

STUDY ON APPLYING BRIGHTNESS FLUOROSCOPY ON STRENGTHENING ART APPEARANCE FOR 3D GAME

YUNG-PIAO CHIU^{1,2,*}, HUNG SUN¹ AND KAI-TUNG LIU³

¹Department of Digital Media Design
Hwa Hsia University of Technology
No. 111, Gongzhuan Road, Zhonghe District, New Taipei City 23568, Taiwan
*Corresponding author: frank@cc.hwh.edu.tw; arionsun@cc.hwh.edu.tw

²Department of Technology Management

³Department of Construction Management

Chung Hua University
No. 707, Wufu Road, Sec. 2, Hsinchu 30012, Taiwan
M10116010@chu.edu.tw

Received November 2015; accepted February 2016

ABSTRACT. *Taiwanese students prefer using expression styles involving cute cartoon renderings when designing games. This is a universal game art style preference across Asia. 3D game graphics by using preset cartoon shader in Unity 3D engine will look “flattening”. Thus, the game image loses its depth perception. In this study, we referenced the evolution of perspective and spatial relationships in Western painting. According to changes in the principles of atmospheric perspective, the lightness perspective effect associated with visually highlighting screen subjects was used to design a shader that detects the Y-axis values of an object’s coordinate system to fill in the gradient hues of vertex colors. A simulated lighting effect can be achieved by changing the shader of a 3D object in a game, substantially reducing the map handling workload and improving the sense of 3D segmentation of the characters and environments in the game image.*

Keywords: Game art, Chromatics, Lightness perspective, Shader

1. Introduction. The use of cartoon-style 3D game, tends to have a visual flattening, inadequate depth of the problem. First, we used a 3ds Max plugin to confirm the image optimization effect. To reduce the workload associated with using shading maps to generate numerous game objects, a shader was developed for use in the Unity engine. This is a simple method of improving the adjustment space in the game image and enhancing the effectiveness of the game’s execution. Furthermore, in addition to determining the differences in the world coordinates defined in the software program used by the game developers, the proposed shader distinguishes between two specialized forms, Z-up and Y-up, to satisfy the requirements of any development environment. In this study, we applied a traditional lightness perspective painting technique to game characters and objects rendered in a cartoon style. Shades were added to the exteriors of the characters and objects in the game to ensure a highly detailed depth perception between objects, highlighting the differences between the characters and the scene and enhancing the quality of the game images.

2. Problem Statement and Preliminaries.

2.1. Motivation. Digital games are a form of comprehensive artistic expression that integrates computer, Internet, animation, art, interaction, and music software, hardware, and media. Large-scale game titles have original and unique worldviews, engaging artistic images, and storylines with various twists. Small games are also fun and captivating.

Pleasant gaming experiences can be ensured by integrating visual image effects and game-play to immerse gamers. Art plays an irreplaceable role in shaping and producing game images. It shoulders the key task of visualizing the game world [1].

Because of most students' preference for Japanese animation [2] and because of the limitations of three-dimensional (3D) rendering techniques, students typically adopt cute cartoon styles with a cell shaded sense in artistic expressions when creating graduation projects (Figure 1). Such student works are often highly saturated and bright with uniform lighting in their visual expressions. Thus, the degree of separation between the characters and the scene is insufficient. The resulting images lack a sense of spatial depth.



FIGURE 1. 3D game involving the use of toon shaders



FIGURE 2. Initial image effects (left) and grayscale examination (right)

2.2. Toon shader. When Unity is applied to developing game projects, the built-in Toon-Basic shader [3] is typically used when a cartoon rendering style is required. The lighting of the scene does not affect this simple cartoon texture shader. It relies on the light and shade of the drawn model maps to create a sense of 3D depth. Figure 2 shows an initial image created using the built-in Toon-Basic shader. Although a hand-painted texture style has been applied to drawing the scenery of this fairy-tale game [4,5], the color scale observed after converting the image into grayscale by using Photoshop shows that limited color levels are used in this image. The two ends of the light and dark regions have no data. Thus, this image has virtually no field depth because each 3D object appears to be attached to the image like a cutout.

2.3. Atmospheric perspective. During the Renaissance, artist Leonardo da Vinci [6] developed the theory of atmospheric perspective, which is based on his own observations [5]. The primary phenomena of atmospheric perspective are detailed as follows.

- A. Closer objects are darker, whereas farther objects are brighter.
- B. Closer objects have greater contrast, whereas farther objects have relatively low contrast.
- C. Closer objects have clearer outlines, whereas farther objects have fuzzier outlines.
- D. Closer colored objects have saturated (bright) colors, whereas farther colored objects have lighter, unsaturated colors. This is because as the distance changes, in addition to the differences in light and shade, the saturation of the objects' colors also varies.

To apply the concept of atmospheric perspective, environmental fog can be added to the game scene in the Unity engine. This simulates the atmospheric perspective of distant objects. Fog masking produces a textured perspective effect that creates a sense of distance and atmosphere in the perspective image (Figure 3). However, virtually no clear improvement is produced in the visual focus area of the character nearest to the player because of the short distance between the objects.

2.4. Lightness perspective. During the conceptual stage of designing characters, illustrators commonly apply shading and color gradients by using compressed lightness perspective techniques to ensuring a clearer physical perception of the objects, enabling the distinction of the characters from the background. The multiplayer real-time strategy game Dota 2 tests the use of both the eyes and hands. In the Dota 2 Character Art Guide [7], the concept art on Page 3 and Pages 11-13 entails using shading to inspect the color levels of the characters, enhancing the clarity and prominence of the game characters against the scenery and meeting the visual requirements of the game.

Figure 4 depicts the effect schematic of `Martinez_Macro_GradientMap.mcr` [8], a plugin for 3ds Max. A new black-and-white gradient map can be generated automatically on the existing UVW map according to bottom to top features of the character. Image processing software such as Photoshop can then be used to superimpose this shading map on the original diffuse map, resulting in a map with lightness perspective effects. When this superimposition process is applied to game effects as shown in Figure 5, using new maps can enable rendering more prominent characters after the characters and objects are processed. A greater sense of layering is observed in the image.



FIGURE 3. Image effect of the scene with added fog

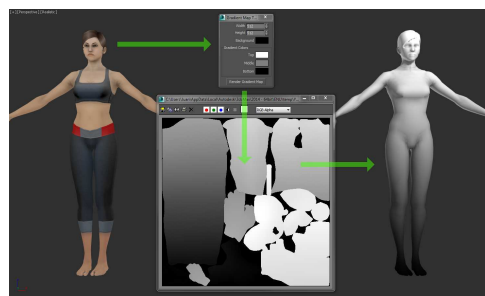


FIGURE 4. Usage schematic of the 3ds Max plugin

3. Methods.

3.1. Shader. Shaders in game engines [9] and materials in 3D animation software generally describe the textures of the 3D models presented in the images. However, shaders and materials are not precisely the same. Shaders primarily refer to algorithm fragments with programmable pipelines. Shaders are further divided into vertex shaders and pixel shaders [10]. Numerous coordinate points define 3D models. Such points are called vertices. Vertex shaders can be used to define the shape of an object model. Because such a model is composed of points, changing the points changes the shape. Pixels can be represented in various colors. Numerous pixels constitute the content on computer displays.



FIGURE 5. Game image after superimposing gradient levels on the maps (left) and close-up comparison before and after use (right)

Pixel shaders serve as the fill colors after the vertices are completely converted. These filling algorithms are directed at every pixel on the screen. Thus, they are denoted as pixel shaders.

In summary, graphics rendering is an execution process similar to an assembly line. Vertex shaders construct 3D models, and the output values of the vertex shaders are subsequently used as input for the pixel shaders to produce visible images. Each shader must at least go through a vertex shader and a pixel shader [11].

3.2. World Pos. shader. When an overlooking, third-person game perspective is adopted, lightness perspective is applied to enhancing the image effects. The initial conception is employed to fill the vertex colors of the model by using the black-and-white gradient from the bottom to the top vertices of the character model. The vertex colors are combined with diffuse blending to produce brightness contrast effects. In Unity, the results obtained using the Strumpy Shader Editor [12] series are not ideal (Figure 6). With the exception of the render pass, the vertex color does not generate transitional colors, indicating that the rendering process is erroneous.

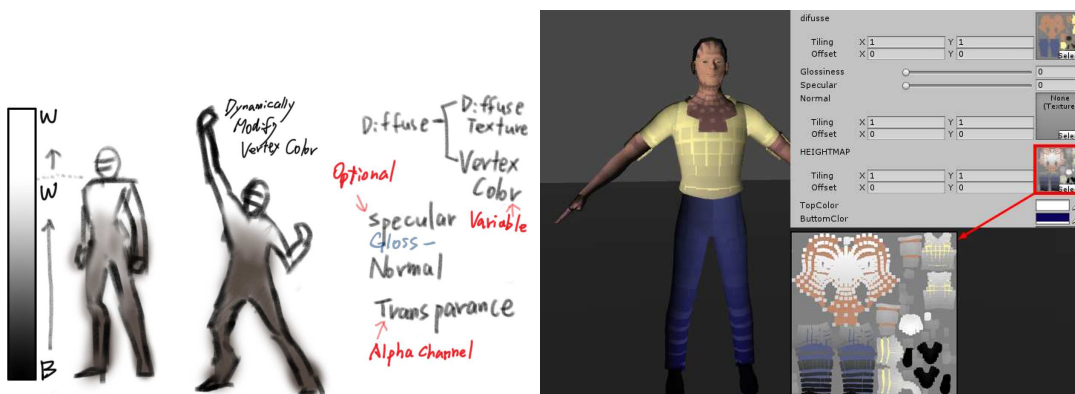


FIGURE 6. Initial conception and failed shader

Pixel shaders are used instead. During the final coloring stage, gradient shading is superimposed onto the relative positions of the models. In this process, the coordinates of the vertices are first transformed from object coordinates to world coordinates before calculation. The coordinates of the heights of the actual points' positions are transformed from object to world coordinates to obtain height values. The principle is to divide the Y height (in world coordinates) of the model vertices by the assumed model height to obtain height ratio values. The height ratios are then used as the UV axes of the gradient map to produce a vertical gradient effect. The vertex and pixel algorithms used are shown in Table 1.

TABLE 1. World Pos. shader pixel calculation formula



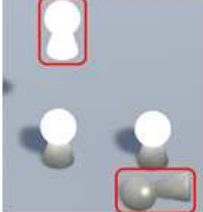
 <p>Original Shader</p>	 <p>World Pos. Shader</p> <p>Calculation Program: float3 pos = mul(_Object2World, v.vertex); o.ramp = (pos.y) / _Height - _GroundYPos;</p>
 <p>Using the World Pos. shader results in problems with angle and height perception.</p>	



FIGURE 7. Image comparison of the World Pos. shader and the gradient map

Figure 7 illustrates a comparison of the images produced using the World Pos. shader and the Martinez_Macro_GradientMap.mcr plugin in the game. The two images are virtually identical. The World Pos. shader also provides height values and gradient color control parameters. Minor adjustments can be conducted at any time in Unity. In addition, color mixing effects can be used in the scene atmosphere.

During actual character movement in the scene of the running game, the gradient shading associated with the World Pos. shader exhibits limitations in height range and direction when the characters jump, climb to high points, or lie horizontally. Because of the shader floor height specified by the program (`_GroundYPos`), when the character models depart from the world coordinate point $Y = 0$, the gradient starting point also changes and does not meet expectations (Table 1).

The amended operations of the script (Table 2) can be used to calculate the height of the surface on which objects are located at all times to reset the coordinate determination value and resolve this height perception problem.

After the height specified in the additional script is revised, the character height problem can be resolved (Figure 8). However, the state of the object when changing axes cannot be managed. In addition, when the revised surface height script is used, each model must execute the specified script individually. It increases the number of drawcalls for static models, such as the existence of numerous trees and rocks sharing a single material. This is a drag on execution performance.

TABLE 2. Revised perception script for surface height

```

using UnityEngine;
using System.Collections;
public class TestCharShaderWorldScript : MonoBehaviour {
    public Transform ground;
    Renderer ren;
    // Use this for initialization
    void Start () {
        if(ground==null){
            ground = this.transform;
        }
        ren = GetComponentInChildren<Renderer>();
    }
    // Update is called once per frame
    void Update () {
        if(ren.material.HasProperty("_GroundYPos")){
            ren.material.SetFloat("_GroundYPos",ground.position.y);
        }
    }
}
    
```

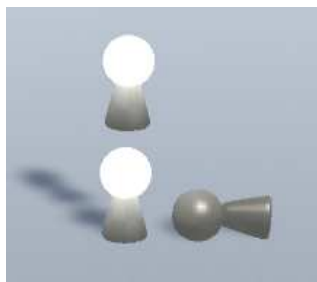


FIGURE 8. Results of script revision on the World Pos. shader

TABLE 3. Vertex and pixel calculation formulae for the Local-Z shader

<p>Original Shader</p>	<p>Vertex: Vertex calculation to obtain the height ratio (ramp) Pixel: The height ratio is used as the UV of the gradient map to obtain the R value for grayscale.</p>	<p>Pixel: The grayscale (heightRamp) is used to calculate the interpolation of black to white between ColorA and ColorB corresponding to ColorA and ColorB.</p>
	<p>Vertex operation formula: float3 pos = v.vertex.xyz; o.ramp = pos.z / _Height; Pixel operation formula: half heightRamp = tex2D(_HeightRamp, float2(0, clamp(IN.ramp, 0, 1))).r;</p>	<p>Pixel operation formula: half3 rampColor = lerp(_ColorA.rgb, _ColorB.rgb, heightRamp);</p>

4. Main Results.

4.1. **Local-Z shader.** When 3ds Max is used to create a 3D character, the character model output to Unity is still Z-up [13]. The position of the model’s pivot is used as the low point to adjust the model’s height. The corresponding operation (Table 3) is then used to complete a Local-Z shader without lightness perspective errors associated with the height and rotational state. Figure 9 depicts the results obtained after applying



FIGURE 9. Comparison of the effects of the unrevised World Pos. shader and the Local-Z shader



FIGURE 10. Comparison of the performance of the double-sided toon shader and the Local-Z shader

this shader to the game. The character model obtained using the Local-Z shader is fundamentally identical to that on the ground obtained using the World Pos. shader without script revision. However, on high platforms, the model maintains its original lightness perspective performance when the Local-Z shader is used.

4.2. Performance testing. This section presents a comparison of the performance of the double-sided toon shader with that of the Local-Z shader. Lighting does not influence the double-sided toon shader used originally, whereas it does affect the Local-Z shader. The models' surfaces generate specular reflections, enhancing their clarity and 3D resolution. However, the triangles and vertices calculated increase exponentially. The Unity status bar showed that the Local-Z shader exhibited slightly higher frames per second compared with the double-sided toon shader. This indicates that the image improvements do not influence the performance, and instead increase it slightly. Unity 5 applies batch processing on skinned mesh renderers, thus improving performance when large numbers of skin models are used (Figure 10).

4.3. Final version. Not all models are suited for pivots set to the (0,0,0) position. Therefore, additional corresponding pivot positions can be added. For example, the pivot can be located at the actual center of a model, enabling the adjustment of height differences and improving the flexibility of the operating parameters. Table 4 shows the optimized code. In other animation software involving a Y-up coordinate system similar to that of Unity, such as Maya, XSI, and Cinema4D, the code is slightly modified to the Local-Y shader used by the corresponding Y-up model. Figure 11 illustrates these effects.

TABLE 4. Optimized shaders and core code

Local-Z used for Z-up models	Local-Y used for Y-up models
float3 pos = v.vertex.xyz; (pos.z + _HeightOffset)/_Height;	float3 pos = v.vertex.xyz; (pos.y + _HeightOffset)/_Height;

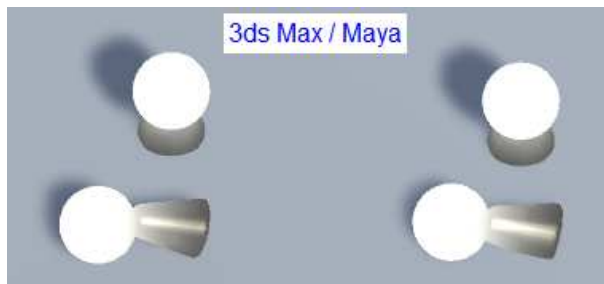


FIGURE 11. Corresponding shaders for various softwares, such as 3ds Max and Maya

5. **Conclusions.** The atmospheric perspective is related to physical phenomena. Because of the effects of the atmosphere on color brightness and saturation, it makes distance objects show lighter and more faded and nearby objects clearer. In this study, we used shaders to strengthen this effect in game images. Lightness changes were added to achieve greater depth in the images. The shaders include tuning parameters for manually manipulating lightness and saturation contrast values according to the atmospheric requirements of the scene to enhance the lightness perspective effect in games rendered in cartoon style [14].

Among animation software used to create 3D game model elements, some programs, such as 3ds Max adopt the Z-up world coordinate system, whereas others, such as Maya, adopt the Y-up world coordinate system. In this study, we created two sets of shaders corresponding to Z-up and Y-up models to meet the requirements of different development environments. These shaders are provided for student use. They can be revised according to project requirements to provide precise adjustments for art. For example, map control vignetting effects or map flow control can be added. The use of the current versions can reduce workloads in 3D games or 3D animation, accelerating the operations of graphics processor units and improving performance.

In the future, research will be based on individual needs, according to project requirements to provide precise adjustments for art. For example, map control vignetting effects or map flow control can be added, to create a magical and sci-fi effect. Further development includes watercolor or charcoal style shaders.

REFERENCES

- [1] J. S. Liu, *The Application of Art in the Production of Game Numerical Control Technology*, Suzhou Art & Design Technology Institute, Art Education, vol.11, 2014.
- [2] Z. G. Zhang, *Art Design in Massively Multiplayer Online Role-Playing Games*, Master Thesis, China Central Academy of Fine Arts, 2012.
- [3] http://wiki.unity3d.com/index.php/Shaders#General_2.
- [4] Y. C. Cheng, *Game Scene Design with Hand-Painted Texture Style*, Master's Thesis, National Taipei University of Education, 2014.
- [5] C. C. Lai, *Color Psychology in Design: Color Imagery and Culture*, Visual Communication, 2003.
- [6] <http://vr.theatre.ntu.edu.tw/fineart/painter-wt/davinci/davinci.htm>.
- [7] *Dota 2 Character Art Guide*, <http://media.steampowered.com/apps/dota2/workshop/Dota2CharacterArtGuide.pdf>.
- [8] *3ds Max Plugin*, https://dl.dropboxusercontent.com/u/2904948/MaxScript/Martinez_Macro_GradientMap.mcr.
- [9] <http://www.dotblogs.com.tw/sonic10690/archive/2009/01/06/6650.aspx>.

- [10] Kenny Lammers, *Unity Shaders and Effects Cook Book*, PACKT Publishing, 2013.
- [11] <http://blog.xuite.net/laishiekai/studio/22146507-Shader%E9%81%8B%E4%BD%9C%E6%B5%81%E7%A8%8B%E7%9A%84%E5%88%9D%E5%BF%83%E5%BE%97>.
- [12] *Strumpy Shader Editor*, <http://forum.unity3d.com/threads/56180-Strumpy-Shader-Editor-4.0a-Massive-Improvements>.
- [13] *Autodesk 3dsMax Online Help*, <http://help.autodesk.com/view/3DSMAX/2015/ENU/?guid=GUID-B98414B9-4F28-45F4-A1F4-9DA994548ED9>.
- [14] X. Zhang, K. L. Chan and M. Constable, Atmospheric perspective effect enhancement of landscape photographs through depth-aware contrast manipulation, *IEEE Transactions on Multimedia*, vol.16, no.3, pp.653-667, 2014.
- [15] I. Condry, *The Soul of Anime: Collaborative Creativity and Japan's Media Success Story*, Duke University Press Books Publishing, 2013.
- [16] L. Goldmann, T. Ebrahimi, P. Lebreton and A. Raake, Towards a descriptive depth index for 3D content: Measuring perspective depth cues, *International Workshop on Video Processing & Quality Metrics for Consumer Electronics*, 2012.
- [17] <https://cg2010studio.wordpress.com/2011/06/29/shader/>.
- [18] *OpenGL Rendering Pipeline*, <http://www.opentk.com/node/1342>.
- [19] C.-H. Kao, *User-Centered Design for Character in Video Game*, Department of Industrial Engineering and Management, National Chiao Tung University, 2008.
- [20] A. M. Wu, D. Xu, H. X. Wang and J. Wu, Object size constancy calculations based on visual psychology, *Electronic Journal of the Institute of Computer Science*, vol.34, Beijing Jiaotong University, 2006.
- [21] H.-W. Tu, *Extravagant and Changeable Color – Experimental Creation Description by Hung-Wei Tu*, Graduate Institute and Department of Visual and Media Arts, Nanhua University, 2011.