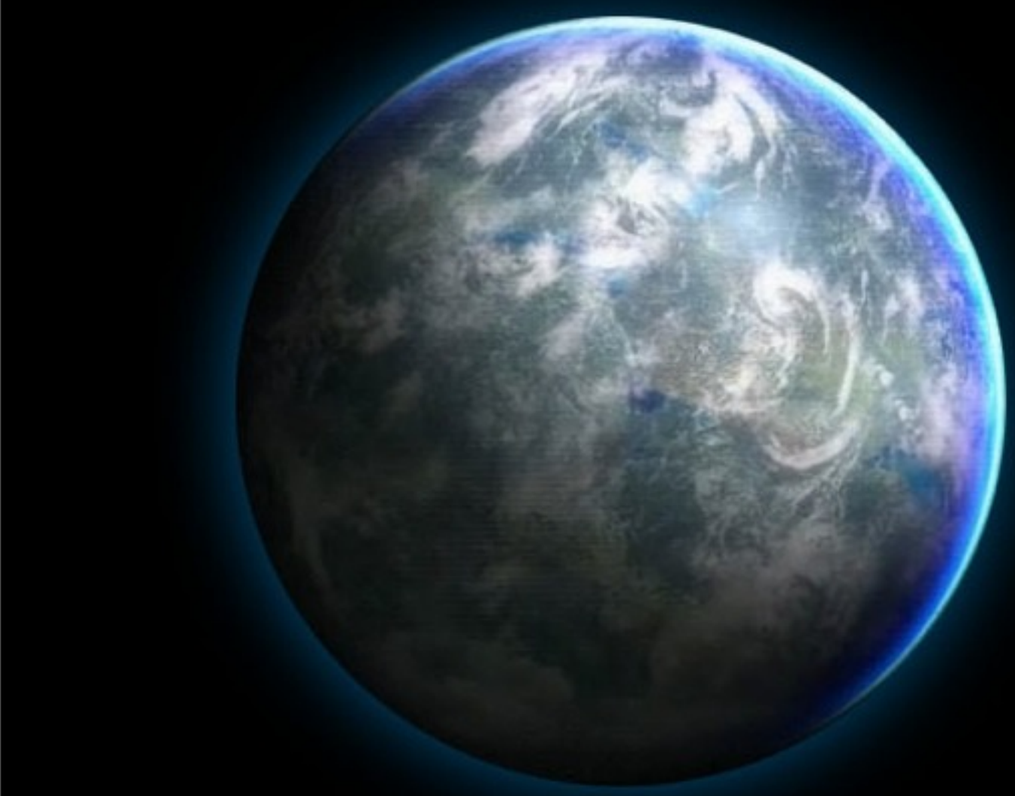
So we use the half lambert function to wrap this gradient around the model from the side facing the light to the side facing away from the light.



And we end up with a result like this. When compared with simple lighting using the NdotL function, you can see why we chose to use Gradient Lighting. It matches our concept art and allows the artists to introduce rich color combinations. We're able to achieve an illustrated look instead of the hard, lifeless appearance of plain NdotL lighting. This technique also gives us the ability to incorporate a bounced light color without resorting to more expensive shader techniques and without interfering with the direct light color. The artists are free to artistically paint the lighting equation through the gradient. This allows them to broaden the effect of the light or transition through a color palette. It produces more painterly and saturated visuals.

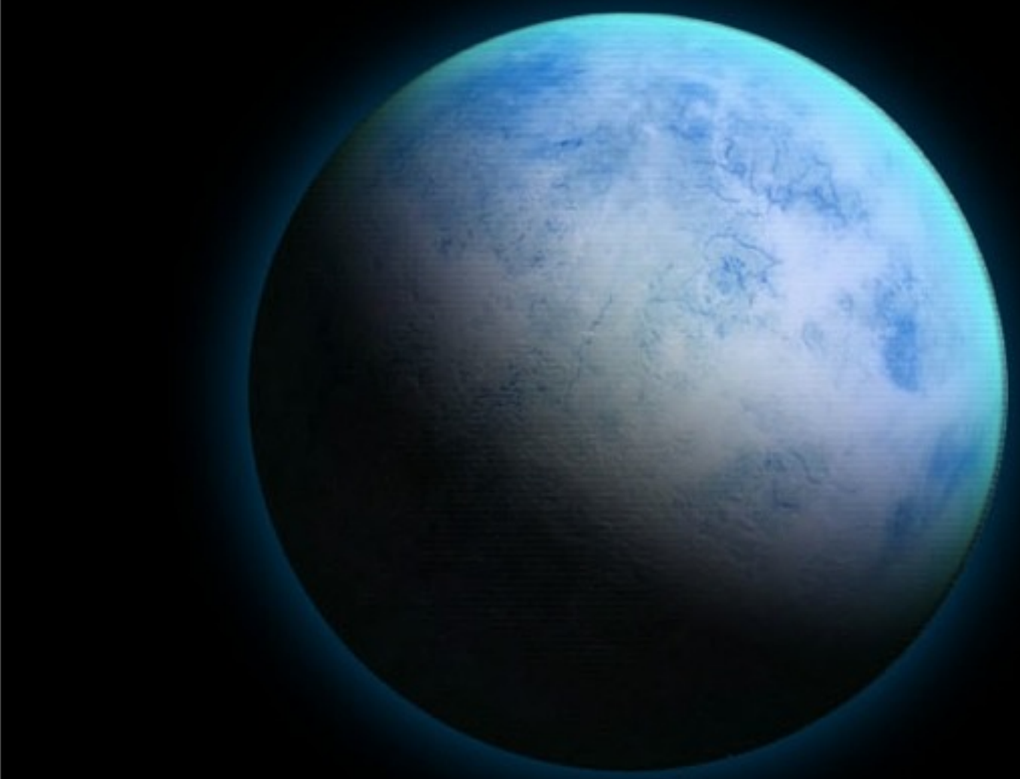
Each of our planets has a single gradient that defines the main appearance of the outdoor areas. Let's look at some examples.

TYTHON



There are a couple of things to notice here. First, the gradient is the single most important element for defining the atmosphere for each planet. Much of the mood of each environment can be defined simply by changing a few colors on the gradient. This makes it easy for the lighting artists to experiment and get the lighting style they want quickly.

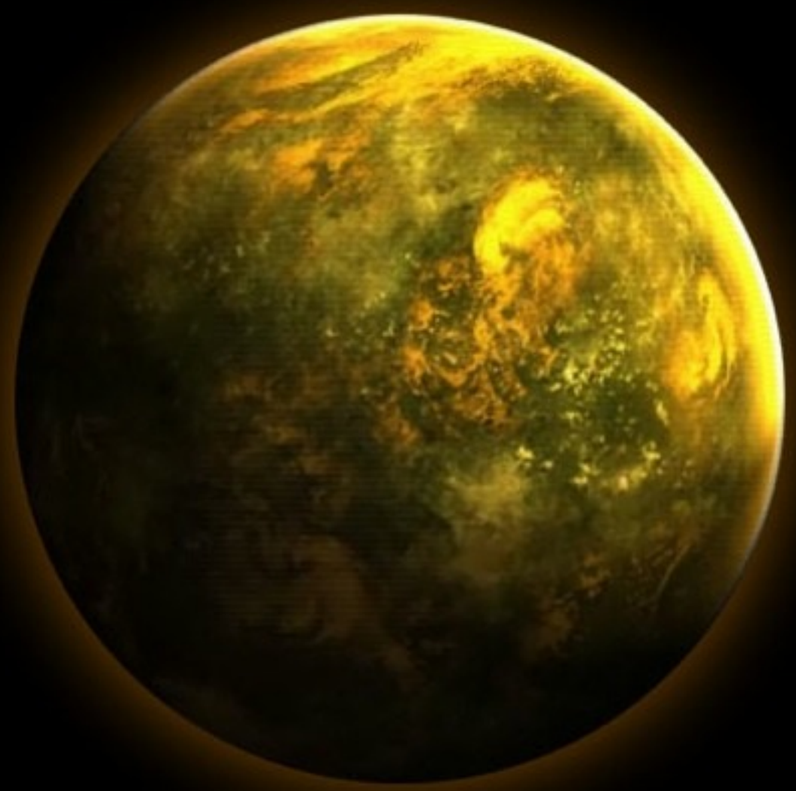
HOTH

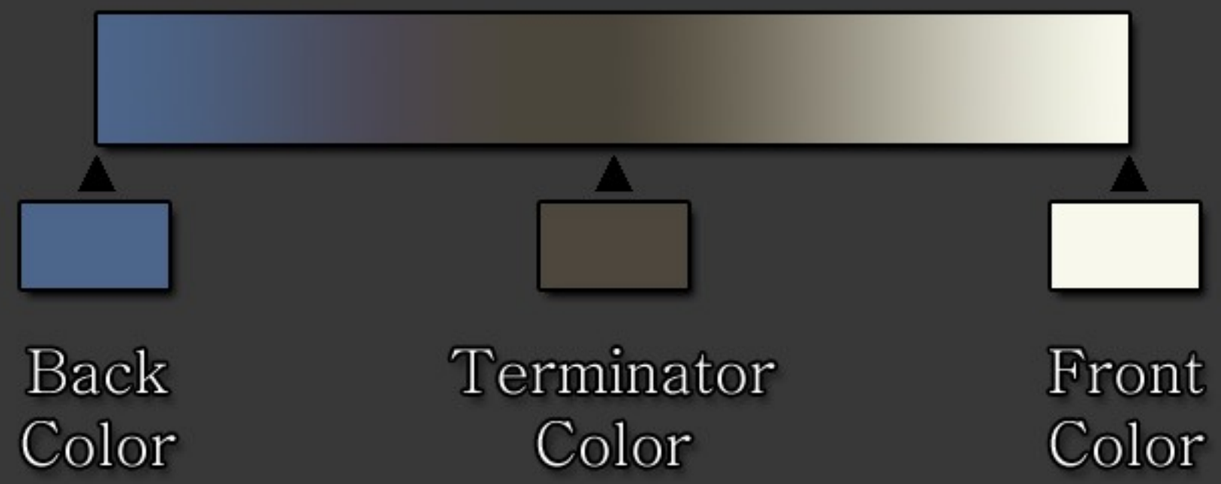


Also, notice the nice saturated quality of our lighting, both on the front and on the back sides of objects. We achieve this result because the lighting is defined by a single gradient instead of with several different lighting systems (diffuse, ambient, fill light, etc.) that often compete with each other and produce a washed-out or desaturated appearance.

KORRIBAN

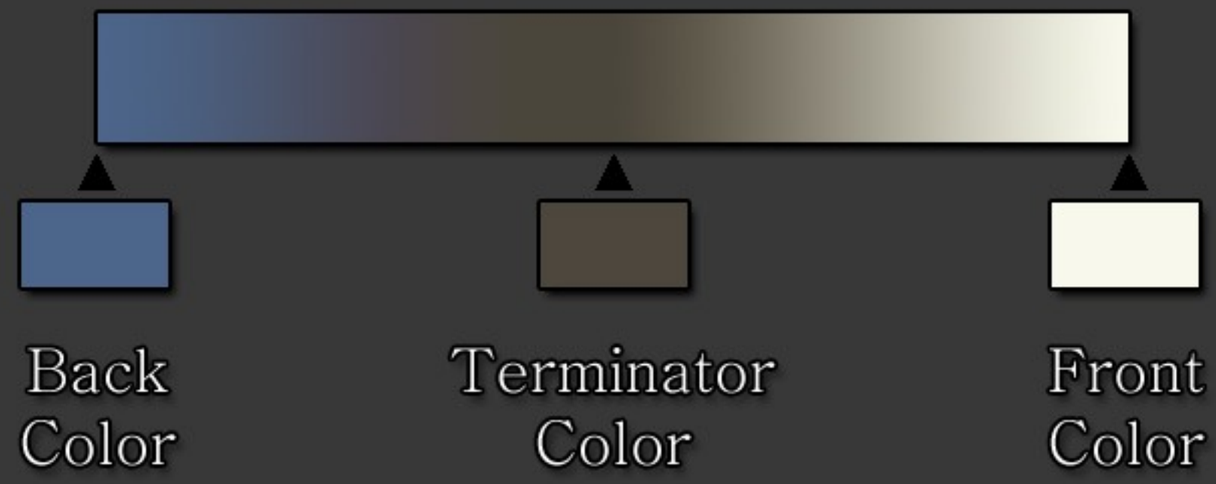
HUTTA





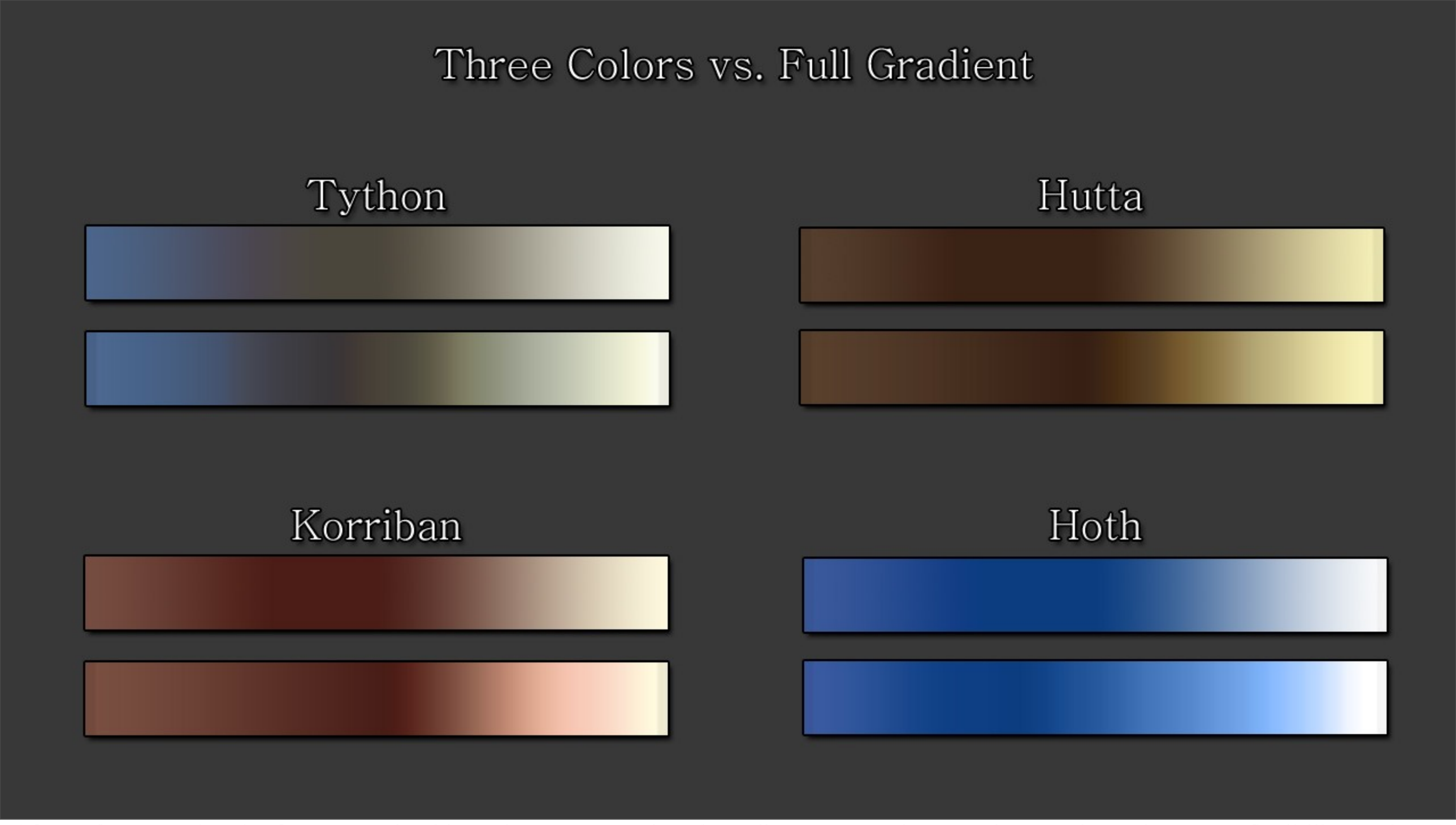
The lighting gradient is a feature that has been in the game from the beginning of development. At the start of the project, gradient lighting was limited to a direct color, a terminator color, and a backside color for each light.

Original Gradient – Just Three Colors



Not enough artist control

Although this method produced similar results to our current method, it provided less control and required more math in the shader.



After a few months, the feature evolved into the current painted gradient texture, because it gave the artists more flexibility and because the resulting pixel shader was cheaper.

Here, the gradient on the top represents the result with just three colors vs. what is possible with an entire gradient. Notice how much depth and extra detail the artists have added because they can control the whole gradient.

Let’s take a look at the visual quality we create through gradient lighting as opposed to standard lighting. The main thing to notice here is that the standard NdotL lighting tends to make things look flat and desaturated.

When we switch to the gradient lighting, we get rich color and the characters feel much more vibrant.

This guy almost looks like a zombie...

...until we use the color gradient to light him and help him fit into the environment.

And here's an example from Ord Mantel.

Using the color gradient for the lighting makes the character feel much more vibrant.

BioWare
A DIVISION OF EA

RIM LIGHTING

RIM LIGHTING

- In film, a light placed behind the character
- Separates the characters from the background
- Highlights shapes and contours
- Game rim light created in the shader
- Our rim light mimics the style of our concepts - not real light behavior

Rim lighting is the second element of our cinematic lighting system. In film, a gaffer will create rim lighting by placing a bright light source behind and slightly to the left or right of the actors.
The rim light serves to separate the characters from the background and highlight shape contours.

In a video game, the player often has control over the camera, so it isn't possible to place light sources that create rim light. Instead, rim lighting is created through shaders that find the silhouette edges and shade them to mimic the effects of a light placed behind the character. In our game, we developed our rim lighting, not to mimic the effects of real light, but to imitate the appearance of our concept art.

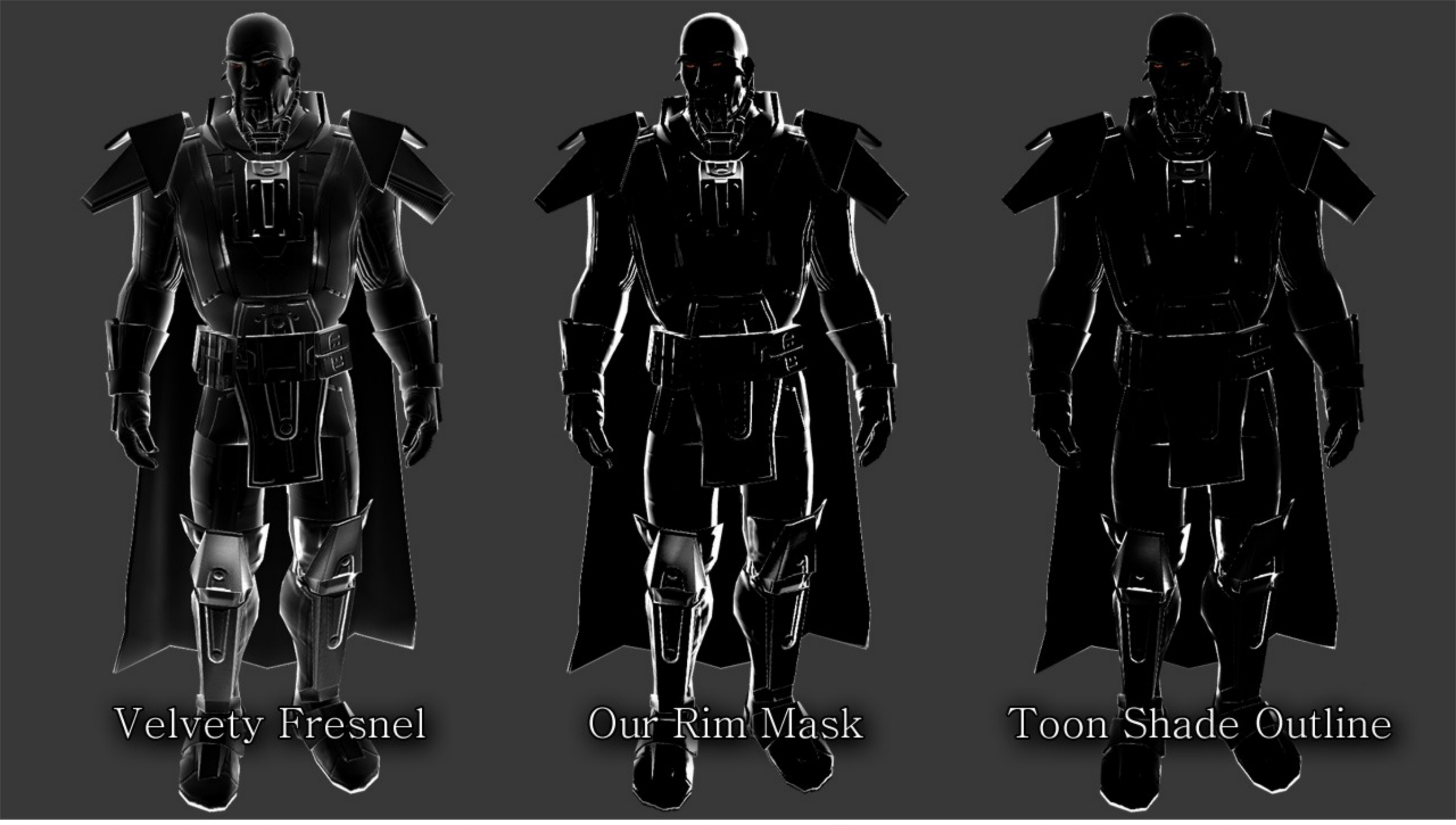
RIM LIGHTING EVOLUTION

- Began as a way to make skin look nice.
- Art director wanted to try it on everything.
- Original version was too velvety and got too blown out.
- The current implementation has gone through lots of refinements.

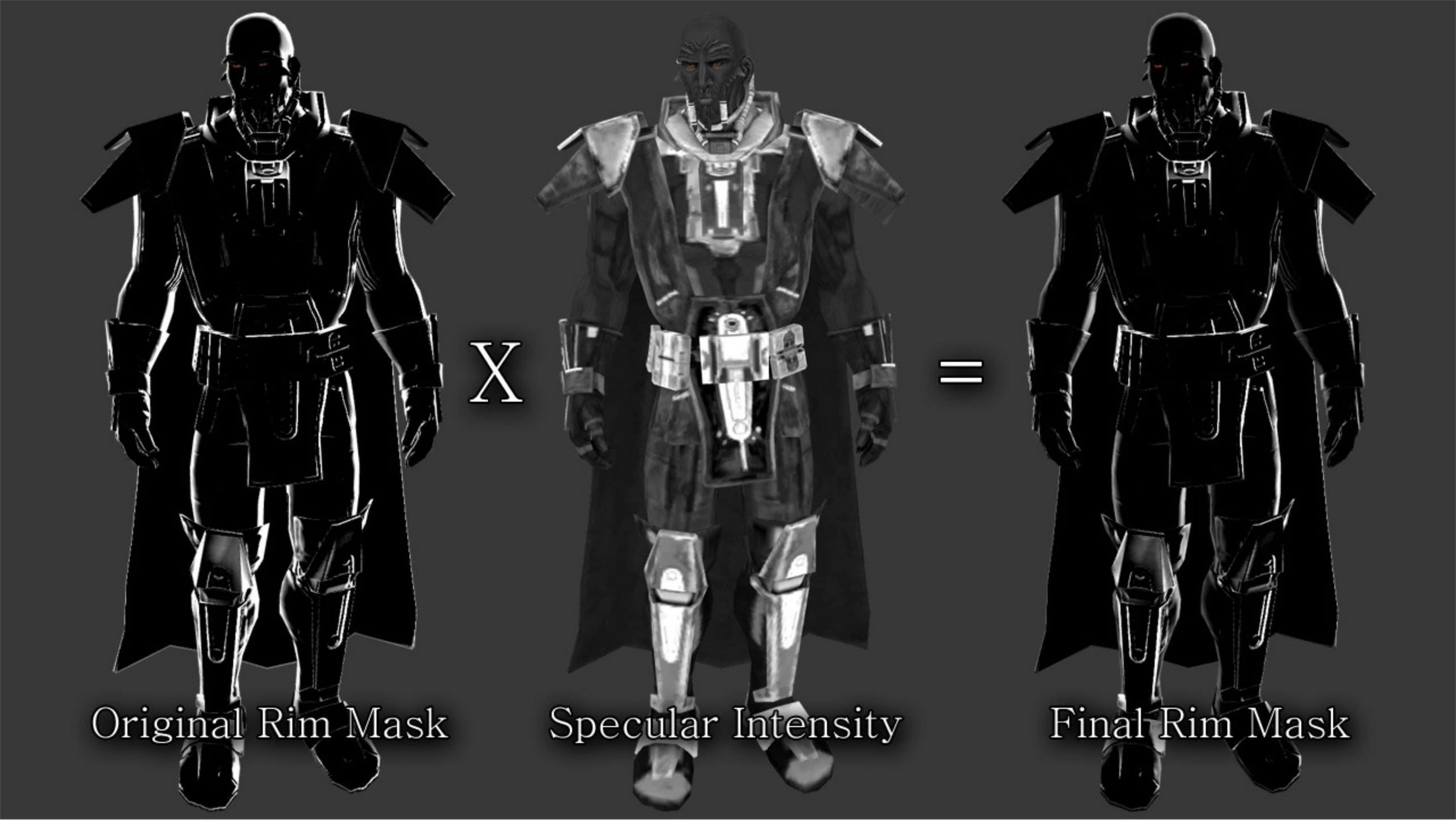
Originally, our rim lighting started out as a way to make our characters' skin look nice. I developed it as part of the skin shader that was applied to a character's face.
However, our art director liked the look and decided that we should apply it to the whole character. I took the code that I had written for the skin shader and added it to our other character shaders.
This original version had a velvety appearance, but because of the way I had implemented it, it had a tendency to get too blown out when there were multiple light sources, and it saturated toward white instead of the color of the light source.
In order to improve the rim lighting, we refined our shader code to produce the look of the rim lighting in our concept art.

So here's an example character without rim lighting. His outfit has a lot of detailed shapes, but they tend to get lost.

In order to add rim lighting, the first thing we need to do is calculate a rim mask.



We calculate a rim mask based on the surface normal and the view vector. The mask is a careful balance that sits about half way between a toon-shaded outline and a velvety fresnel fall-off.



Typically rim lighting is a uniform color with a uniform intensity. This can make objects appear to be of a single, uniform material. Instead, we control the intensity of rim lighting with the intensity of the specular color. This gives less rim to surfaces that have less sheen like cloth, and more rim light to surfaces such as metals. This also brings in another level of visual detail because the specular texture will have small details that differ from those in the base diffuse layer.

If we just use the pure rim mask for the rim light, the white color makes the character feel like it doesn't belong in the environment. We need the rim to have a color that matches the surroundings. If we just use a single, solid color to tint our mask, the result looks too flat and uniform.

Instead of having a uniform color for the rim light, our rim lighting uses the light gradient color. Rim light color on the light side can therefore be significantly different than the color on the dark side. This gives the rim a more painterly look.

Since rim lighting is light, it would be natural to think that it should blend with the other elements in the shader using addition. However, we found that when rim lighting was additive, it often causes the rim color to blow out to white instead of giving rich, saturated color.

Instead of adding the rim light, we calculate the rim mask, and then we do a linear interpolation between the pure light gradient color and the diffuse color. Using linear interpolation instead of addition is not physically accurate, but it ensures that the rim colors stay saturated instead of blowing out to white and it achieves our goal of matching the look of the concept art.

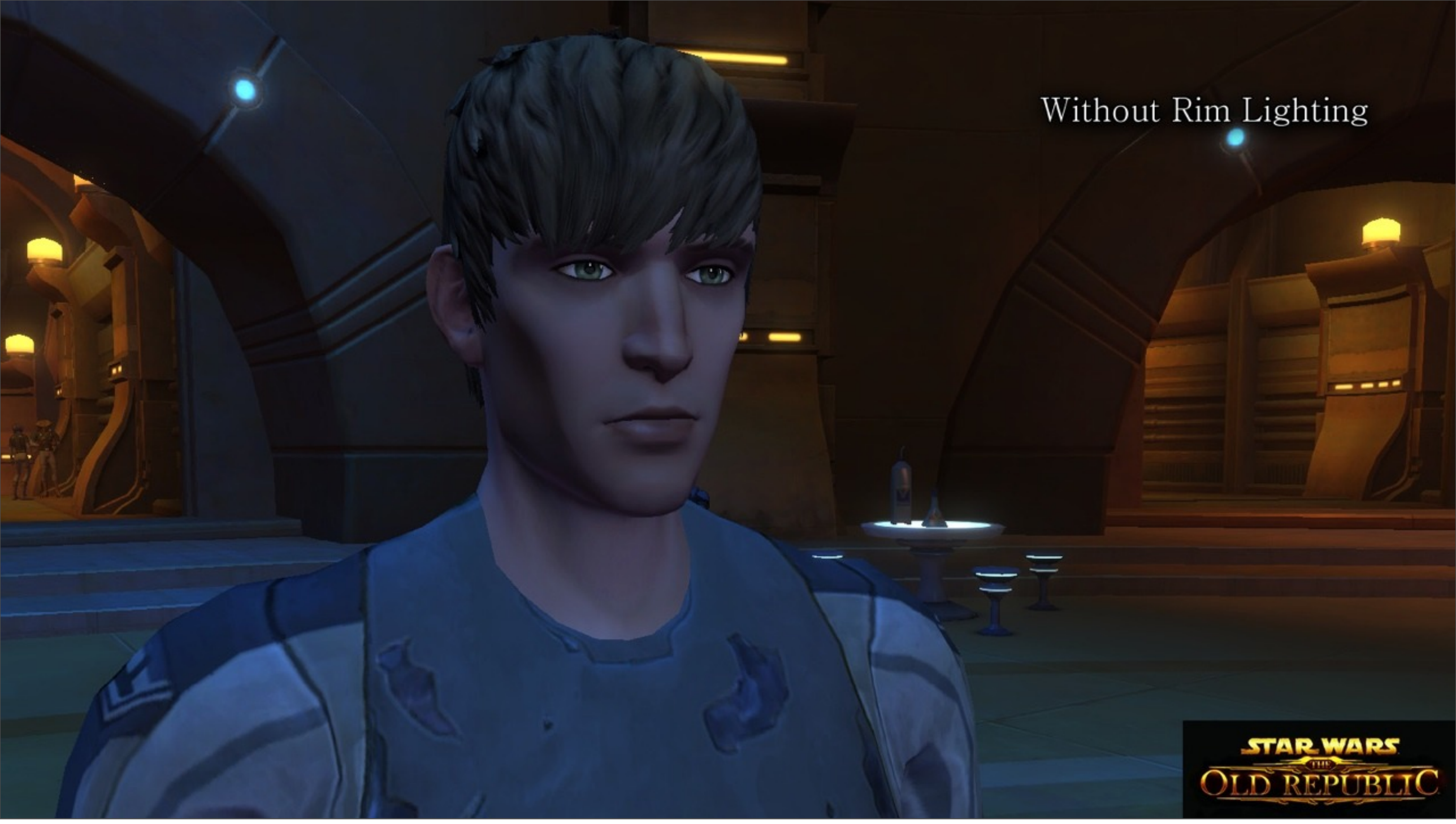
Notice how the rim lighting highlights the details in the character's outfit and helps us to see all of the complex shapes in his clothing.


Without Rim Lighting
STAR WARS
THE
OLD REPUBLIC


With Rim Lighting
STAR WARS
THE
OLD REPUBLIC

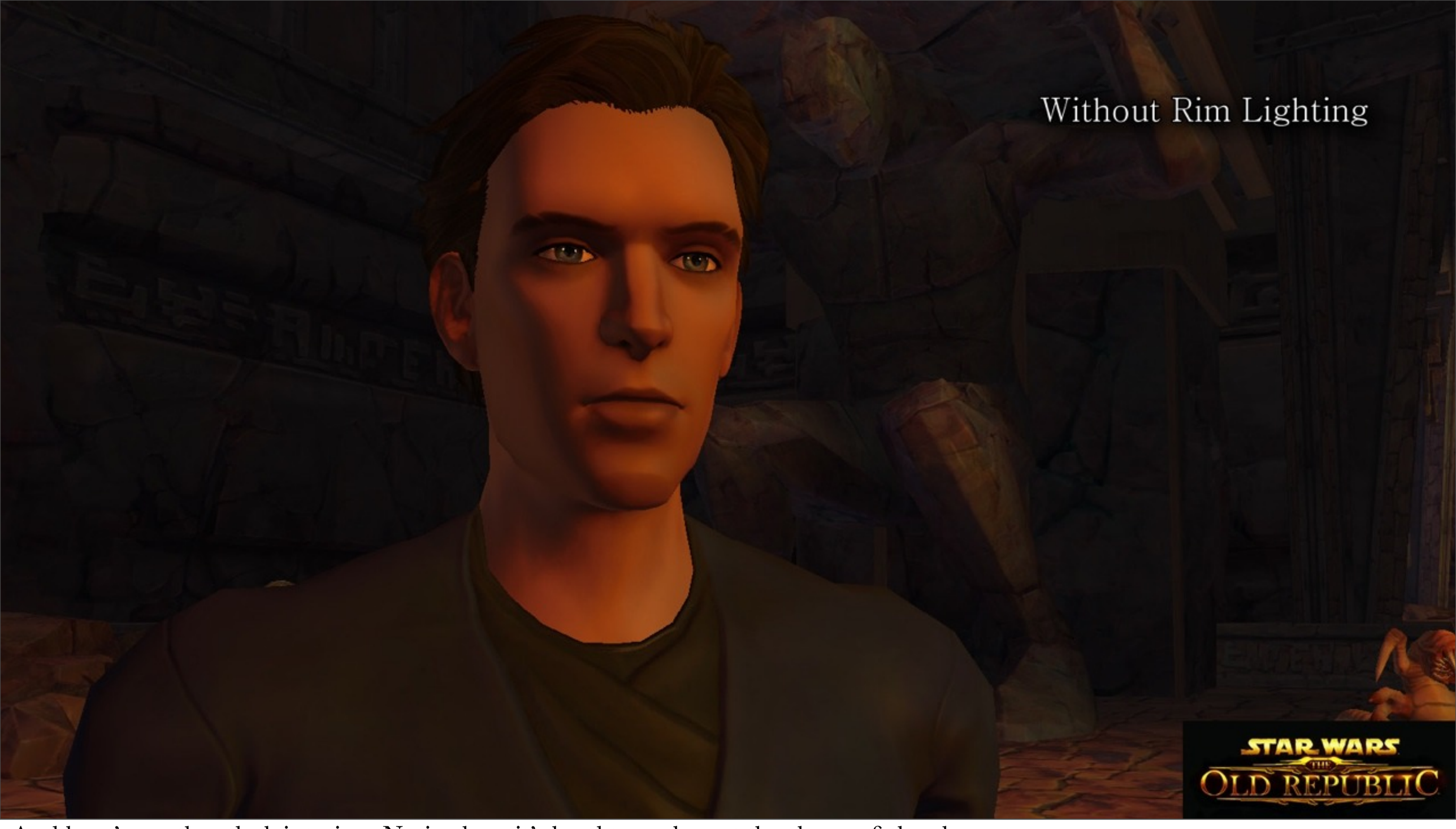
I this example, take a look at the hair on the left side when I turn on the rim lighting.

Did you see how that detail popped out? There's all that shape there that was completely lost is shadow, but the rim lighting pulls it out. That's what we're after.

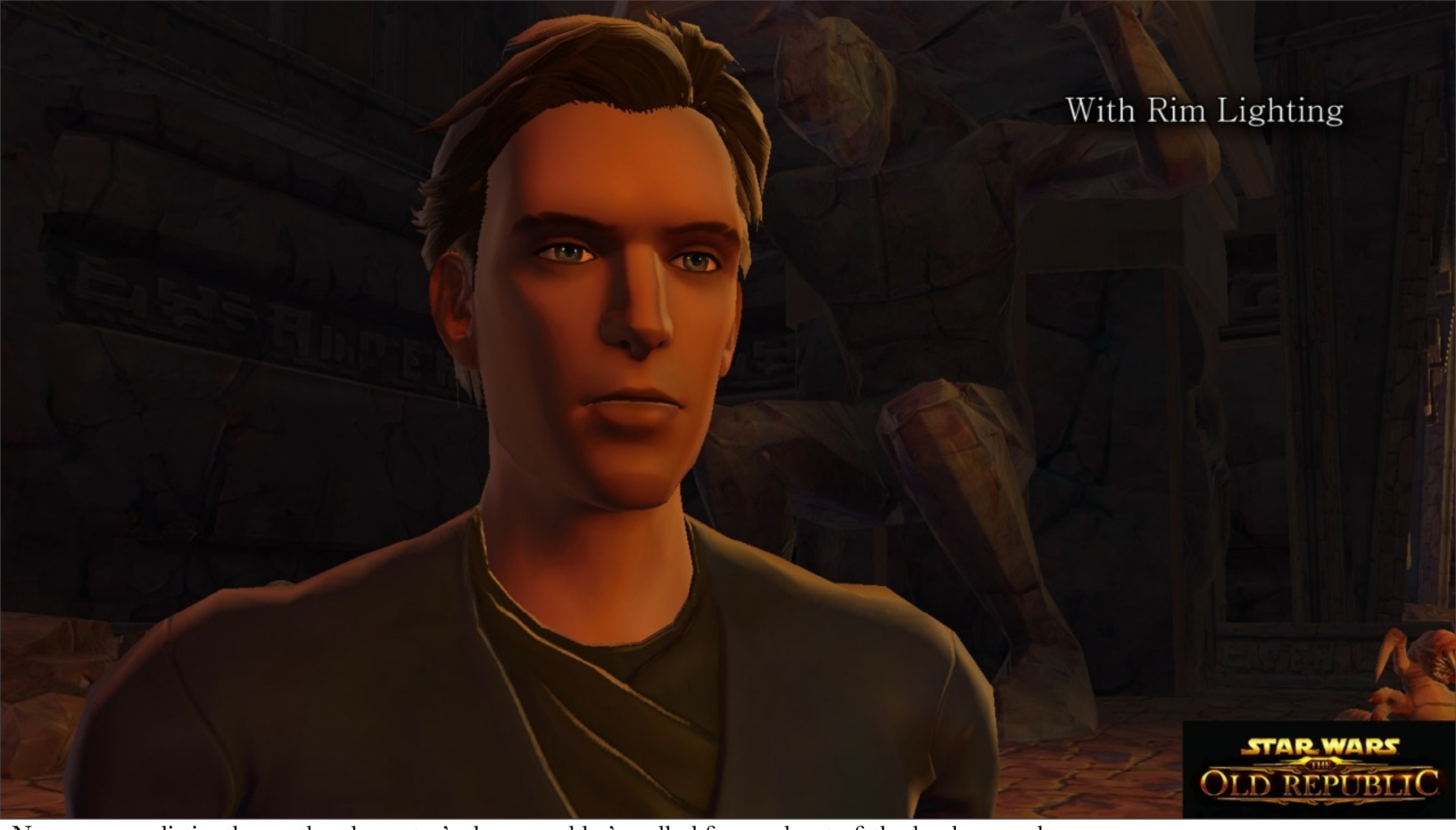


Here's an example in a darker interior environment. Here the character has a tendency to blend in with the background.

Adding the rim lighting pulls the character out from the background and really helps define his shape.



And here's another dark interior. Notice how it's hard to make out the shape of the character.



Now we can distinctly see the character's shape and he's pulled forward out of the background.

So - with rim lighting added, we've done a pretty good job of matching the appearance of our concept art. The rim light pops the characters out of the background, but since it also uses color from the light gradient, the rim light makes characters feel like they belong in the environment.

CONTROLLED LIGHT DIRECTION

Now let’s talk about Controlled Light Direction.

CONTROLLED LIGHT DIRECTION

- In film, light is set up to come from the best possible direction.
- The real direction of the light from the environment is completely ignored.
- We also want to put our game characters in the best possible light.
- It's too time consuming to place lights for each shot - or even for each cinematic.

Characters look best when light sources come from a specific range of directions. In film, lighting is set up to come from the best possible direction to light the characters.
Gaffers often completely ignore the direction that the actual natural light would be coming from and set up lighting to make the actors look their best and convey the mood of the shot.

We would also like to do this for our game so that we put our characters in the best possible light.

LIGHT DIRECTION IN GAME-PLAY

- Light direction is too steep for the characters.
- We clamp the angle of the main directional light to prevent steep angles on characters.

The main problem for our in-game lighting is that light angles are often too steep for our characters. The environment artists placed the main directional light source at a high angle to avoid long shadows in the environments, but steep light angles are not ideal for characters.

To solve this issue, we apply a clamping technique to the main directional light in the scene, while the point light directions remain unchanged. By clamping the light vector of the main directional light on the characters to a more aesthetically pleasing angle, we improve the lighting without making the character feel out of place in the environment. This technique allows us to achieve well-lit environments without sacrificing quality on the characters.

In these examples, it's easy to see that the lighting on the characters is pretty poor before our adjustments.

Once we clamp the light angle, we get a big improvement. It's almost as good as having hand-placed lights, and we get it without intense manual effort.


Before Light Angle Clamp
STAR WARS
THE
OLD REPUBLIC

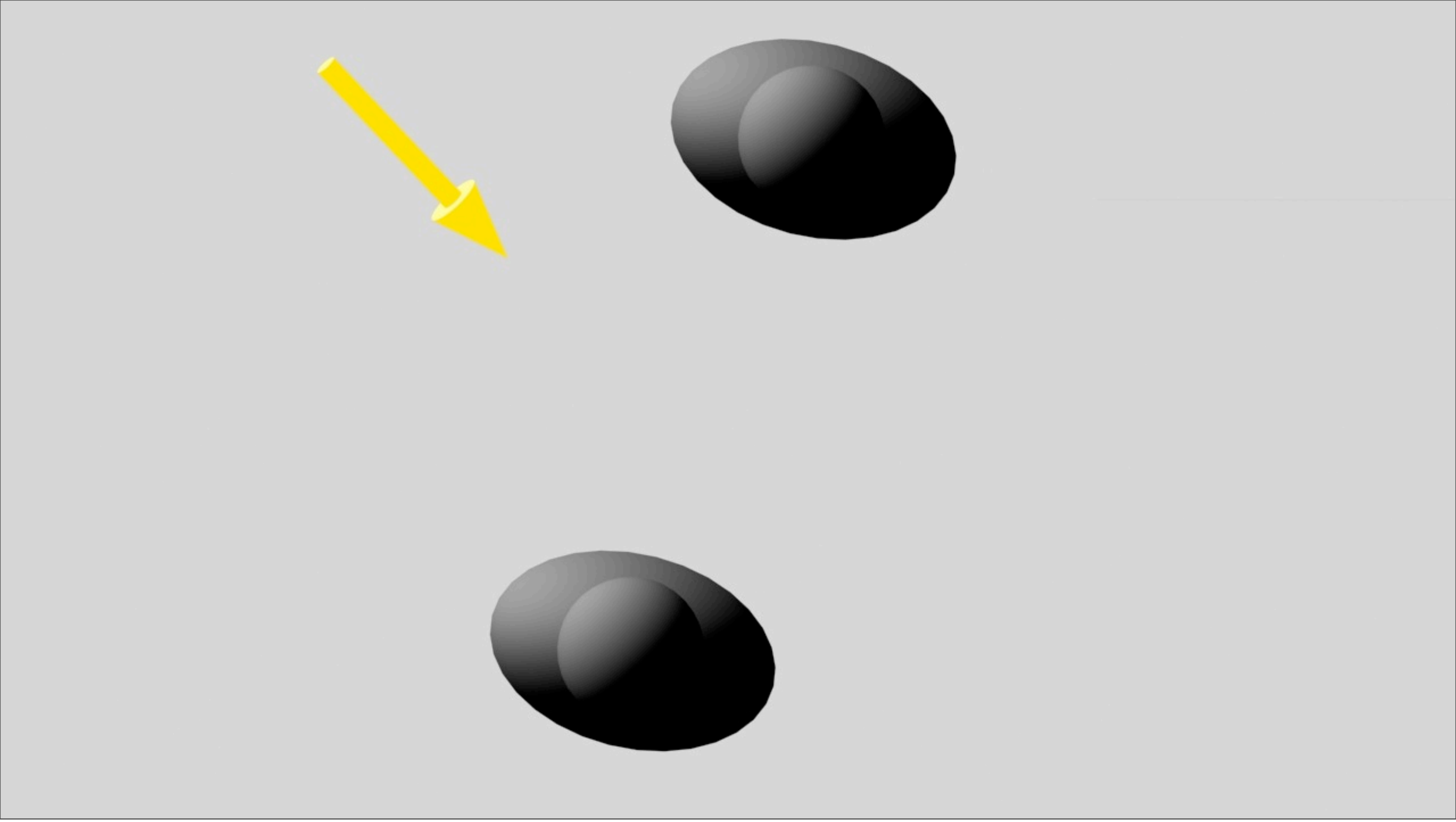

After Light Angle Clamp
STAR WARS
THE
OLD REPUBLIC

LIGHT DIRECTION IN CINEMATICS

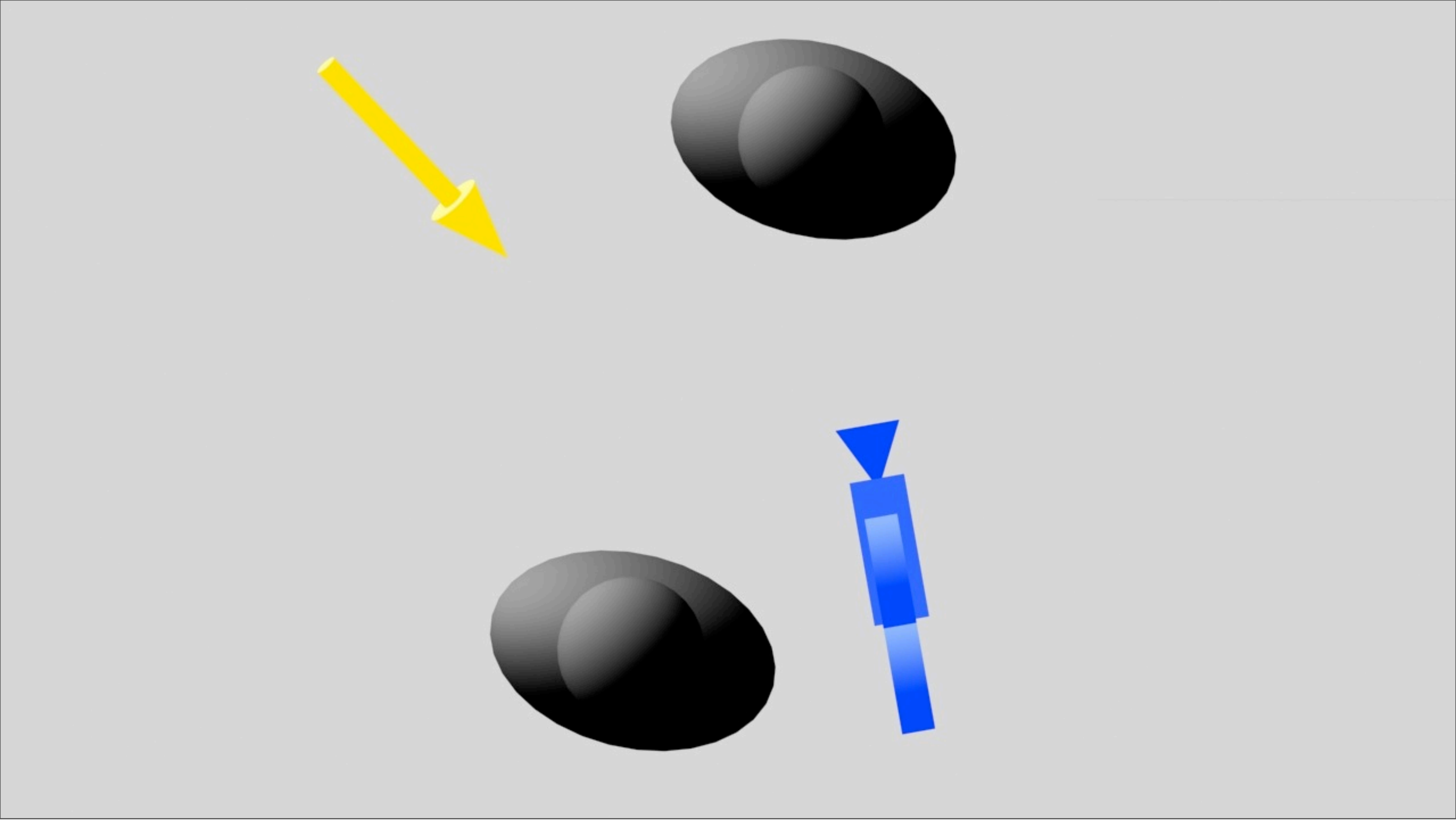
- Environment lighting was poor for most cinematic shots.
- Hand-placement of lights not possible because of massive volume.
- Cinematic light is forced to a camera-relative direction.
- We create a small set of pre-defined, camera relative lighting setups.
- Designers can select the best lighting setup for the shot.

For cinematics, the lighting already found in the environments is usually pretty poor for most of the shots. Characters are often lit from behind, or from other poor angles.
It was not possible for us to hand-place lights for each shot because of the massive volume of shots that needed lighting.
Instead, we decided to force all lights to come from a camera relative direction during cinematics, ignoring their actual positions and directions in the environment.
We have created a small set of these predefined camera relative lighting setups (including one that does no adjustment at all), and the cinematic designers can select the one that best fits the mood of the cinematic. For example, one lighting set up causes the light to come from 30 degrees over and 30 degrees up from the camera vector. Forcing the lights in the cinematics to be camera relative means that we always have a light source coming from a pleasing angle for the characters in every shot - without having to do any manual light adjustment.

So let's take a look at a specific conversation between the player and another character in the world. Here you can see from the shadow that the light is coming from this direction.

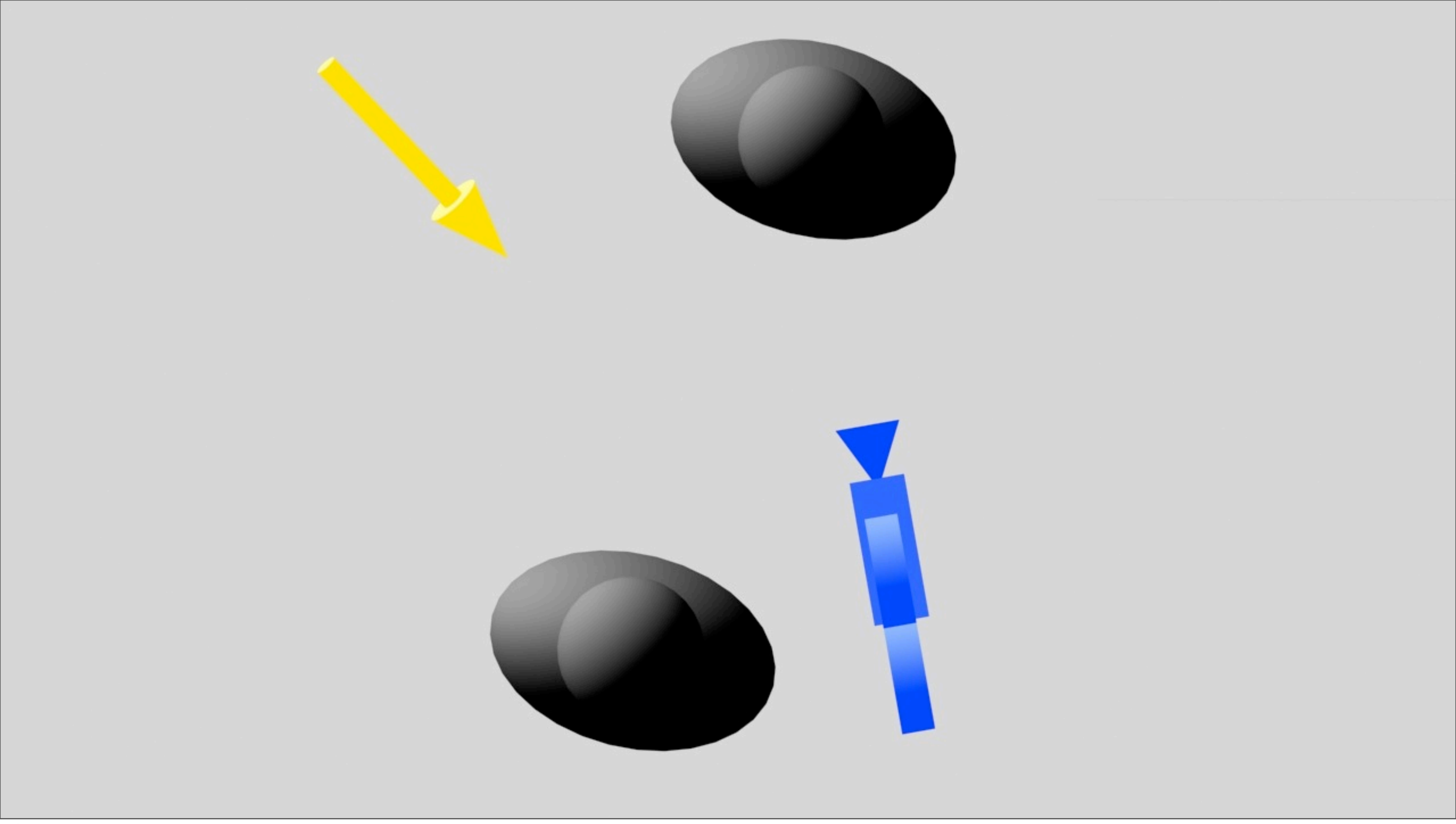


Here's an illustration that shows the problem. The direction of the light source isn't good for either of the characters.

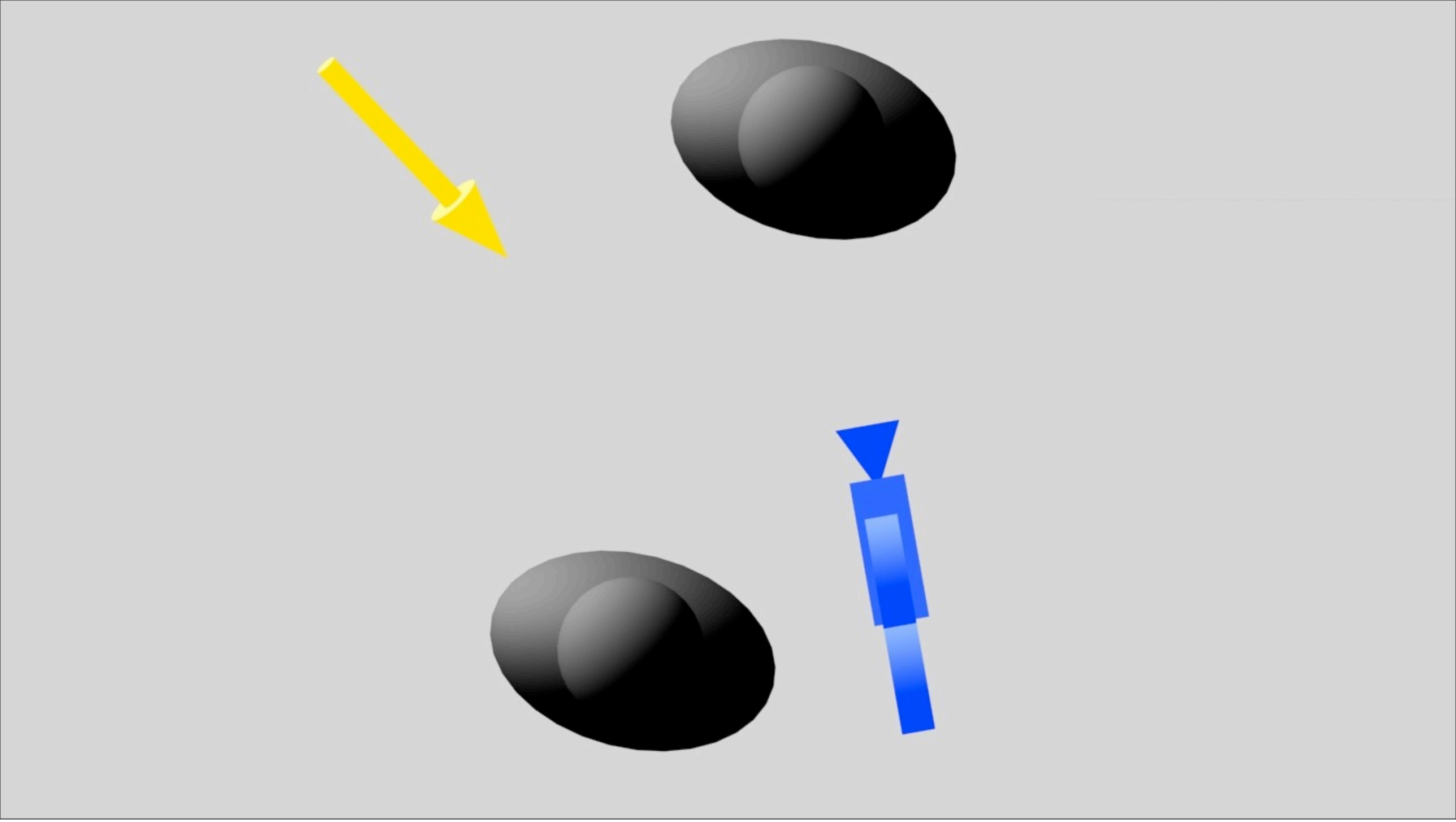


If we place the camera
here...

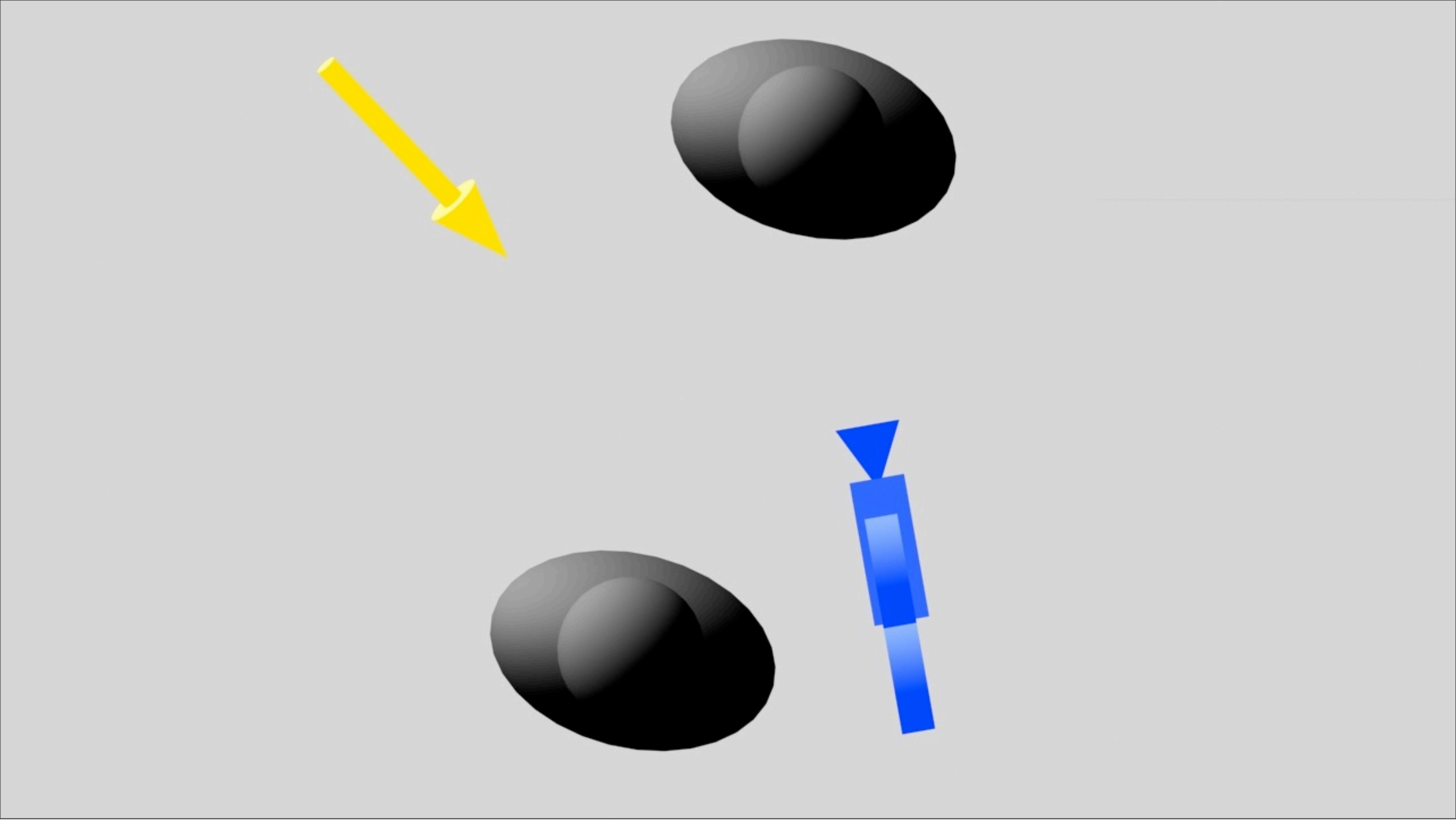
We end up with a shot that looks like this. The lighting doesn't work for the character.



So instead of just settling for the poor lighting we get from the environment, what we've done is to force the light on the characters to come from the direction of the camera.



And then we offset it’s rotation a little bit so that it’s not coming directly from the camera.



This makes the light on our characters look nice regardless of the direction of the light in the environment - because the character lighting is always camera-relative.

And we end up getting something that looks much nicer. When we apply this same camera-relative light adjustment to each shot, we end up with much better lighting overall.

Now let's take a look at what this looks like in a couple of sample cinematic shots. For every example in this video, notice that changing the light to be camera relative really improved the appearance of the character.

VIDEO EXAMPLE #1

Did you notice how much of a difference the simple light direction adjustment made? This adjustment is done automatically, and so we basically get really nice-looking lighting on the characters, without the artists having to do any extra lighting work at all.

CAMERA-RELATIVE LIGHT PROBLEM

- Light looks odd if it's always coming from the upper left.
- Our system determines, for each shot, whether the light should come from the left or right.
- We use the actual scene directional light to determine if right or left is a better match.

When dealing with camera relative lighting, one visual issue you run into is the feeling that the light is moving around from shot to shot. It is visually jarring to have the light always come from the upper left of every shot. It can make the light feel as though it is crossing the line of action in a conversation and flipping back and forth. To fix this visual oddity, our camera relative lighting will determine per camera cut if the light should be coming from the right or the left. The cinematic designer specifies the desired horizontal angle that the light should come from and the lighting system will determine if lighting from the right or the left best matches the main directional light in the scene. This solves the vast majority of the cases. In some cases the light may still be from the same direction per camera cut, but these only occur in lighting situations where the camera is facing into the light or away from the light. In such cases we found that the incorrect lighting angle is less jarring anyway because the background is not cueing the viewer into a clear lighting direction.

BioWare™
A DIVISION OF EA

SKY SPECULAR

SKY SPECULAR

- Lighting technique that helps characters stand out in dark environments.
- Approximates a specular highlight from an overcast sky.
- Replaces other ambient techniques.

Our gradient lighting already has some ambient included.

Overhead half lambert, or sky lights are common in games. These lights shine from above and their diffuse contribution wraps around the object. These have a tendency to flatten the appearance of the normals because the light is wrapping around the object)

SKY SPECULAR COMPUTATION

- Reflect the view vector about the normal.
- Compute dot product with world up vector.
- Renormalize dot product result from 0 to 1.
- Raise to a reduced specular power.
- We use (minSpecPower + specPower / 4).

SKY SPECULAR RESULTS

- Produces a broad specular highlight that is softer in appearance.
- Lighting is view dependent which conveys shapes well.
- Alters overall lighting less than other techniques.
- Introduces detail from the specular color map.

(view dependent - lighting changes as the viewer moves, rather than when the object moves. This conveys the shape of the objects better. It alters the lighting less because it highlights smaller details rather than broadly lighting the tops of objects.)

Here's an example of a character without sky specular. Notice how the shapes of the character pop when we add in the sky specular element.

Sky specular is especially helpful in dark interiors.

BioWare
A DIVISION OF EA

DEPTH OF FIELD

DEPTH OF FIELD

- Depth of Field (DOF) is important for making conversations feel cinematic.
- DOF directs the viewer's attention which aids in story telling.

1-Viewers expect to see depth of field in movies, without it, conversations in game do not have a cinematic feel
2- Artists can change focal depth, or produce a narrow focal range, to highlight the important elements in the scene.
**Depth of field is of course common in games these days, but we feel we have a slightly unique implementation worth mentioning

Here is a shot without depth of field. Notice how the busy background distracts from the character.

With DoF on, the background becomes less important, and the viewer focuses on the character.

Notice how the similar near and far ground colors make it difficult to focus on the character.

With depth-of-field, it's much easier to focus on the character.

DEPTH OF FIELD REQUIREMENTS

- Correctly handle any arrangement of objects in game.
- Correctly handle transparencies
  - Important for both foreground and background
  - Particles, holograms, Lightsabers, etc
- Runs on DX9 hardware.

In swtor we can enter a cinematic almost anywhere. We don't have the luxury of manipulating the contents of the background to fit the cinematic.

Therefore our DOF solution must handle any arrangement of objects in game including transparencies.

We need a depth of field implementation that is efficient and can run on a broad range of DX9 hardware.

All these requirements directed our technical solution for Depth of Field.

DOF IMPLEMENTATION DETAILS

- Implementation similar to the one described in GPU Gems 3“Practical Post-Process Depth of Field” - Earl Hammon, Jr.
- We made some changes to suit our performance and engine needs.
- We need a DOF implementation that automatically handles transparencies.

- We use an implementation similar to the one described in GPU Gems3

opted to ship without the additional passes that soften near geometry edges.

We output our circle of confusion (blur amount) to a single channel rather than output depth to a texture we can read back in.

DOF AND TRANSPARENCY

- DOF uses the z-depth to determine how much blur to apply.
  - But transparencies do not write to z-depth.
  - Multiple transparencies can contribute to pixel color from multiple z-depths.
  - Each of these color contributions require varying amounts of blur.

- Trans don't write to z-depth. We do not know how much to blur a transparent object by looking at its z-depth.

- This means that some of the color contributions should blur and others should not.

- we cannot accumulate depth like we do colors.

We cannot store the z-depths and colors separately. We cannot blur while rendering the transparencies. The blur must occur as a post processing step. At the time when post processing occurs, we have lost all the information we need for applying DOF to transparencies.

DOF INCORRECT SOLUTIONS

- Can't draw transparencies without blur.
  - This causes sharp edges to appear through the blur
- Can't allow z-depth to control blur.
  - This over blurs transparencies
  - And creates odd variation in DOF focus

To illustrate the issue more clearly, we will show some naive solutions.

Transparencies have low frequency detail, but not blurring them with DOF creates odd hard edges that show through the blur.

But notice how the near transparencies look as though they are correct.

Here we are blurring with Z-Depth.

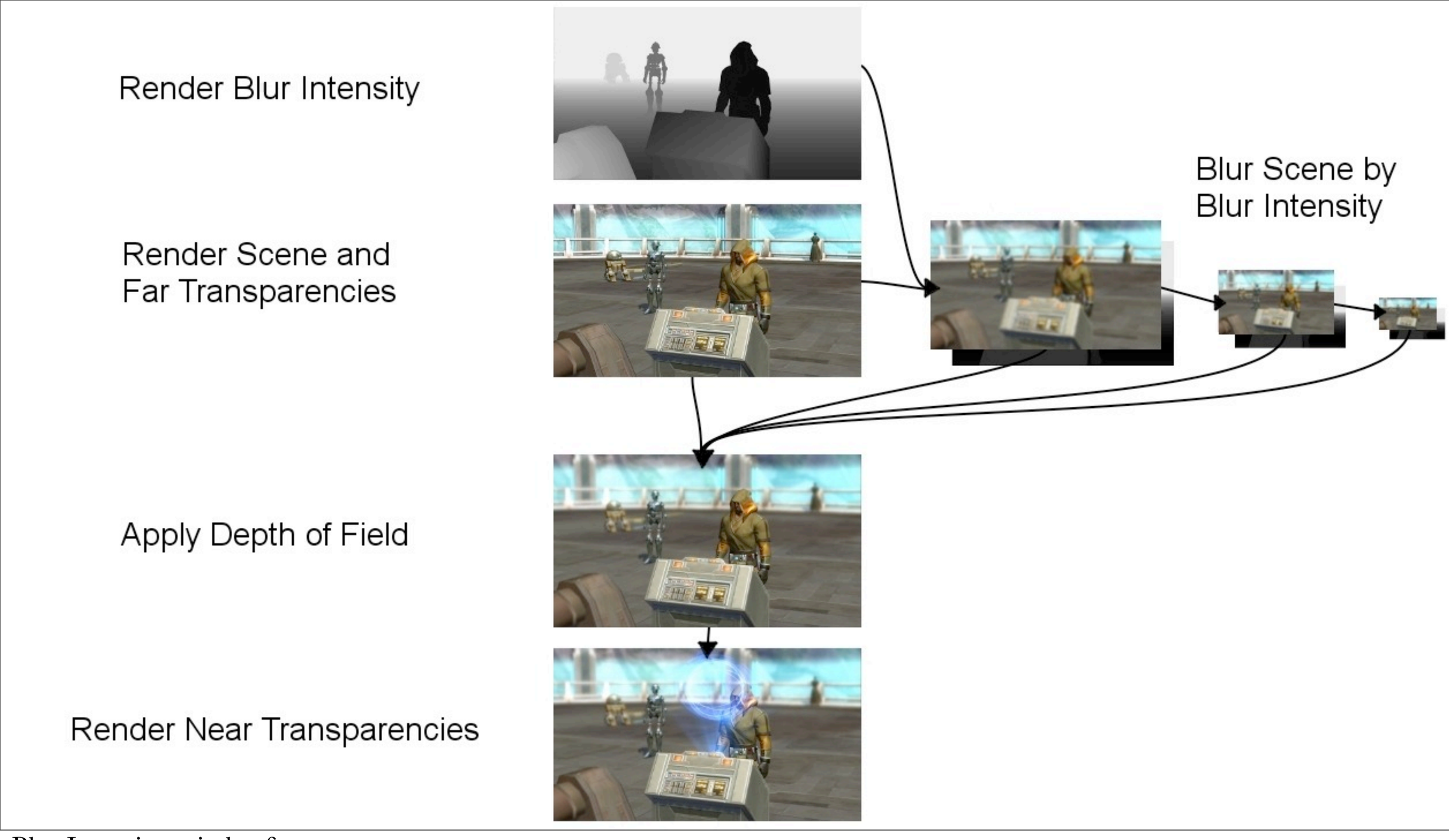
The near transparencies are too blurry when the background can be seen through them but they are in focus over the character.

Notice how the background transparencies look correct even though they are not being blurred with the exact correct z-depth.

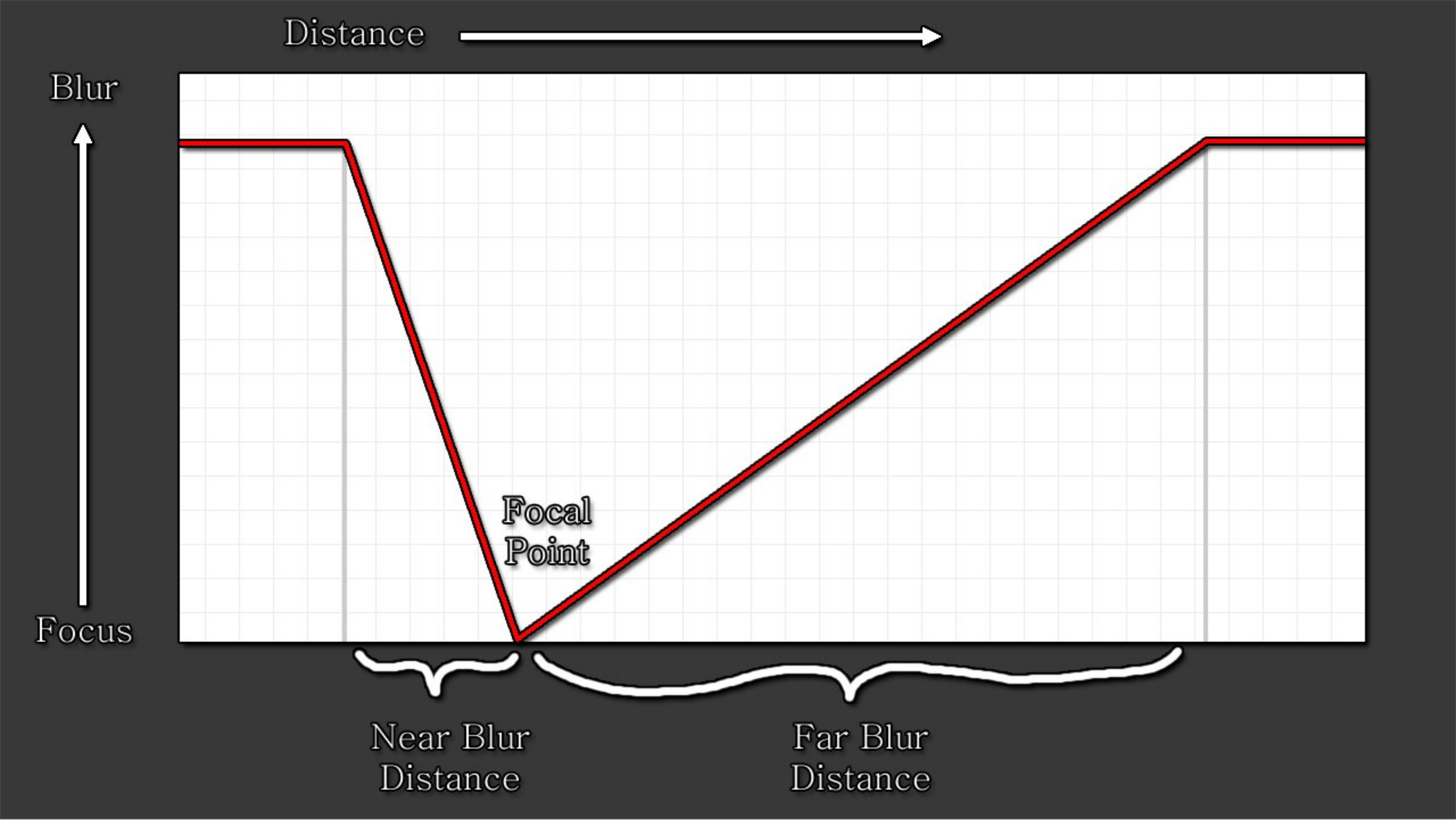
Here you can see what the scene should look like. The near transparencies are in focus and the far are blurred by the depth of field. Note that where the near and far transparencies overlap, the far color contributions are blurred and the near contributions remain in focus.

OUR DOF SOLUTION

- Allow distant transparencies to blur with z-depth.
- Draw near transparencies after DoF.
- Cross fade between near and far based on blend mode.



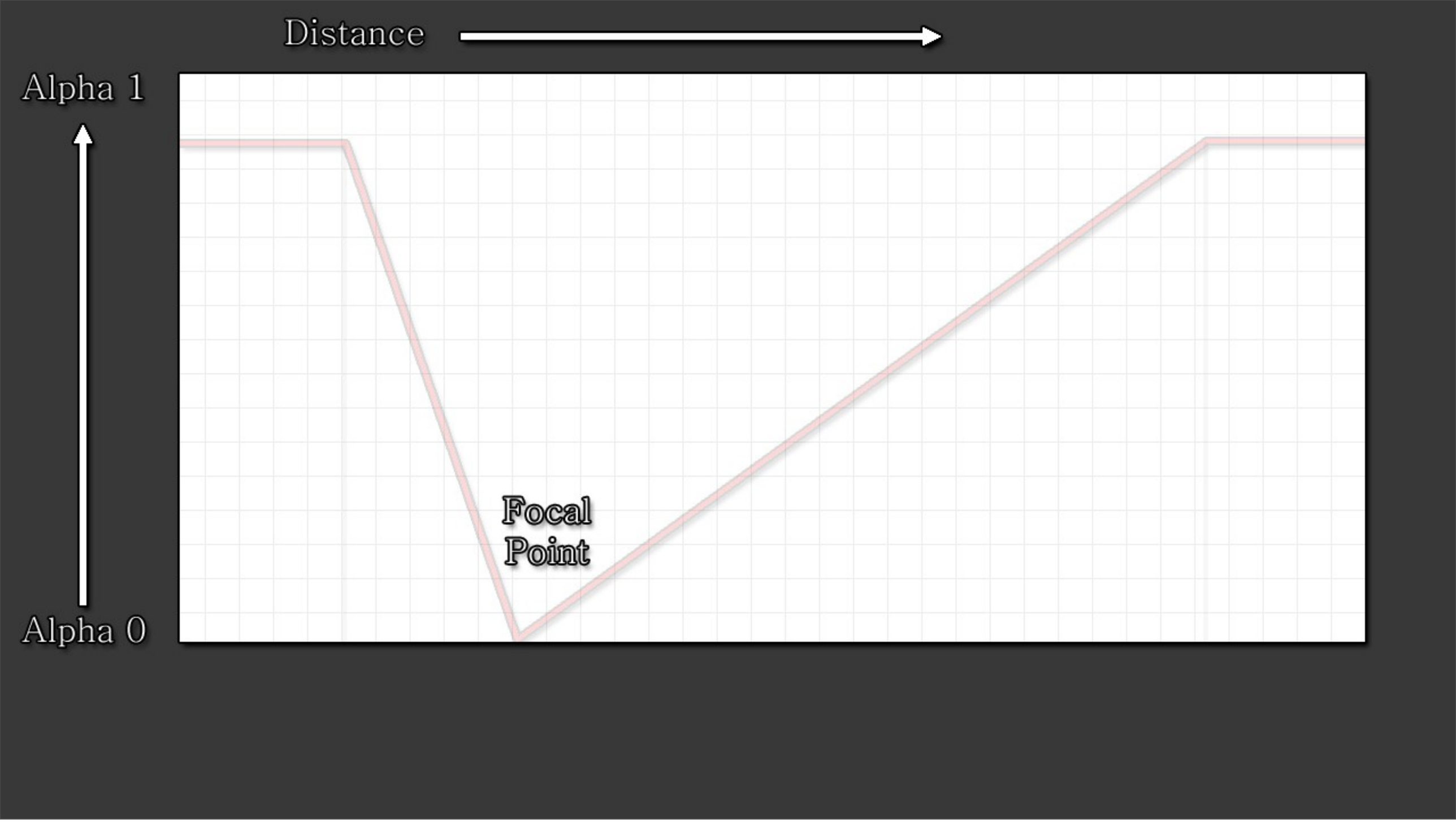
Blur Intensity - circle of confusion.



Graph illustrating blur intensity with distance from the camera. Y axis is blur, and X axis is distance from the camera.

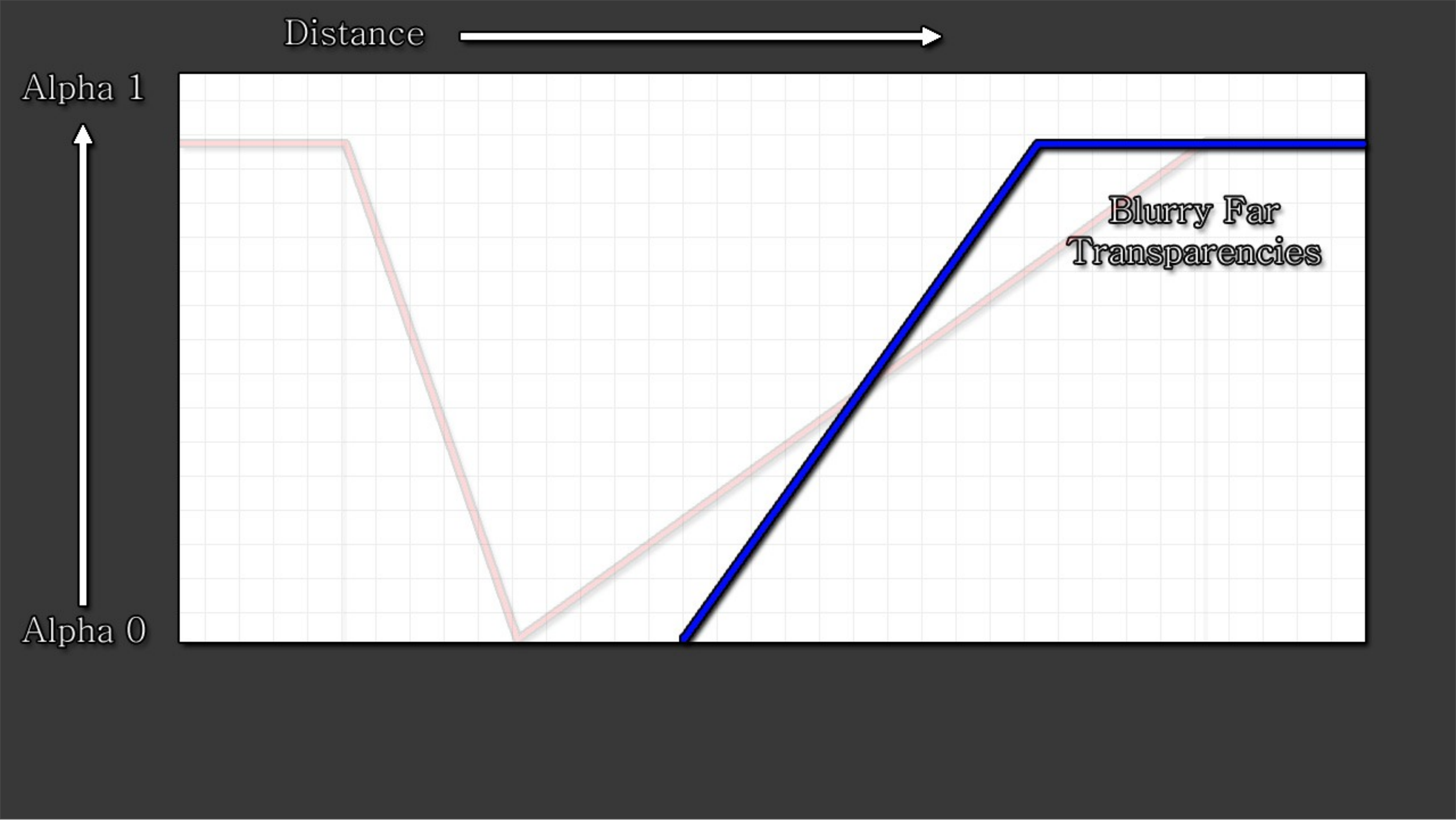
Artists control focal point as well as the near and far blur distances.

We want to cross fade near and far transparencies over these artist adjusted ranges.



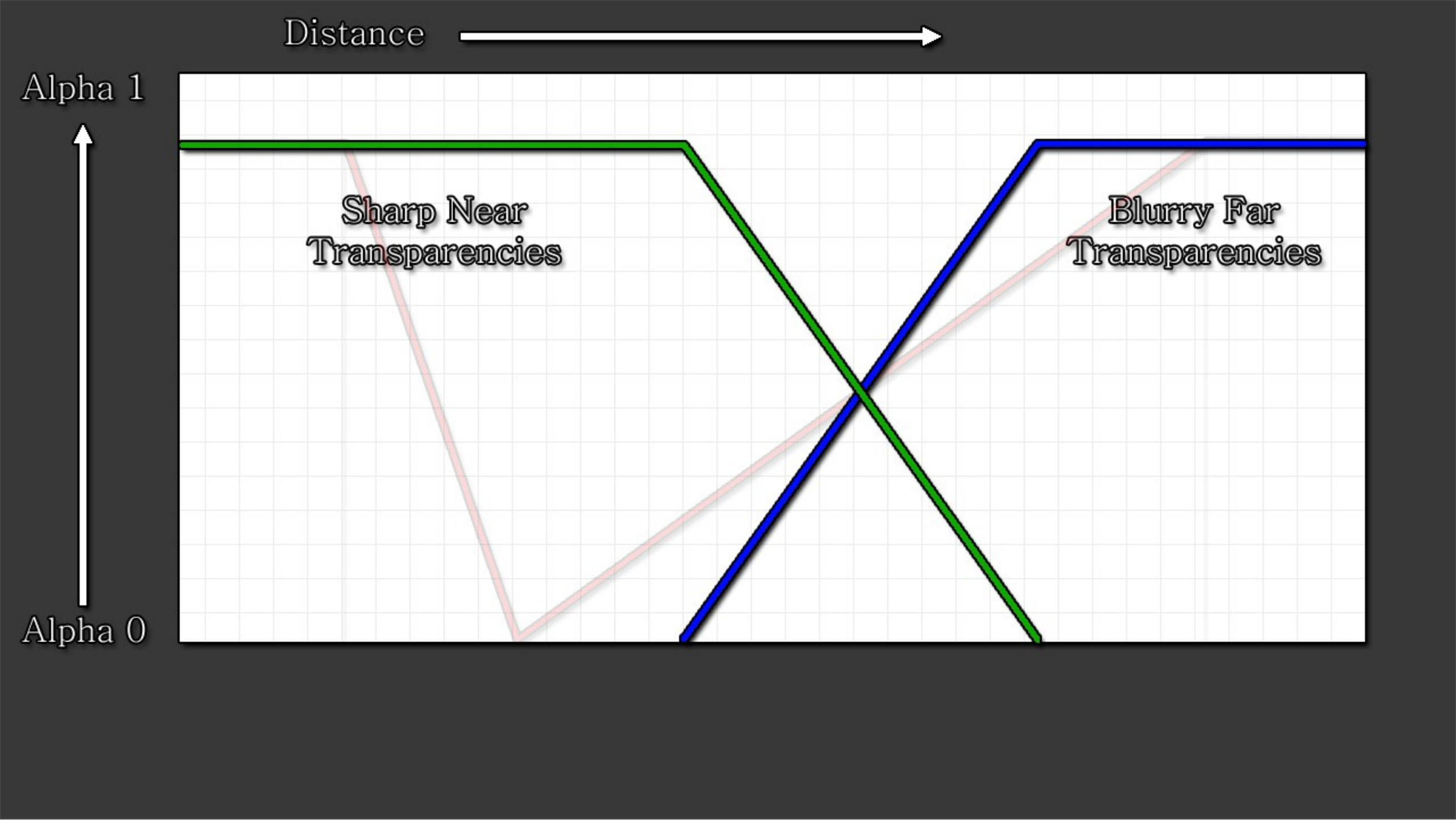
We are going to render transparencies over the scene and change their blending with distance.

Y axis has been changed to represent the alpha of the transparencies

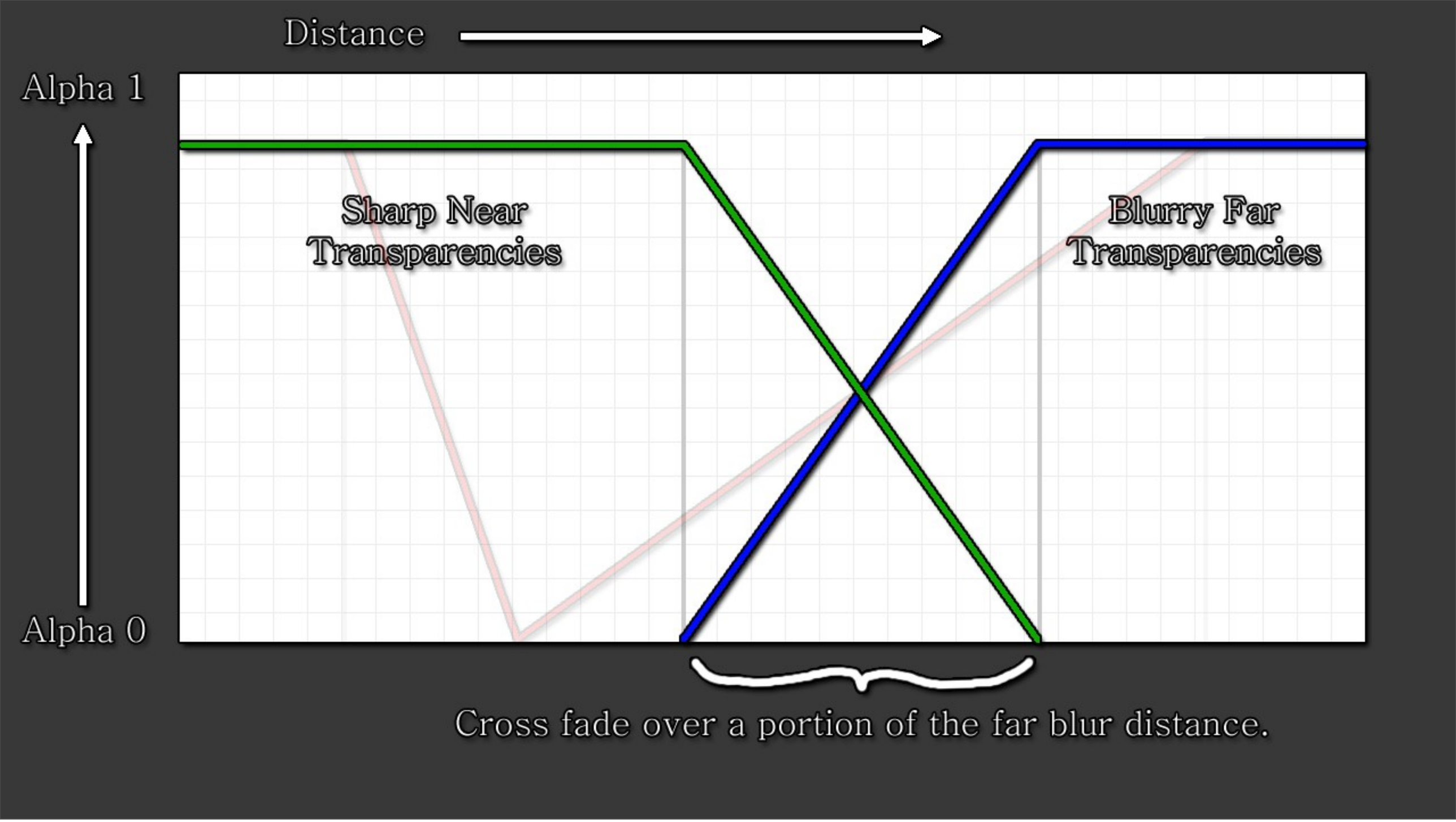


Fade the far transparencies in with distance.

We fade them in over a subrange of the far blur distance



We fade out the near transparencies over this same distance.



This creates a cross fade between near and far transparencies over a sub range of the far blur distance. The subrange we use is the range from 25%-65% of the far blur distance.

This produces an area of overlap where transparencies are drawn twice, but this is a reasonable expense.

Cross Fade differently based on blend mode

- Alpha blend - fade the alpha
- Additive blend - fade to black

BioWare
A DIVISION OF EA

VIDEO EXAMPLE #2

DOF TAKE-AWAY

- Effective technique for mixing transparency with DoF.
- Automatic transparency handling.

BioWare™
A DIVISION OF EA

PUTTING IT ALL TOGETHER

PUTTING IT ALL TOGETHER

- Achieves the Artistic Vision

Now that we have gone over each of the visual elements of our cinematic character lighting, lets go back to the original goals and challenges we faced and see how the added visual elements work together to solve them.

Our first goal was to achieve the artistic vision - to make our game look like our concept art.

Here are the character concept images we looked at originally. This time we have placed the actual in-game characters along side each of them.

Notice how the lighting on the in-game characters matches the style that the concept artists created. Instead of trying to create physically real looking lighting, we have achieved something much more stylized.

The light gradient and the rim lighting work together to create a look that closely resembles concept art.

PUTTING IT ALL TOGETHER

- Achieves the Artistic Vision
- Separates Characters From the Background

Our second goal was to separate our characters from the background.

Here are the original images we showed that demonstrate how our characters sometimes get lost in the background.

Adding rim lighting helps to define the edges of the character and separate him from the background.

And then adding in our depth-of-field effect directs the eye to the characters and helps push the environment into the background.

And here in this dark environment, it’s hard to see the shape of the character and distinguish him from the background.

Adding in our rim lighting really draws out the edges of the character and defines his shape.

And adding depth-of-field makes the background fall away and focuses our attention on the character.

PUTTING IT ALL TOGETHER

- Achieves the Artistic Vision
- Separates Characters From the Background
- Cinematic Quality Lighting Without Artist Effort

Finally, we wanted to achieve quality lighting in our cinematics, but without requiring our artists to hand place lights for each shot of every cinematic. Generally, well-lit characters have a key light, fill light, and a back light to bring out edge details and separate them from the background.

Take’s take a look at some example shots from our cinematics.

When we apply our adjusted, camera relative light direction, we get our key light.

Now let’s use our color gradient on the light instead of just a plain light.

With the color gradient, we get both the key light and the fill light from a single light source.

Next let's add our rim lighting.

Now we're getting key light, fill light, and back light all with one light source. Because this light source is camera relative, we get the same three point light set up for every shot of our cinematics.

Finally, we add in our sky specular for the final touch.

So with our additional lighting features, we go from this...


Corrected Light Direction
Light Color Gradient
Rim Lighting
Sky Specular
STAR WARS
THE
OLD REPUBLIC

...to this.

Here's another example. In this cantina environment, the character is too dark and gets lost in the background.

In this case, correcting the directional light doesn't do much, because we're in an indoor environment lit mostly by point lights - and we only apply the directional correction to out-door directional lights.

But our light color gradient will help . . .

See how using the light gradient gives us both key and fill light?

And the rim lighting really pulls the character out from the background.

And we round it out with our sky specular.

So we go from this . . .


Corrected Light Direction
Light Color Gradient
Rim Lighting
Sky Specular
STAR WARS
THE
OLD REPUBLIC

. . . to this.

And here's another example on Korriban.

Fix the light direction.

Use our light color gradient.

Add rim lighting

And sky specular.

So we start with this . . .

. . . and end up with this.

Here’s our dark tomb example.

Again - fixing the light direction does very little for interior environments.

But the color gradient really lights up the character.

And the rim light pulls him out of the background.

Then the sky specular gives him the extra boost of lighting that he needs.

So we go from this . . .


Corrected Light Direction
Light Color Gradient
Rim Lighting
Sky Specular
STAR WARS
THE
OLD REPUBLIC

. . . to this.

And our final example from Ord Mantel.

Fix the light direction.

Add in our light color gradient.

Pop the character out with rim lighting.

And boost the lighting with sky specular.

And our end result looks like . . .


Corrected Light Direction
Light Color Gradient
Rim Lighting
Sky Specular
STAR WARS
THE
OLD REPUBLIC

. . . This.

PUTTING IT ALL TOGETHER

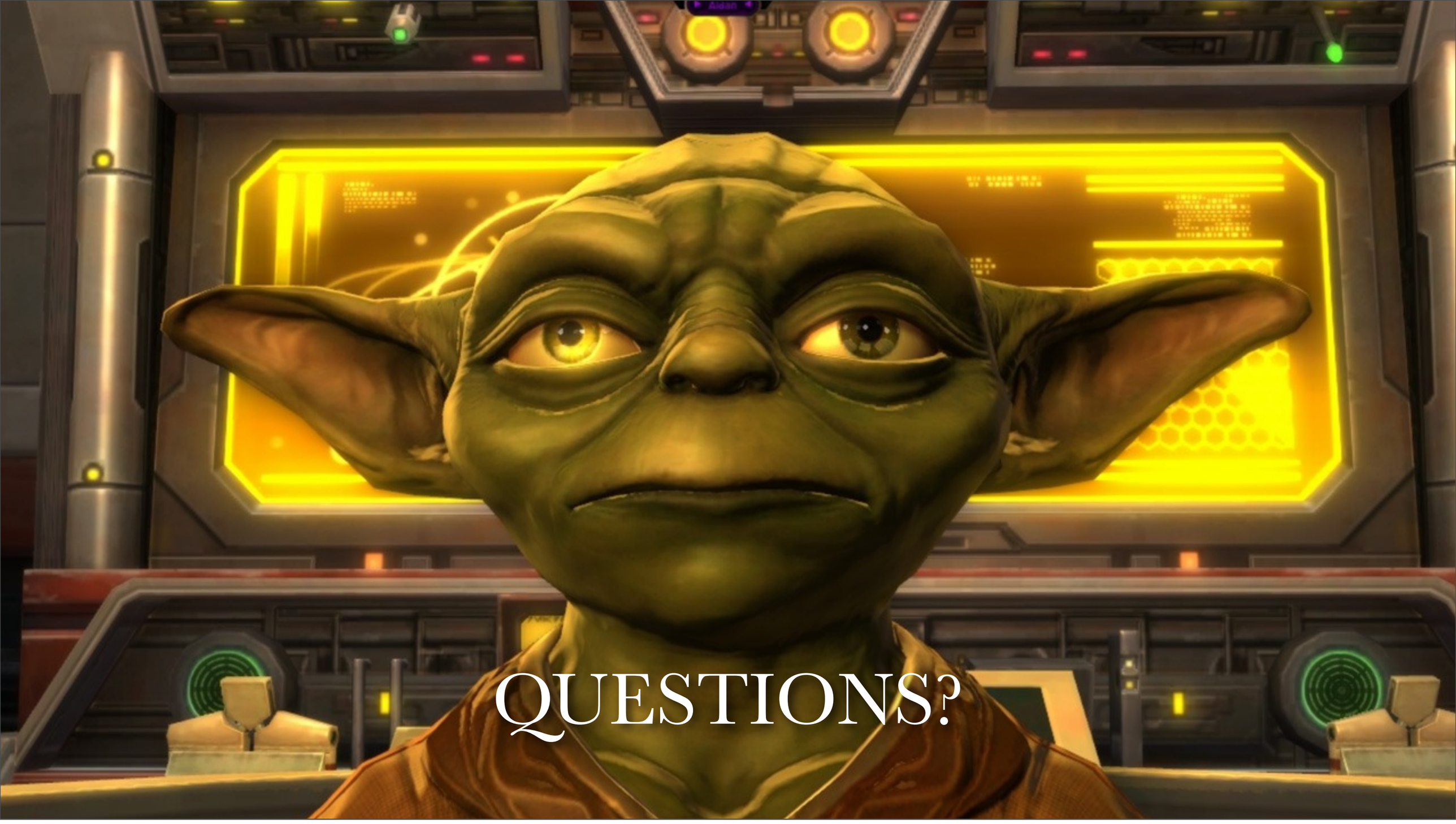
- Achieves the Artistic Vision
- Separates Characters From the Background
- Cinematic Quality Lighting Without Artist Effort

So I hope that you've been able to see how we've used our corrected light direction, light color gradient, rim lighting, sky specular, and depth of field our meet our three challenges - to match the visuals in our concept art, pop our characters out of the background, and achieve cinematic-quality lighting without requiring lots of manual effort.

THE IMPORTANCE OF COLLABORATION

- Who should write shaders? Artists or programmers?
- We've learned that it works best when technical artists and programmers work together.
- All of these techniques were developed through close collaboration.
- None of what we have presented would have been possible without teamwork.

There has been quite a lot of debate in the last few years about who should write shaders. Should it be artists, or should it be programmers? Through our experiences working on this project, we have learned that the shader writing process works best when technical artists and programmers work together. All of the techniques and shader features that we described in this talk were developed through a close collaboration between our technical artists and graphics programmers. Frequently, I would develop a prototype of an idea, and then share it with Aaron. He would refine the idea, make it really shine, and then develop the final implementation. Sometimes, Aaron would develop an idea and then come to me to ask for artistic input. I would make recommendations and tweak parameters to improve the idea. None of what we have presented would have been possible without this teamwork.


QUESTIONS?