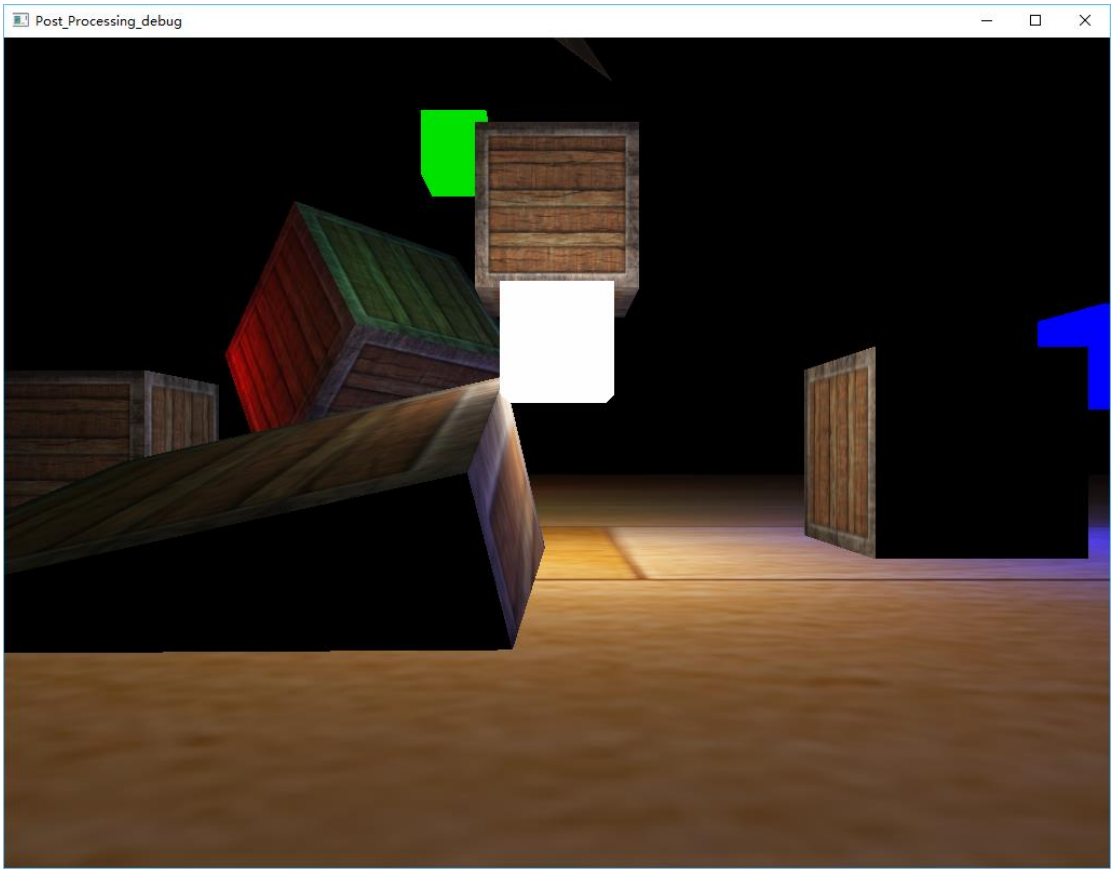
Projet LMG 2016

Yuheng ZHAO

Bloom post-processing

1. Implémentation du Bloom

Dans mon projet, l'implémentation du Bloom est basée sur la rendu avec HDR colorbuffer. La luminosité des couleurs est entre 1,5 et 15,0.



Ensuite, on extrait les fragments qui dépasse un seuil de luminosité. Pour obtenir la couleur des fragments qui ont des luminosités suffisant, il faut utiliser `GL_COLOR_ATTACHMENT0` et `GL_COLOR_ATTACHMENT1`. On va dire à OpenGL qu'il faut dessiner les couleurs de la scène dans `GL_COLOR_ATTACHEMENT0 (m_iColorBuffersFBO[0])`, et les couleurs qui dépasse le seuil dans `GL_COLOR_ATTACHEMENT1(m_iColorBuffersFBO[0])`.

Dans le shader `FS_bloom.glsl`, on a calculé les lumières normales. Et puis on

transforme les la couleur de lumière en niveau de gris pour déterminer s'il dépasse le seuil.

```
    float brightness = dot(result, vec3(0.2126, 0.7152, 0.0722));
    if(brightness > 1.0)
        BrightColor = vec4(result, 1.0);
```

On fait la même chose pour rendre les boites qui émit de la lumière. Donc on aura finalement un buffer qui contient que les lumières qui dépassent le seuil.

Pour faire du blur, on a besoin faire du Gaussian blur de 2 passes. Une horizontale et une verticale, pour éviter des calculs lourd avec le noyau gaussien.

Dans le shader FS_blur.glsl, on a un certain nombre de poids gaussiens :

```
uniform float weight[5] = float[] (0.2270270270, 0.1945945946, 0.1216216216, 0.0540540541, 0.0162162162);
```

En utilisant ces poids, on calcule à nouveau des couleurs en changeant le coordonné de texture.

```
if(horizontal)
{
    for(int i = 1; i < 5; ++i)
    {
    result += texture(image, TexCoords + vec2(tex_offset.x * i, 0.0)).rgb * weight[i];
    result += texture(image, TexCoords - vec2(tex_offset.x * i, 0.0)).rgb * weight[i];
    }
}
else
{
    for(int i = 1; i < 5; ++i)
    {
    result += texture(image, TexCoords + vec2(0.0, tex_offset.y * i)).rgb * weight[i];
    result += texture(image, TexCoords - vec2(0.0, tex_offset.y * i)).rgb * weight[i];
    }
}
```

Comme on a crée deux FBOs et chacun a un buffer associé :

```
GLuint m_iPingPongFBO[2], m_iPingPongColorBuffersFBO[2];
glGenFramebuffers(2, m_iPingPongFBO);
glGenTextures(2, m_iPingPongColorBuffersFBO);
```

On peut premièrement binder le m_iPingPongFBO[0] avec m_iColorBuffersFBO[1]

pour démarrer des itérations ci –dessous :

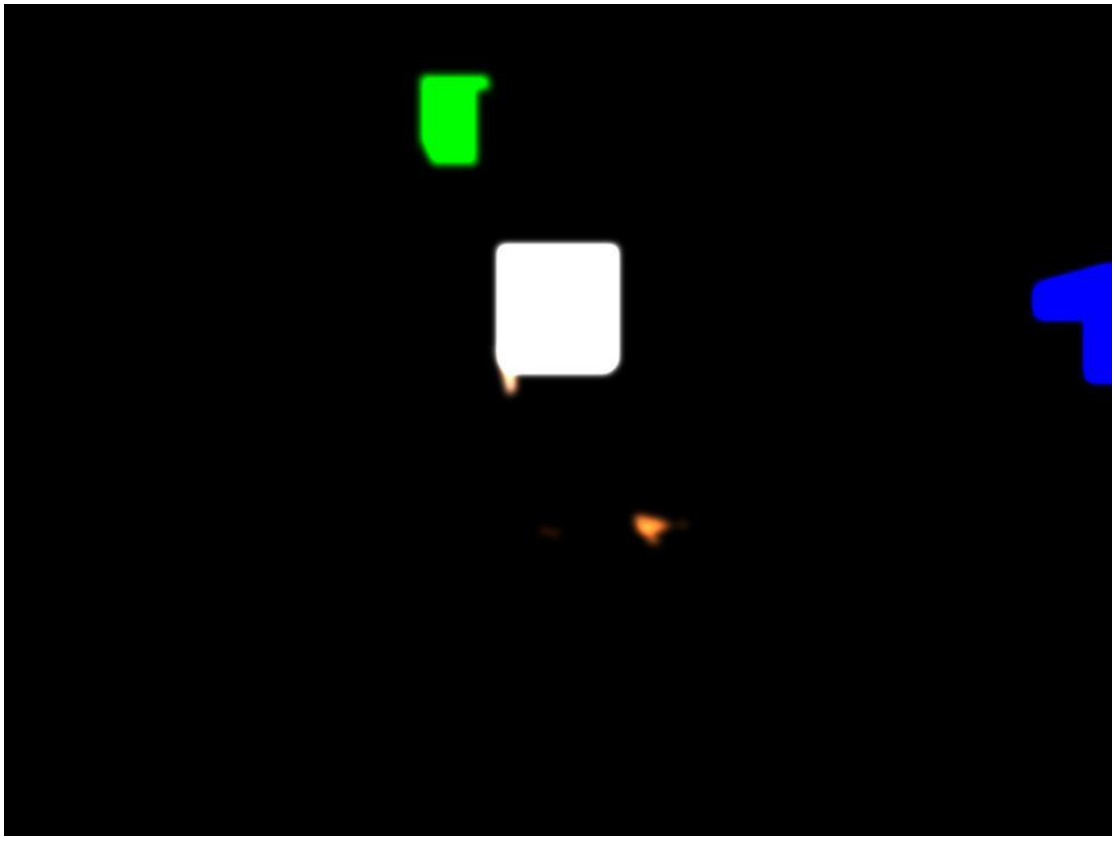
```
GLboolean horizontal = true, first_iteration = true;
GLuint amount = 10;
m_GPUProgramBlur.bind();
for (GLuint i = 0; i < amount; i++)
{
   glBindFramebuffer(GL_FRAMEBUFFER, m_iPingPongFBO[horizontal]);
   glUniform1i(glGetUniformLocation(m_GPUProgramBlur.getID(),
"horizontal"), horizontal);
   glBindTexture(GL_TEXTURE_2D,         first_iteration         ?
m_iColorBuffersFBO[1] : m_iPingPongColorBuffersFBO[!horizontal]);
    // bind texture of other framebuffer (or scene if first
iteration)
   RenderQuad();
   horizontal = !horizontal;
   if (first_iteration)
          first_iteration = false;
   }
```

On fait du blur avec le premier FBO `m_iPingPongFBO[0]` et on a une texture

`m_iColorBuffers[0]`, ensuite on utilise cette texture pour le rendu avec

`m_iPingPongFBO[1]` qui produit une texture `m_iColorBuffers[1]` et

`m_iPingPongFBO` réutilise cette texture pour continuer l'itération.



Après pour combiner les deux rendus qu'on a obtenu, on va blender ces deux rendus en faisant l'addition :

```
vec3 hdrColor = texture(scene, TexCoords).rgb;
vec3 bloomColor = texture(bloomBlur, TexCoords).rgb;
hdrColor += bloomColor;
```