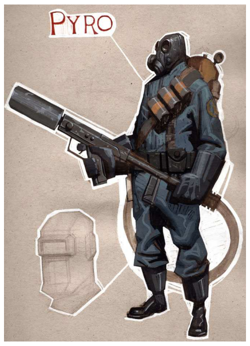
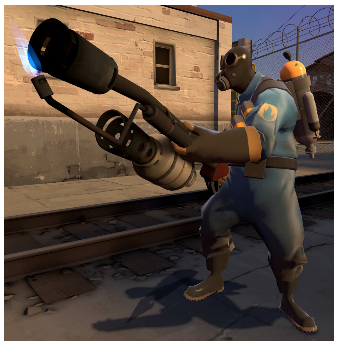
Illustrative Rendering in *Team Fortress 2*

Jason Mitchell[*] Valve
Moby Francke[†] Valve
Dhabih Eng[‡] Valve



(a) Concept art



(b) Character in the game

Figure 1: *(a) Concept Art (b) Character as seen by players during gameplay*

Abstract

We present a set of artistic choices and novel real-time shading techniques which support each other to enable the unique rendering style of the game *Team Fortress 2*. Grounded in the conventions of early 20th century commercial illustration, the look of *Team Fortress 2* is the result of tight collaboration between artists and engineers. In this paper, we will discuss the way in which the art direction and technology choices combine to support artistic goals and gameplay constraints. In addition to achieving a compelling style, the shading techniques are designed to quickly convey geometric information using rim highlights as well as variation in luminance and hue, so that game players are consistently able to visually "read" the scene and identify other players in a variety of lighting conditions.

CR Categories: I.3.0 [Computing Methodologies]: Computer Graphics—General; I.3.6 [Computing Methodologies]: Computer Graphics—Methodology and Techniques

Keywords: non-photorealistic rendering, lighting models, shading, hardware rendering, video games

1 Introduction

We present a set of artistic choices and real-time shading techniques which support each other to enable the unique Non-Photorealistic Rendering (NPR) style of *Team Fortress 2*. Grounded in the conventions of early 20th century commercial illustration with 1960s industrial design elements, the look of *Team Fortress 2* is the result of close collaboration between artists and engineers. At Valve, we believe in having the two disciplines heavily influence each other and, in this paper, we will discuss the ways in which art and technology choices combine to support stylistic goals and gameplay constraints. While most non-photorealistic rendering techniques are presented on a single rigid model in no particular lighting environment, we will demonstrate a series of interactive rendering techniques which result in a cohesive and dynamic environment. Rather than merely achieving a stylized look, the shading techniques are designed to quickly convey geometric information in our desired illustrative style using variation in luminance and hue, so that game players are consistently able to visually "read" the scene and identify other players in a variety of lighting conditions.

For *Team Fortress 2*, the 2007 sequel to the popular *Half-Life* mod *Team Fortress Classic*, we sought to explicitly differentiate ourselves from other multiplayer deathmatch games which typically embrace a modern photorealistic look. In *Team Fortress 2*, we chose to employ an art style inspired by the early to mid 20th century commercial illustrators J. C. Leyendecker, Dean Cornwell and Norman Rockwell [Schau 1974]. These artists were known for illustrating characters using strong, distinctive silhouettes with emphasis on clothing folds and they tended to use shading techniques which accentuated the internal shape of objects and characters with patterns of value while emphasizing silhouettes with rim highlights rather than dark outlines, as shown in the concept art in Figure 1a.

Contributions of this paper include the codification of key conventions of the commercial illustrations of Leyendecker, Cornwell and Rockwell as well as methods for generating such renderings in a real-time game. Specific technical contributions include the implementation of a diffuse light warping function appropriate for illustrative rendering, a novel formulation of rim lighting and an overall balance of photorealistic and non-photorealistic shading techniques to achieve the desired look while serving gameplay goals.

In the next section, we will discuss previous work which relates to ours. In Section 3, we will enumerate the specific properties common to commercial illustration which define our style. In Section 4, we will briefly discuss the creation of art for *Team Fortress 2*. In

[*]e-mail: jasonm@valvesoftware.com
[†]e-mail: moby@valvesoftware.com
[‡]e-mail: dhabih@valvesoftware.com

Section 5, we will discuss our shading algorithms in detail, before concluding with the topics of abstraction and future work.

2 Related Work

Non-photorealistic rendering styles can vary greatly, though they all ideally draw from some real-world artistic techniques under the assumption that such techniques developed by humans have inherent value due to the evolutionary nature of art. In the existing NPR literature, the commercial illustrative styles which inspired the look of *Team Fortress 2* are most closely related to the technical illustration techniques codified in [Gooch et al. 1998]. In Gooch shading, the traditional Phong model [Phong 1975] is modified using a cool-to-warm hue shift to indicate surface orientation relative to a given light source. As a result, extreme lights and darks are reserved for edge lines and highlights, resulting in a clearer perception of 3D object structure under difficult lighting situations than traditional computer graphics lighting models. In the world of *Team Fortress 2*, characters and other objects can be viewed under a wide variety of lighting conditions and thus we employ a similar system so that characters are clearly identifiable and aesthetically pleasing even in difficult lighting situations.

While the Gooch shading algorithm maps an unclamped Lambertian term to a warm-to-cool hue shift to improve shape perception, others have created a cel-shaded look by mapping this term to a very small set of colors with a hard transition at the terminator (where the Lambertian term crosses zero) [Decaudin 1996] [Lake et al. 2000] [Barla et al. 2006]. To achieve a cel-shaded look, Decaudin rendered objects with constant diffuse colors and relied upon shadow mapping to attenuate pixels facing away from a given light source. Lake does not rely upon shadow mapping but instead uses a 1D texture lookup based upon the Lambertian term to simulate the limited color palette cartoonists use for painting cels. Lake's technique also allows for the inclusion of a view-independent pseudo specular highlight by including a small number of bright texels at the "lit" end of the 1D texture map. Most recently, Barla has extended this technique by using a 2D texture lookup to incorporate view-dependent and level-of-detail effects. Barla also uses a Fresnel-like term to create a hard "virtual backlight" which is essentially a rim-lighting term, though this term is not designed to correspond to any particular lighting environment.

3 Commercial Illustration Techniques

In the work of the early 20th century commercial illustrators J. C. Leyendecker, Dean Cornwell and Norman Rockwell as well as our own internal concept art, we observed the following consistencies which we used to define the look of *Team Fortress 2*:

- Shading obeys a warm-to-cool hue shift. Shadows go to cool, not black.
- Saturation increases at the terminator with respect to a given light source. The terminator is often reddened.
- High frequency detail is omitted where possible
- On characters, interior details such as clothing folds are chosen to echo silhouette shapes
- Silhouettes are emphasized with rim highlights rather than dark outlines

With these fundamental principles in mind, we set out to create art assets (characters, environments and texture maps) and real-time shading algorithms that result in renderings with these properties. In the next section, we will discuss creation of art assets for *Team Fortress 2*, before moving on to the technical shading details in Section 5.

4 Creating Art Assets

In this section, we will discuss 3D character and world modeling as well as the principles we followed when generating texture maps necessary to meet both our gameplay and artistic goals.

4.1 Character Modeling

Players of multiplayer combat games such as *Team Fortress 2* must be able to visually identify other players very quickly at a variety of distances and viewpoints in order to assess the possible threat. In *Team Fortress 2* in particular, the player's class—Demo, Engineer, Heavy, Medic, Pyro, Spy, Sniper, Soldier or Scout—is extremely important to gameplay and hence the silhouettes of the nine classes were carefully designed to be very distinct from one another, as shown in Figure 2.

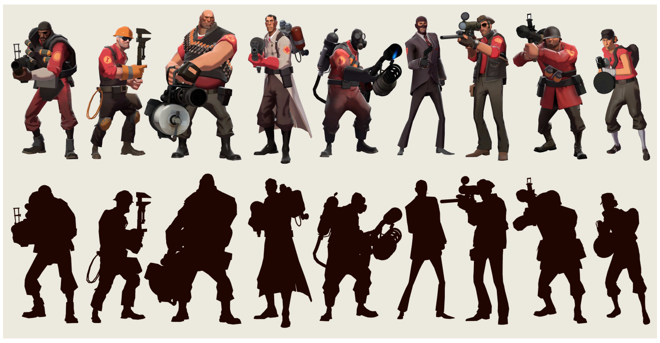


Figure 2: *The nine character classes of Team Fortress 2 were designed to be visually distinct from one another. Even when viewed only in silhouette with no internal shading at all, the characters are readily identifiable to players. While characters never appear in such unflattering lighting conditions in Team Fortress 2, demonstrating the ability to visually read the characters even with no internal detail was used to validate the character design during the concept phase of the game design.*

The body proportions, weapons and silhouette lines as determined by footwear, hats and clothing folds were explicitly designed to give each character a unique silhouette. In the shaded interior areas of a character, the clothing folds were explicitly designed to echo silhouette shapes in order to emphasize silhouettes, as observed in the commercial illustrations which inspired our designs. As we will discuss in Section 5, the shading algorithms used on these characters complement our modeling choices to enhance shape perception.

4.2 World Modeling

The unique look of the world of *Team Fortress 2* is borne out of well-defined design principles. For the architectural elements of the world associated with each of the two teams, blue and red, we defined specific contrasting properties. While the red team's base tends to use warm colors, natural materials and angular geometry, the blue team's base is composed of cool colors, industrial materials and orthogonal forms, as illustrated by the concept paintings of opposing building structures in the top row of Figure 3.

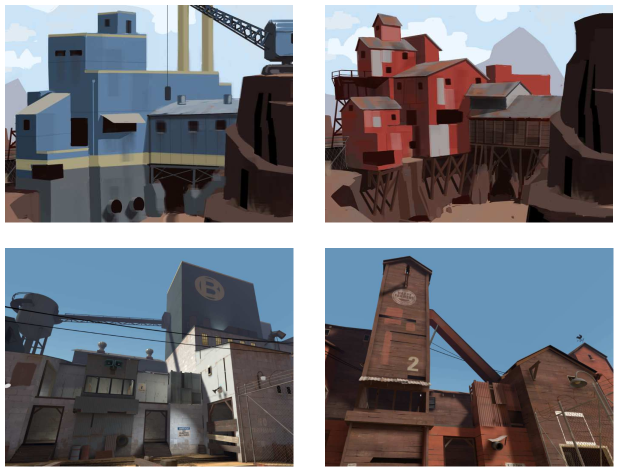


Figure 3: *World concept art for blue and red team bases (top) and in-game screenshots from Team Fortress 2 (bottom)*

Ultimately, the geometry of the game environments was modeled on these concept paintings, as shown in the bottom row of Figure 3. Though there is clearly more detail in the 3D modeled world than there is in the concept paintings, we still we deliberately avoided modeling the world in an overly complex or geometrically off-kilter manner as this would add an unnecessary level of visual noise—not to mention memory-hungry vertices—to the scene. We also found that keeping repetitive structures such as the bridge trusses, telephone poles or railroad ties to a minimum is preferable for our style, as conveying the impression of repetition in the space is more important than representing every detail explicitly.

By maintaining a minimal level of repetition and visual noise, we serve many of our gameplay goals while employing an almost impressionistic approach to modeling. This philosophy was also central to our texture painting style throughout the game.

4.3 Texture Painting

In *Team Fortress 2*, colors used on characters and the game world border on realism, but with increased saturation and value contrast. The blue and red teams in the game each have one base in a game level. The red and blue colors used to paint opposing bases are analogous to one another, as guided by the reference color swatch in Figure 4, with muted colors dominating and small areas of saturation to give further visual interest.

In addition to the dominant reds and blues, secondary and tertiary complimentary colors are added in smaller environmental props such as fire extinguishers and telephones. In general, the texture maps used on the 3D world are impressionistic, meaning that they are painterly and maintain a minimum level of visual noise. This is consistent with the style of painting used on background plates in many animated films, particularly those of Hayao Miyazaki, in which broad brush strokes appear in perspective, as if present in the 3D world rather than on the 2D image plane [Miyazaki 2002]. For our 3D game, we apply this same approach because we feel that its inherent frame-to-frame coherence has superior perceptual

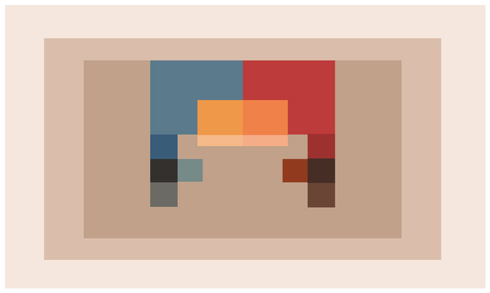


Figure 4: *Color scheme for the opposing blue and red teams*

properties to an image-space painterly approach. Of course, it also helps that portraying brush strokes on the surfaces of 3D objects in a game world rather than the 2D image plane is already supported in any 3D game engine by definition.

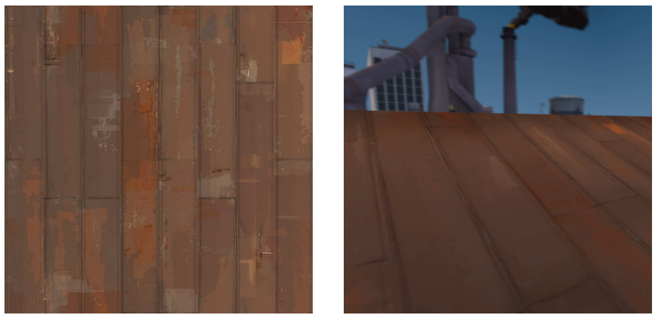


Figure 5: *World texture with loose, visible brush strokes*

Much of the world texture detail in *Team Fortress 2* comes from hand-painted albedo textures which intentionally contain loose details with visible brush strokes that are intended to portray the tactile quality of a given surface, as shown in Figure 5. In the early stages of development, many of these 2D textures were physically painted on canvas with watercolors and scanned to make texture maps. As we refined the art style of the game, texture artists shifted to using photorealistic reference images with a series of filters and digital brush strokes applied to achieve the desired look of a physically painted texture.

Not only does this hand-painted source material create an illustrative NPR style in rendered images, but we have found that these abstract texture designs hold up under magnification better than textures created from photo reference due to their more intentional design and lack of photo artifacts. Furthermore, we believe that high frequency geometric and texture detail found in photorealistic games can often overpower the ability of designers to compose game environments and emphasize gameplay features visually using intentional design choices such as changes in color value.

Now that we have discussed *Team Fortress 2* asset creation, we will move on to the unique aspects of our character and model shading algorithms which work together with our asset choices to achieve a unique illustrative style and enable players to easily identify other players in the scene.

5 Interactive Character and Model Shading

In this section, we will discuss the non-photorealistic shading algorithms used on the characters and other models in *Team Fortress 2* in order to achieve our desired illustrative style. For characters and most other models in our game worlds, we combine a variety of view independent and view dependent terms as shown in Figure 6.

The view independent terms consist of a spatially-varying directional ambient term plus modified Lambertian lighting terms. The view dependent terms are a combination of Phong highlights and customized rim lighting terms. All lighting terms are computed per pixel and most material properties, including normals, albedos, specular exponents and various masks are sampled from texture maps. In the following two sections, we will discuss how each of these lighting terms differs from conventional approaches and contributes to our aesthetic goals.

5.1 View Independent Lighting Terms

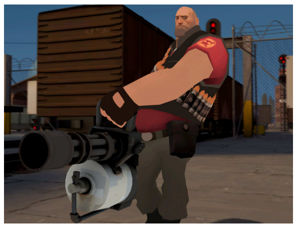
The view-independent lighting terms in *Team Fortress 2* consist of a summation of modified Lambertian terms and a spatially varying directional ambient term. The view-independent lighting terms can be summarized in Equation 1:

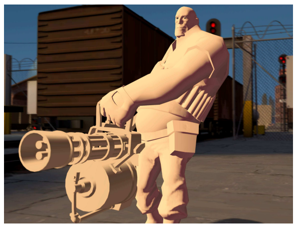
$$k_d \left[a(\hat{n}) + \sum_{i=1}^{L} c_i w\left(\left(\alpha \left(\hat{n} \cdot \hat{l}_i \right) + \beta \right)^{\gamma} \right) \right] \tag{1}$$

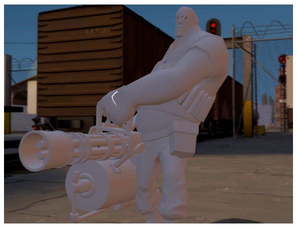
where L is the number of lights, i is the light index, c_i is the color of light i, k_d is the albedo of the object sampled from a texture map, $\hat{n} \cdot \hat{l}_i$ is a traditional unclamped Lambertian term with respect to light i, the scalar constants α, β and γ are a scale, bias and exponent applied to the Lambertian term, $a()$ is a function which evaluates a directional ambient term as a function of the per-pixel normal $\hat{n}$ and $w()$ is a warping function which maps a scalar in the range of 0..1 to an RGB color.

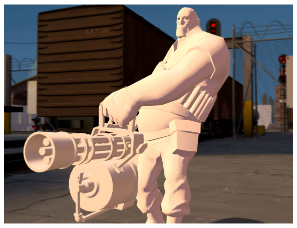
Half Lambert One unusual feature of Equation 1 is the scale, bias and exponentiation applied to $\hat{n} \cdot \hat{l}_i$. Since our first game *Half-Life*, which shipped in 1998, we have been applying a scale by 0.5, bias by 0.5 and square to diffuse lighting terms to prevent characters from losing a sense of shape on the back side with respect to a given light source ($\alpha = 0.5$, $\beta = 0.5$ and $\gamma = 2$). Even in our games which feature a more photorealistic look, we perform this operation so that the dot product which normally lies in the range of -1 to $+1$, instead lies in the range of 0 to 1 and has a pleasing falloff [Mitchell et al. 2006]. Due to the scale and bias of the Lambertian term by 0.5, we refer to this technique as "Half Lambert." This kind of scale and bias of the traditional Lambertian term appears in other NPR shading work including [Rusinkiewicz et al. 2006]. In *Team Fortress 2*, we leave α and β at 0.5 but set the exponent γ to 1 since we can express any shaping that we might want to get from the exponentiation in the warping function $w()$.

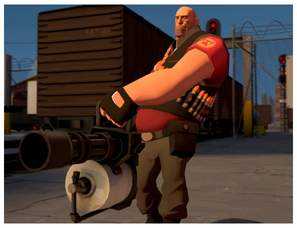
Diffuse Warping Function The second interesting feature of Equation 1 is the warping function $w()$ which is applied to the Half Lambert term. The goal of this function is to retain the shading information conveyed by the Half Lambert term while introducing the dramatic terminator observed in commercial illustration. In *Team Fortress 2*, this warping function is evaluated with a lookup into the artist-generated 1D texture shown in Figure 7. This is the same approach taken by [Lake et al. 2000] but instead of using this texture indirection to create a cartoon "hard shading" look, we preserve the variation in illumination for all normals while tightening the transition from light to dark around the terminator as shown in Figure 6b.

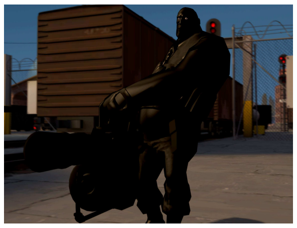

(a) Albedo

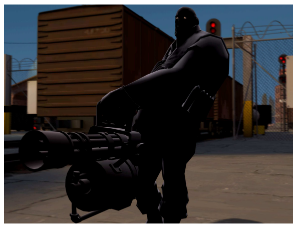

(b) Warped diffuse

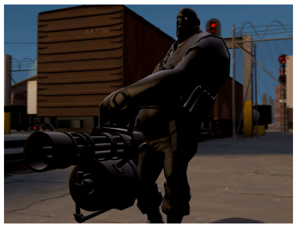

(c) Ambient cube


(d) (b) + (c)


(e) (a) * (d)


(f) Specular


(g) Rim lighting


(h) Specular + Rim Lighting

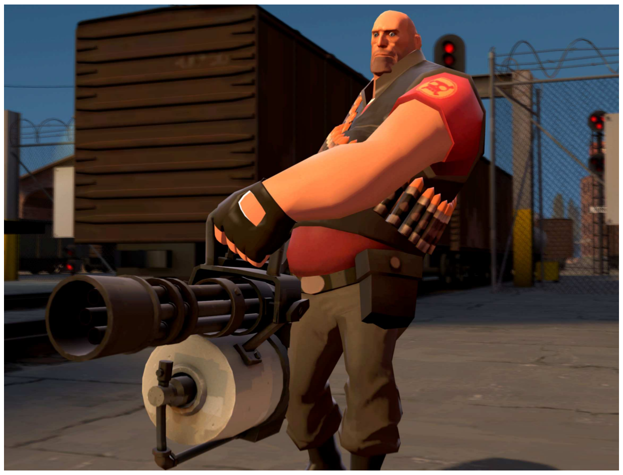

(j) Final result

Figure 6: *Individual character and model shading terms*

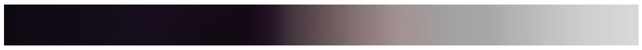


Figure 7: *Typical diffuse light warping function*

Besides the general "shaping" of the lighting described above, the 1D light warping texture in Figure 7 has a number of interesting features. First, the rightmost value in the texture is not white but is, rather, only slightly brighter than mid-gray. This is done because there is a multiplication by 2 in the pixel shader after the lookup from this texture map. This allows the artists to paint values into this 1D lookup texture which are up to two times "overbright," meaning that while the input to this function is a Half Lambert term in the 0..1 range, the output is in the 0..2 range. It is also important to note that this texture has essentially three regions: a grayscale gradient on the right, a cool gradient on the left and a small reddish terminator region in the middle. This is consistent with our observations that illustrated shadows often tend toward cool colors, not black, and that there is often a slight reddening at the terminator. We will discuss potential future extensions to this model such as tuning the warping function to suit the hue of the underlying albedo in Section 7. As shown in Equation 1, this warping function is applied to the scalar Half Lambert term for each of the diffuse light sources affecting an object, resulting in an RGB color which is subsequently modulated with c_i, the color of the light source, resulting in a diffuse lighting component as illustrated in Figure 6b.

Directional Ambient Term In addition to the simple summation of warped diffuse lighting terms, we also apply a directional ambient term, $a(\hat{n})$. Though the representation is different, our directional ambient term is equivalent to an irradiance environment map, as discussed in [Ramamoorthi and Hanrahan 2001]. Rather than a 9-term spherical harmonic basis, however, we use a novel 6-term basis which we call an "ambient cube," using cosine-squared lobes along positive and negative x, y and z axes [McTaggart 2004] [Mitchell et al. 2006]. These ambient cubes are precomputed by our offline radiosity solver and stored in an irradiance volume for fast access at run time [Greger et al. 1998]. Despite the simplicity of this lighting component, shown in isolation in Figure 6c, the ambient cube term contributes bounced light which is critical to truly grounding characters and other models in the game world. The summation of these view-independent lighting terms, shown in 6d, is multiplied with k_d, the albedo of the base material (6a), resulting in a diffusely lit character as shown in Figure 6e.

Now that we have discussed our modifications to traditional view-independent lighting algorithms, we will move on to the extensions we have made to typical approaches to view-dependent lighting.

5.2 View Dependent Lighting Terms

Our view dependent lighting terms consist of traditional Phong highlights combined with a set of customized rim lighting terms as summarized in Equation 2.

$$\sum_{i=1}^{L}\left[c_i k_s max\left(f_s\left(\hat{v}\cdot\hat{r}_i\right)^{k_{spec}}, f_r k_r\left(\hat{v}\cdot\hat{r}_i\right)^{k_{rim}}\right)\right]+\left(\hat{n}\cdot\hat{u}\right)f_r k_r a\left(\hat{v}\right) \quad (2)$$

where L is the number of lights, i is the light index, c_i is the color of light i, k_s is a specular mask painted into a texture channel, $\hat{v}$ is the view vector, f_s is an artist-tuned Fresnel term for general specular highlights, $\hat{r}_i$ is the reflection of the light vector from light i about $\hat{n}$, $\hat{u}$ is a world-space up vector, k_{spec} is the specular exponent fetched from a texture map, k_{rim} is a constant exponent which controls the breadth of the rim highlights, f_r is another Fresnel term used to mask rim highlights (typically just $(1-(\hat{n}\cdot\hat{v}))^4$), k_r is a rim mask texture used to attenuate the contribution of the rim terms on certain parts of a model and $a(\hat{v})$ is an evaluation of the ambient cube using a ray from the eye *through* the pixel being rendered.

Multiple Phong Terms The left side of Equation 2 contains a summation of Phong highlights calculated with the familiar expression $(\hat{v}\cdot\hat{r}_i)^{k_{spec}}$ which is modulated with appropriate constants and a Fresnel term. However, inside the summation, we also combine each Phong highlight using a $max()$ function with additional Phong lobes that use a different exponent k_{rim}, Fresnel term f_r and mask k_r. In *Team Fortress 2*, k_{rim} is constant for a whole object and significantly lower than k_{spec}, yielding broad rim highlights from the lights in the scene, independent of the object's material properties. These broad rim highlights are masked with a Fresnel term f_r, ensuring that they are only present at grazing angles (the very definition of a rim light). This combination of Phong highlights that match the material properties of a given object with broad rim highlights helps to give *Team Fortress 2* its signature illustrative look.

Dedicated Rim Lighting In situations where a character has moved away from the light sources in the game level, rim lighting based solely on Phong terms from local light sources may not be as prominent as we would like. For this reason, we also add in the dedicated rim lighting term shown on the right side of Equation 2. This term consists of an evaluation of the ambient cube using the vector from the eye through the point being shaded $a(\hat{v})$ modulated with an artist-painted mask texture k_r, Fresnel term f_r and the expression $\hat{n}\cdot\hat{u}$. This last expression is merely the per-pixel normal dotted with the up vector, clamped to be positive. This causes the dedicated rim lighting term to appear to add in indirect light from the environment, but only for upward facing normals. This is both an aesthetic choice and a perceptual decision designed exploit the human instinct to assume that lighting tends to come from above.

Unlike photorealistic games, which place a major emphasis on micro details for realism, we feel that our impressionistic approach favors the audience of fast-paced action games who typically play a game like *Team Fortress 2* many thousands of times and are more concerned with perceiving gross shape and shading as emphasized by our focus on distinctive silhouettes and rim lighting.

The complete pixel shader used on characters and other models in *Team Fortress 2* is merely the summation of Equations 1 and 2 in addition to some other operations such as optional environment mapping and a distance fog term, which we have left out for brevity. Since we evaluate fog directly in pixel shader code, we chose not to build it into the per-light texture indirection as proposed by [Barla et al. 2006].

6 Abstraction

Abstraction at distance is an important property of many NPR systems. For the most part, we rely on automatic mechanisms of abstraction such as depth fogging, minification of normal maps which tends to smooth out diffuse lighting, as well as SpecVar mapping to attenuate and broaden specular highlights at a distance [Conran 2005]. Since we do not apply image-space techniques such as outlining, we do not have to deal with issues of varying line weights, though our designers do intentionally simplify the shape and shading of our 3D skybox. In our graphics engine, the 3D skybox is a separate model which surrounds the game world but which is unreachable by players. This distant environment is not merely a painted backdrop on the inside of some simple geometry such as a large cube or sphere, but is a distant geometric model which provides parallax cues within itself and relative to the reachable game world. For *Team Fortress 2* in particular, 3D skybox geometry tends to be more painterly and is specifically modeled with less detail than it would be if it were in the interactive portions of the environment. This is not just to manage level of detail, but also fits with the overall visual style and prevents the skybox from generating high-frequency noise that would distract players.

7 Future Work

In future projects that call for illustrative rendering, we would like to further extend the model described above. For example, we have already experimented with extending our modification of the traditional Lambertian term to increase saturation of the particular hue of the albedo texture. To do this, we compute the hue of the albedo using shader operations and use the hue as the second coordinate into a 2D map which is an extension to the 1D map shown in Figure 7, where hue varies along the new axis. In practice, even a tightly optimized RGB-to-Hue conversion routine compiles to more than twenty pixel shader cycles, and we weren't willing to bear this expense on *Team Fortress 2* era hardware. In the future, we would like to be able to include this kind of extension to our diffuse model.

For our specular model, we have employed traditional Phong highlights in addition to our dedicated rim lighting. At the very least, we would like to extend this to allow for anisotropic materials such as cloth or brushed metals, using a combination of methods from [Gooch et al. 1998], [Heidrich and Seidel 1998] and [Gooch et al. 1999]. It might also be interesting to use stylized material-specific highlights by employing translation, rotation, splitting and squaring operations as discussed in [Anjyo and Hiramitsu 2003].

On parts of models which have relatively coarse tessellation, the slowly-varying tangent frames can result in overly-broad Fresnel terms, leading to rim highlights that are more obtrusive than we would like. In the future, we would like to look for methods to address this so that we can reliably generate more consistent and subtle rim highlights on polygonal models of varying mesh density.

Since our Source game engine has been previously used to develop a set of games such as the *Half-Life 2* series, *Counter-Strike: Source* and *Day of Defeat: Source*, which employ a more photorealistic rendering style, we have access to existing techniques for generating realistic effects such as motion blur, high dynamic range rendering, environment mapping as well as reflective and refractive water [McTaggart and Green 2006]. While rendering water with faithful photorealistic reflections of an otherwise NPR world is compelling, we would like to experiment with processing our reflection, refraction and environment maps with a filter such as an edge-preserving median filter to further stylize our water and other reflective effects.

Many traditional artists use image-space lightening and darkening techniques to increase contrast at important feature edges, as discussed in [Luft et al. 2006]. We believe it would be appropriate for our visual style to use scene depth information to integrate this type of technique into the look of *Team Fortress 2*.

References

ANJYO, K., AND HIRAMITSU, K. 2003. Stylized Highlights for Cartoon Rendering and Animation. *IEEE Comput. Graph. Appl. 23*, 4, 54–61.

BARLA, P., THOLLOT, J., AND MARKOSIAN, L. 2006. X-Toon: An Extended Toon Shader. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, ACM.

CONRAN, P. 2005. SpecVar Maps: Baking Bump Maps into Specular Response. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, ACM Press, New York, NY, USA, 22.

DECAUDIN, P. 1996. Cartoon Looking Rendering of 3D Scenes. Research Report 2919, INRIA, June.

GOOCH, A. A., GOOCH, B., SHIRLEY, P., AND COHEN, E. 1998. A Non-Photorealistic Lighting Model for Automatic Technical Illustration. ACM Press/ACM SIGGRAPH, New York, M. Cohen, Ed., 447–452.

GOOCH, B., SLOAN, P.-P. J., GOOCH, A. A., SHIRLEY, P., AND RIESENFELD, R. 1999. Interactive Technical Illustration. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, ACM Press, New York, J. Rossignac, J. Hodgins, and J. D. Foley, Eds., 31–38.

GREGER, G., SHIRLEY, P., HUBBARD, P. M., AND GREENBERG, D. P. 1998. The Irradiance Volume. *IEEE Computer Graphics and Applications 18*, 2 (/), 32–43.

HEIDRICH, W., AND SEIDEL, H. 1998. Efficient Rendering of Anisotropic Surfaces Using Computer Graphics Hardware. In *Image and Multi-dimensional Digital Signal Processing Workshop*, 315–318.

LAKE, A., MARSHALL, C., HARRIS, M., AND BLACKSTEIN, M. 2000. Stylized Rendering Techniques for Scalable Real-Time 3D Animation. ACM Press, New York, J.-D. Fekete and D. Salesin, Eds., 13–20.

LUFT, T., COLDITZ, C., AND DEUSSEN, O. 2006. Image Enhancement by Unsharp Masking the Depth Buffer. *ACM Transactions on Graphics 25*, 3 (jul), 1206–1213.

MCTAGGART, G., AND GREEN, C. 2006. High Dynamic Range Rendering in Valve's Source Engine. In *ACM SIGGRAPH 2006 Course Notes. Course on High-Dynamic-Range Imaging: Theory and Applications*. ACM Press/ACM SIGGRAPH.

MCTAGGART, G. 2004. Half-Life 2 Shading. In *Game Developers Conference. Direct3D Tutorial*.

MITCHELL, J. L., MCTAGGART, G., AND GREEN, C. 2006. Shading in Valve's Source Engine. In *SIGGRAPH Course on Advanced Real-Time Rendering in 3D Graphics and Games*.

MIYAZAKI, H. 2002. *The Art of Spirited Away*. VIZ Media.

PHONG, B. T. 1975. Illumination for Computer Generated Pictures. *Commun. ACM 18*, 6, 311–317.

RAMAMOORTHI, R., AND HANRAHAN, P. 2001. An Efficient Representation for Irradiance Environment Maps. In *SIGGRAPH 2001: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 497–500.

RUSINKIEWICZ, S., BURNS, M., AND DECARLO, D. 2006. Exaggerated Shading for Depicting Shape and Detail. *SIGGRAPH 2006 25*, 3 (July), 1199–1205.

SCHAU, M. 1974. *J. C. Leyendecker*. Watson-Guptill.