Fachhochschule Luzern

3. Semester

Shaderprogramming Fragmentshader (teamcolormanipulation)

Studiengang Digital Ideation, Bachelor
GLSL, Fragmentshader, Webgl, Javascript

Hischier Simon
simon.hischier@stud.hslu.ch

January 15, 2018

Inhalt

1 Abstract

This work explores the dynamic setup of colors for different Teams. This work is mainly based on Graphicsmanipulation via shader programming. The work won't go into detail on how to create gameobjects and will not really go into detail on how color composition works.

2 Introduction

The idea for this work is to manipulate Unit- or Building-textures via Shader. In Real time strategy (RTS) games based on sprites the teamcolors were added in the shader pipeline. This was due to space limitations and therefore games shouldn't have the same sprites multiple times just colored differently for different teams. This idea was popular during the RTS boom, the main inspiration for this work was Age of Empires.

Figure 1: Image from Age of Empires. Source: https://gamedev.stackexchange.com/questions/131169/how-do-i-change-color-of-one-part-of-sprite-in-game-maker

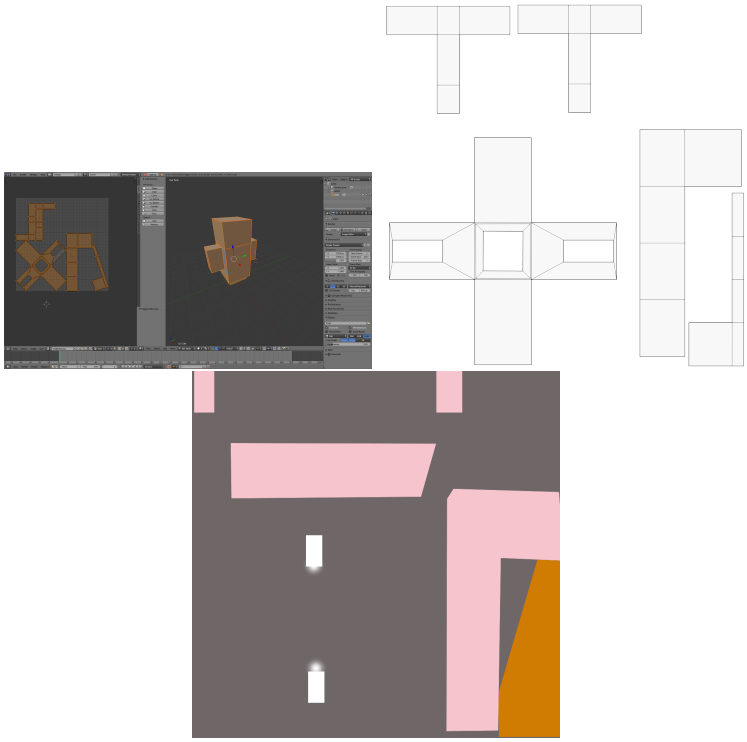
This workaround for space limitations has lost its purpose in the days of digital downloads and Storage space in the terabytes. There is another appeal to manage team colors by shadermanipulation. Games with highly customizable parts or almost unlimited numbers of teams could profit from changable teamcolors based on a programming solution. Users can choose their favorite color from the complete Red/Green/Blue (RGB) colorspace instead of choosing from a fixed set of predefined colors. Of course those colors will not always be ideal for the setting but that would just be a limitation that could be set by the gamedesigner.

3 Work

3.1 Texture work

Webgl does need a model and a shader to display anything at all. To start that project I had to create a 3D model and UV map it correctly. After that the model has to be loaded into webgl, the shader has to be compiled and the

material which includes the shader has to be applied to the model. The Texture is passed to the shader and the uv mapping will then read the color values for each fragment from the texture. The white texturepart in the texturemap is transparent or semitransparent to test the shader.



The following Textures are property of Valve Corporation and show the normal Texture and the modified texture with semitransparent parts. The shader code will mix the teamcolor with the texture based on the alphachannel translucency for each pixel.

3.2 Colorchanging

The changeable colors are calculated in realtime on the graphics card. Multiple objects of the same team mix the choosen color into the texture before displaying. This is done via alpha channel probing. The alphachannel (a value between 0 and 1 from black to white) is used to calculate a mixcolor between texturecolor and defined teamcolor. If the alphachannel for a pixel is 1 only the teamcolor is used and if it is 0 only the teamcolor is displayed. Everything inbetweent gets a mixed color from teamcolor and texturecolor. On top of the changeable colors this can be used to create an enemy color. This is shown as example on the image number 3. The color for objects on the left can be chosen and the color on the right side objects are then calculated in hsv space with a fixed offset degree from the left. This can ensure that the enemy team could always have a complementary color or the like. Artists could define that teams have to pick colors in a fixed degree space from each other to guarantee, that the teams fit together visually and so on.

3.3 Animations

Animations depend on changing input from Javascript. Animations in a Vertexshader can only manipulate the coordinates per vertex (see beam animation) and it would need a geometryshader to add more vertexpoints to a model. This was not investigated further during this work. The teamcolor can here be used to visually fit animations to the teams instead of an animation that breaks with the visuals. The beam interpolates the teamcolor based on the time remaining until the unit vanishes. The closer it is to port, the more the unit is colored in

the teamcolors. In the hologramm the teamcolor is interpolated based on a sine wave on top of the texture interpolation.

Figure 2: Beam animation.

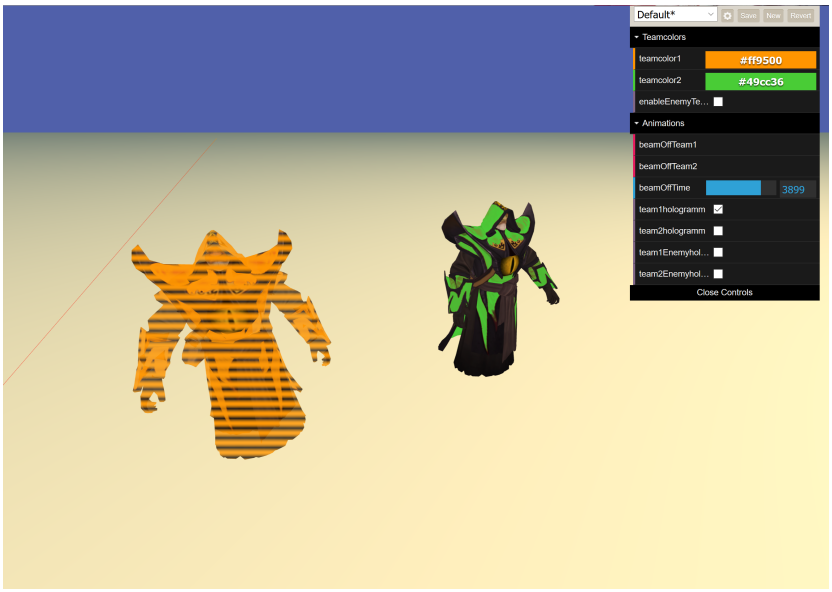


Figure 3: Hologramm animation.

3.4 Shader programm

3.4.1 Code explanation

The Vertex shader contains very little that is needed for the teamcolor changes. It passes the uvmapping to the Fragmentshader. The Vertexshader was used to manipulate the geometry for the beam animation, the relevant codepart can be seen in the beamOff brackets. The Fragmentshader interpolates the teamcolors with the texture colors based on the alpha map at all times. The Fragmentshader changes the alphamap transparency based on the current progress of a beamoff and interpolates the teamcolor stronger at the end of the beamoff. If the object should be displayed as hologramm, the Fragmentshader will overlay the teamcolors over the normal texture and the alpha transparency is set to a fixed semitransparency for hologramms. Holograms have an animation where the beamlines slowly move in a vertical line.

3.4.2 Code

The Vertexshader:

```
<script type="x-shader/x-vertex" id="teamcolorVertexShader">
   precision mediump float;
   varying vec2 vUv;
   varying vec3 vertexWorldPos;
   uniform int timeStamp;
   uniform bool beamOff;
   uniform float beamOffTime;

   void main()
   {
      vUv = uv;
      vertexWorldPos = position;

      float posx = position.x;
      float posy = position.y;
      float posz = position.z;
      if (beamOff)
      {
         posx = position.x * (1.0 - beamOffTime);
         posz = position.z * (1.0 - beamOffTime);
         posy = position.y + position.y * beamOffTime * 1.5;
      }
      vec4 modelViewPosition = modelViewMatrix * vec4(
         posx,
         posy,
         posz,
         1.0);
      gl_Position = projectionMatrix * modelViewPosition;
   }
```

```
</script>
```

The Fragmentshader:

```
<script type="x-shader/x-fragment" id="teamcolorFragmentShader">
   precision mediump float;
   #define M_PI 3.14159265358979323846264338327795
   uniform sampler2D texture;
   uniform vec3 teamcolor;
   uniform int timeStamp;
   uniform bool beamOff;
   uniform float beamOffTime;
   uniform bool isHologramm;
   uniform float hologrammOffset;
   varying vec3 vertexWorldPos;

   varying vec2 vUv;

   void main()
   {
      vec4 fragColor = texture2D(texture, vUv);
      if (beamOff)
      {
         fragColor.a = fragColor.a * sin(1.0 - beamOffTime);
      }

      // interpolate teamcolor
      fragColor.r =
         fragColor.r * fragColor.a + teamcolor.r * ( 1.0 - fragColor.a);
      fragColor.g =
         fragColor.g * fragColor.a + teamcolor.g * ( 1.0 - fragColor.a);
      fragColor.b =
         fragColor.b * fragColor.a + teamcolor.b * ( 1.0 - fragColor.a);
      fragColor.a = 1.0;

      if (isHologramm)
      {
         float lineVal = sin(
            (gl_FragCoord.y + M_PI * hologrammOffset * 100.0)
            * 0.3);
         float lineVal2 = sin(
            (gl_FragCoord.y + (M_PI + M_PI / 2.0) * hologrammOffset *
                100.0)
            * 0.3);
         float lineVal3 = sin(
            (gl_FragCoord.y + (M_PI / 2.0) * hologrammOffset * 100.0)
            * 0.3);
         float lineVal4 = sin(
            (gl_FragCoord.y + (M_PI * mix(-1.0, 1.0, hologrammOffset)) *
                10.0)
```

```
            * 0.1);

         fragColor.r = mix(fragColor.r, teamcolor.r, max(0.0,
             abs(lineVal4)));
         fragColor.g = mix(fragColor.g, teamcolor.g, max(0.0,
             abs(lineVal4)));
         fragColor.b = mix(fragColor.b, teamcolor.b, max(0.0,
             abs(lineVal4)));
         fragColor.a = 0.8;
      }

      if (beamOff)
      {
         fragColor.a = max(0.5, 1.0 - beamOffTime);
      }

      gl_FragColor = fragColor;
   }
</script>
```

4 Conclusion

The solution based on the alpha channel is limiting the usefullness of textures. The alpha channel can not be used to create transparent parts in objects and therefore all objects must have holes etc. implemented based on the geometry. Semitransparent parts are not possible in a game object where a teamshader is used. A different approach would be to have a defined RGB color that gets replaced by the team color. This would only allow for either full teamcolor or no teamcolor per pixel and it would not be possible to blend/mix the teamcolor with a texture. A possible third solution would be to have a second black and white texture on which the teamcolor parts would be one color and the normal texture colors would be the other. Blending of the two would be possible based on a RGB color range from black to white. The alpha channel of the texture could be normally used and it would not put any limitations on the texture image. The only drawback would be an increase in required space and a prolonged work pipeline due to additional textures, putting more workload on the artists.

5 Further research

The coloring based on shaderprogramming is interesting for programmers but limits the freedom of artists. This two fields could work together by limiting the color choice based on a range of possible colors set by the artist. The artist could specify a range of colors that will work well with the overall perception of the game world. Further work could include interesting variations in the reflection factor of the teamcolors based on a good/evil system where the teams with a

good background whould be more reflective (shiny) and enemy teams would reflect less hence they would absorb the light which is a represent of something bad. A glow effect could be worked into the texture based on the teamcolor and the good/bad system. The possibilities would allow for a lot of interesting projects based on this colorinterpolation.

6 References, Acronys and Figures

Glossary

Fragmentshader A subprogramm run on the graphics card to calculate and manipulate the color for each Fragment which later is mapped to the monitor. 7

Vertexshader A subprogramm run on the graphics card to manipulate vertexpoints of a geometry. 5, 7

Acronyms

RGB Red/Green/Blue. 3, 9

RTS Real time strategy. 3

List of Figures