PRACTICA 4

Aumento de Realismo

Parte obligatoria

Adrián David Morillas Marco y Guillermo Meléndez Morales
Diseño y Desarrollo de Videojuegos + Ingeniería de Computadoras

Parámetros del Motion Blur a través del teclado

Los parámetros del Motion Blur son definidos en la función renderFunc () de la clase main, mediante la siguiente función:

glBlendColor (red, Green, blue, alfa)

Siendo alfa, el índice de transparencia.

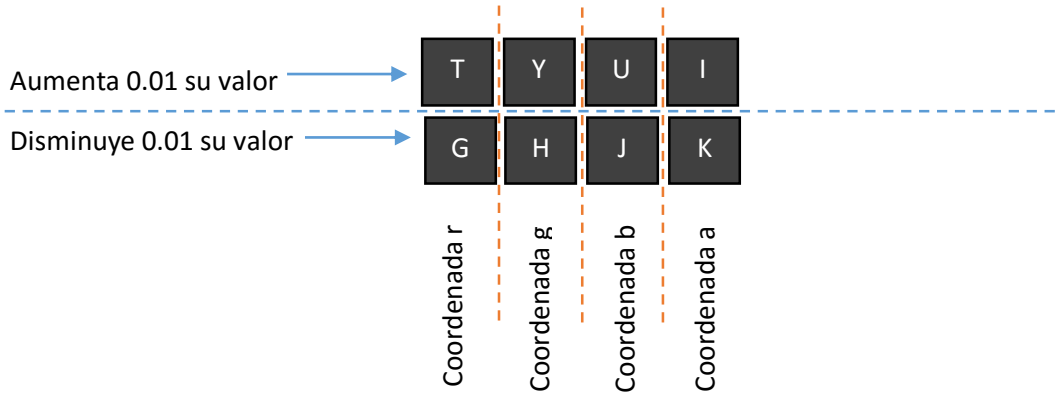
Para poder controlar los parámetros de dicha función será necesario crear un vector de cuatro coordenadas:

```
// Creamos un vector de 4 coordenadas, el cual será sustituido por los parametros actuales
glm::vec4 glColor = glm::vec4(0.8f, 0.8f, 0.8f, 0.0f);
```

Una vez definido el vector, se le asigna a la función ya creada, donde se definen los parámetros del Motion Blur, las coordenadas de este vector:

```
// le definimos el vector glColor a glBlenColor y lo mostramos por consola
glBlendColor(glColor.x, glColor.y, glColor.z, glColor.w);
```

Finalmente, en el método keyboardFunc () se define el control por teclas. El control de dichos parámetros se efectúa de la siguiente manera:



Esto en código sería:

```
// aumenta la coordenada x o red
if (key == 't'){
        if (glColor.x < 1){
                glColor.x += 0.01;
        }
}
// disminuye la coordenada x o red
if (key == 'g'){
        if (glColor.x > 0){
                glColor.x -= 0.01;
        }
}
// aumenta la coordenada y o green
if (key == 'y'){
        if (glColor.y < 1){
                glColor.y += 0.01;
        }
}
```

```
// disminuye la coordenada y o green
if (key == 'h'){
		if (glColor.y > 0){
			glColor.y -= 0.01;
		}
}
// aumenta la coordenada z o blue
if (key == 'u'){
		if (glColor.z < 1){
			glColor.z += 0.01;
		}		}
// disminuye la coordenada z o blue
if (key == 'j'){
		if (glColor.z > 0){
			glColor.z -= 0.01;
		}
}
// aumenta la coordenada w o alpha
if (key == 'i'){
		if (glColor.w < 1){
			glColor.w += 0.01;
		}
}
// disminuye la coordenada w o alpha
if (key == 'k'){
		if (glColor.w > 0){
			glColor.w -= 0.01;
		}
}
```

Control de los parámetros del DOF (Depth of Field)

Los parámetros de DOF son definidos en el shader Postproceso de fragmentos:

```
const float focalDistance = -25.0;
const float maxDistanceFactor = 1.0/5.0;
```

Para poder controlarlos a través del teclado, será necesario, crear las dos variables en la clase main, y subirlos mediante uniforms al shader de Postproceso de fragmentos:

- Creamos las variables en la clase main:

```
// creamos los valores del DOF
float FocalDistance = -15.0f;
float MaxDistanceFactor = 1.0 / 5.0;
```

- Creamos los identificadores de las variables, en la clase main:

```
int uFocalDistance;
int uMaxFocalDistance;
```

- Definimos el valor de las variables uniform en RenderFunc (), en la clase main:

```
float focalDistance = FocalDistance;
float maxDistanceFactor = MaxDistanceFactor;

if (uFocalDistance != -1){
        glUniform1f(uFocalDistance, focalDistance);
}
if (uMaxFocalDistance != -1){
        glUniform1f(uMaxFocalDistance, maxDistanceFactor);
}
```

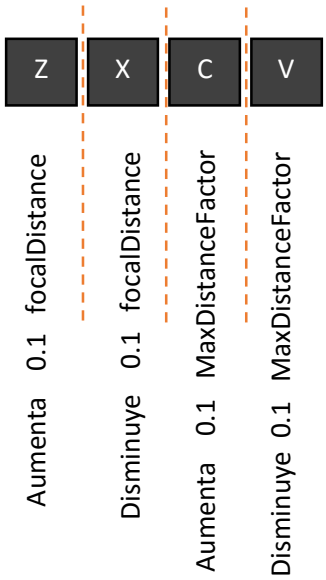
- Asignamos las variables en initShaderPP (), en la clase main:

```
uFocalDistance = glGetUniformLocation(postProccesProgram, "focalDistance");
uMaxFocalDistance = glGetUniformLocation(postProccesProgram, "maxDistanceFactor");
```

- Cambiamos las variables uniform en el shader de Postproceso de fragmentos:

```
uniform float variable_near;
uniform float variable_far;
//const float focalDistance = -25.0;
//const float maxDistanceFactor = 1.0/5.0;
```

A continuación en la función keyboardFunc (), definimos el control de estas dos variables. Los controles son los siguientes:



Esto en código sería:

```
if (key == 'z'){
		FocalDistance += 0.1;
}
// disminuye la distancia focal
if (key == 'x'){
		FocalDistance -= 0.1;
}
// aumenta el factor de la distacia (desenfoque)
if (key == 'c'){
		if (MaxDistanceFactor < 1){
			MaxDistanceFactor += 0.1;
		}
}
// disminuye el factor de la distancia (desenfoque)
if (key == 'v'){
		if (MaxDistanceFactor > 0.1){
			MaxDistanceFactor -= 0.1;
		}
}
```

Buffer de profundidad para controlar DOF (Depth of Field)

El primer paso a efectuar será cambiar vertexTex por depthTex, en el calculo del método main, del shader de Postproceso de fragmentos:

```
float dof = abs(texture(vertexTex,texCoord).z - focalDistance) * maxDistanceFactor;
```

```
float dof = abs(texture(depthTex,texCoord).r - focalDistance) * maxDistanceFactor;
```

Para ello subiremos el buffer de profundidad al shader de Postproceso:

- Se crea el identificador, en la clase main:
  ```
  unsigned int depthBuffTexId;
  ```

- Se crea el identificador uniform, en la clase main:
  ```
  int uDepthTexPP;
  ```

- Definimos la variable en renderFunc (), en la clase main:
  ```
  if (uDepthTexPP != -1)
  {
                  glActiveTexture(GL_TEXTURE0 + 2);
                  glBindTexture(GL_TEXTURE_2D, depthBuffTexId);
                  glUniform1i(uDepthTexPP, 2);
  }
  ```

- Referenciamos la variable:
  ```
  uDepthTexPP = glGetUniformLocation(postProccesProgram, "depthTex");
  ```

- Definimos la variable en el shader de Postproceso de fragmentos:
  ```
  uniform sampler2D depthTex;
  ```

Finalmente se hará un cambio de rango, para obtener el valor del buffer de profundidad, en coordenadas de la cámara. Para ello primero, se subirán al shader de Postproceso, las variables Far y Near:

- Creamos las variables en la clase main:
  ```
  float Far = 50.0;
  float Near = 1.0;
  ```

- Creamos los identificadores en la clase main:
  ```
  unsigned int uFar;
  unsigned int uNear;
  ```

- Definimos el valor de las variables:
  ```
  float variable_near = Near;
  float variable_far = Far;

  if (uFar != -1){
          glUniform1f(uFar, Far);
  }
  ```

```
if (uNear != -1){
        glUniform1f(uNear, Near);
}
```

- Referenciamos las variables en initShaderPP, en la clase main:
  uNear = glGetUniformLocation (postProccesProgram, "variable_near");
  uFar = glGetUniformLocation(postProccesProgram, "variable_far");

- Creamos las variables en el shader de Postproceso de fragmentos:
  uniform float variable_near;
  uniform float variable_far;

A continuación se efectúa el cambio de coordenadas, para ello se empleará la siguiente formula:

$$z_c = \frac{-\text{near} * \text{far}}{(z_b * (\text{near} - \text{far}) + \text{far})}$$

ara ello se sustituye la función:

```
float dof = abs(texture(depthTex,texCoord).r - focalDistance) * maxDistanceFactor;
```

Por:

```
float dof = abs((-variable_near*variable_far/(texture(depthTex,texCoord).r*(variable_near-variable_far) +
variable_far)) - focalDistance) * maxDistanceFactor;
```

Añadir mascaras de convolucion y controlarlas

Una mascara de convolucion está definida por un array de 25 float en esta práctica, como se puede comprobar, con la máscara definida inicialmente en el shader de Postproceso de fragmentos:

```
const float mask[MASK_SIZE] = float[](
        1.0*maskFactor, 2.0*maskFactor, 3.0*maskFactor,2.0*maskFactor, 1.0*maskFactor,
        2.0*maskFactor, 3.0*maskFactor, 4.0*maskFactor,3.0*maskFactor, 2.0*maskFactor,
        3.0*maskFactor, 4.0*maskFactor, 5.0*maskFactor,4.0*maskFactor, 3.0*maskFactor,
        2.0*maskFactor, 3.0*maskFactor, 4.0*maskFactor,3.0*maskFactor, 2.0*maskFactor,
        1.0*maskFactor, 2.0*maskFactor, 3.0*maskFactor,2.0*maskFactor, 1.0*maskFactor);
```

Siendo maskFactor = 1/ (La suma de todos los float de la mascara)

```
const float maskFactor = float (1.0/16);
```

La mascará que efectuaría este array sería:

$$\begin{matrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 3 & 4 & 3 & 2 \\ 3 & 4 & 5 & 4 & 3 \\ 2 & 3 & 4 & 3 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{matrix}$$

En nuestro caso, las máscaras han de ser subidas al shader de Postproceso a través de una variable uniform. Para ello:

- Creamos la variable mascara y factorMascara en la clase main:

```
float Mascara[25] = { 1.0f, 2.0f, 3.0f, 2.0f, 1.0f,
                      2.0f, 3.0f, 4.0f, 3.0f, 2.0f,
                      3.0f, 4.0f, 5.0f, 4.0f, 3.0f,
                     2.0f, 3.0f, 4.0f, 3.0f, 2.0f,
                      1.0f, 2.0f, 3.0f, 2.0f, 1.0f};

float factorMascara = 1.0f;
```

- Creamos el identificador de la variable en la clase main:

```
unsigned int uMascaraTex[25] {2};
```

Se le ha colocado el valor ‘2’ para evitar posible fallo

- Definimos el valor de las variables en renderFunc () de la clase main:

```
for (int j = 0; j < MASK_SIZE; j++){
                        factorMascara += Mascara[j];
}

factorMascara = 1 / factorMascara;

float mascara[25] = {Mascara[0] * factorMascara, Mascara[1] * factorMascara, Mascara[2]*factorMascara,
Mascara[3]*factorMascara, Mascara[4] * factorMascara, Mascara[5] * factorMascara, Mascara[6] * factorMascara,
Mascara[7]*factorMascara, Mascara[8] * factorMascara, Mascara[9] * factorMascara, Mascara[10] *factorMascara,
Mascara[11]*factorMascara, Mascara[12] * factorMascara, Mascara[13] * factorMascara, Mascara[14]*factorMascara,
Mascara[15] * factorMascara, Mascara[16] * factorMascara, Mascara[17] * factorMascara,
Mascara[18]*factorMascara, Mascara[19] * factorMascara, Mascara[20] * factorMascara, Mascara[21] *factorMascara,
Mascara[22] * factorMascara, Mascara[23] * factorMascara, Mascara[24] * factorMascara};

if (uMascaraTex [25] != -1){
        glUniform1fv(uMascaraTex [25], 25, mascara);
}
```

- Referenciamos las variables en initShaderPP, en la clase main:

```
uMascaraTex [25] = glGetUniformLocation(postProccesProgram, "mascara");
```

- Creamos la variable en el shader Postproceso de fragmentos
  uniform float mascara[25];

- Cambiamos la variable mask por mascara en el main () del shader de Postproceso:
  color += texture(colorTex, iidx,0.0) * mascara[i];

Finalmente en la función keyboardFunc(), definimos nuevas mascaras de convolucion:

1

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & -8 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

2

$$\begin{matrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 3 & 4 & 3 & 2 \\ 3 & 4 & 5 & 4 & 3 \\ 2 & 3 & 4 & 3 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{matrix}$$

3

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & -1 & 2 & 0 \\ 0 & -1 & -8 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

4

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & -9 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Esto en código se traduce a:

```
if (key == '1'){

        Mascara [0] = 0;
        Mascara [1] = 0;
        Mascara [2] = 0;
        Mascara [3] = 0;
        Mascara [4] = 0;

        Mascara [5] = 0;
        Mascara [6] = 1;
        Mascara [7] = 1;
        Mascara [8] = 1;
        Mascara [9] = 0;

        Mascara [10] = 0;
        Mascara [11] = 1;
        Mascara [12] = -8;
        Mascara [13] = 1;
        Mascara [14] = 0;
```

```
                Mascara [15] = 0;
                Mascara [16] = 1;
                Mascara [17] = 1;
                Mascara [18] = 1;
                Mascara [19] = 0;

                Mascara [20] = 0;
                Mascara [21] = 0;
                Mascara [22] = 0;
                Mascara [23] = 0;
                Mascara [24] = 0;

}
if (key == '2'){

                Mascara[0] = 1;
                Mascara[1] = 2;
                Mascara[2] = 3;
                Mascara[3] = 2;
                Mascara[4] = 1;

                Mascara[5] = 2;
                Mascara[6] = 3;
                Mascara[7] = 4;
                Mascara[8] = 3;
                Mascara[9] = 2;

                Mascara[10] = 3;
                Mascara[11] = 4;
                Mascara[12] = 5;
                Mascara[13] = 4;
                Mascara[14] = 3;

                Mascara[15] = 2;
                Mascara[16] = 3;
                Mascara[17] = 4;
                Mascara[18] = 3;
                Mascara[19] = 2;

                Mascara[20] = 1;
                Mascara[21] = 2;
                Mascara[22] = 3;
                Mascara[23] = 2;
                Mascara[24] = 1;
}
if (key == '3'){
                Mascara[0] = 0;
                Mascara[1] = 0;
                Mascara[2] = 0;
                Mascara[3] = 0;
                Mascara[4] = 0;

                Mascara[5] = 0;
                Mascara[6] = -2;
                Mascara[7] = -1;
                Mascara[8] = 0;
                Mascara[9] = 0;

                Mascara[10] = 0;
                Mascara[11] = -1;
                Mascara[12] = 1;
                Mascara[13] = 1;
                Mascara[14] = 0;

                Mascara[15] = 0;
                Mascara[16] = 2;
                Mascara[17] = 1;
                Mascara[18] = 0;
                Mascara[19] = 0;

                Mascara[20] = 0;
```

```
                Mascara[21] = 0;
                Mascara[22] = 0;
                Mascara[23] = 0;
                Mascara[24] = 0;
}
if (key == '4'){
                Mascara[0] = 0;
                Mascara[1] = 0;
                Mascara[2] = 0;
                Mascara[3] = 0;
                Mascara[4] = 0;

                Mascara[5] = 0;
                Mascara[6] = -1;
                Mascara[7] = -1;
                Mascara[8] = -1;
                Mascara[9] = 0;

                Mascara[10] = 0;
                Mascara[11] = -1;
                Mascara[12] = 9;
                Mascara[13] = -1;
                Mascara[14] = 0;

                Mascara[15] = 0;
                Mascara[16] = -1;
                Mascara[17] = -1;
                Mascara[18] = -1;
                Mascara[19] = 0;

                Mascara[20] = 0;
                Mascara[21] = 0;
                Mascara[22] = 0;
                Mascara[23] = 0;
                Mascara[24] = 0;
}
```