11-1-2016

Práctica Obligatoria 4

Informática Gráfica

Sebastián Guisado Barreras y Susana Pineda De Luelmo

1. **Controla los parámetros del Motion Blur a través de teclado:**
   El Motion Blur es el rastro dejado por los objetos en movimiento en una fotografía o en una secuencia de imágenes.
   En nuestro caso contaremos con el color del rastro que dejan nuestros cubos al moverse (glBlendColor()) el cual cambiaremos y variaremos mediante teclado gracias a una nueva variable que se restará al color y se sumará al alpha de nuestro glBlendColor() dando la sensación de Motion Blur.

   En el archivo main.cpp creamos una nueva variable para ajustar el motion blur y la inicializamos a 0:

```
float blurIntensity = 0.0f;
```

   A continuación en la función de renderizado cambiamos la forma de hacer el motion blur.
   Pasamos de:

```
glBlendColor(0.5f, 0.5f, 0.5f, 0.6f);
```

   a :

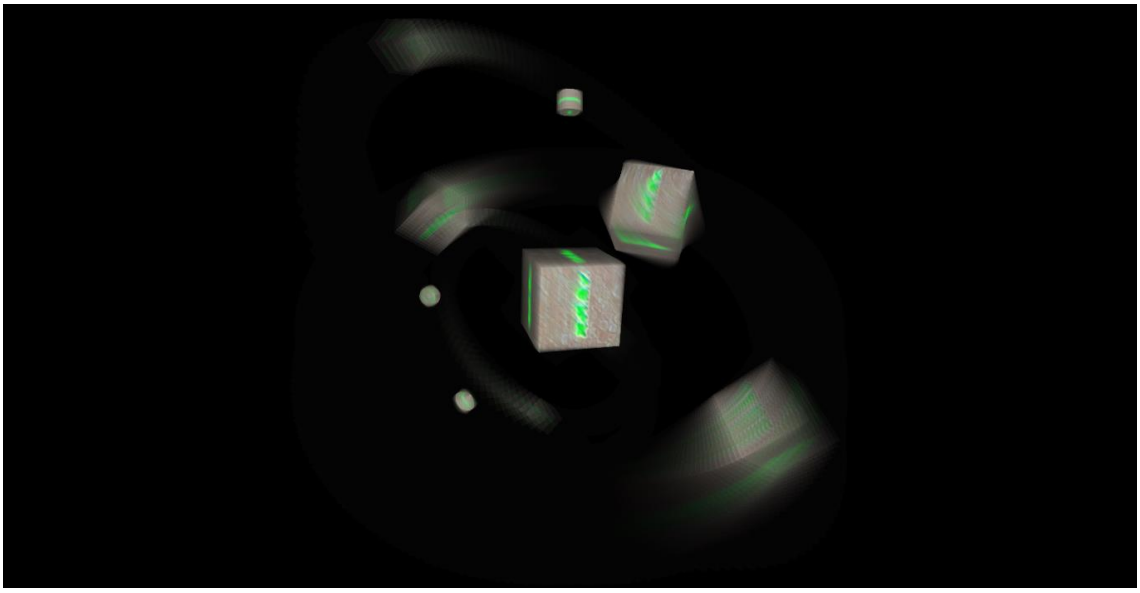
```
float colorIntensity = 0.5f - blurIntensity;
float alphaIntensity = 0.5f + blurIntensity;
glBlendColor(colorIntensity, colorIntensity, colorIntensity,
alphaIntensity);
```

   De esta forma podemos controlar la intensidad del color y del canal alpha (transparencia) mediante teclado. Una vez que tenemos la nueva forma de calcular el motion blur implementada pasamos a controlar los valores de los que depende por teclado. Para ello en la función keyboardFunc(unsigned char key, int x, int y) pondremos lo siguiente:

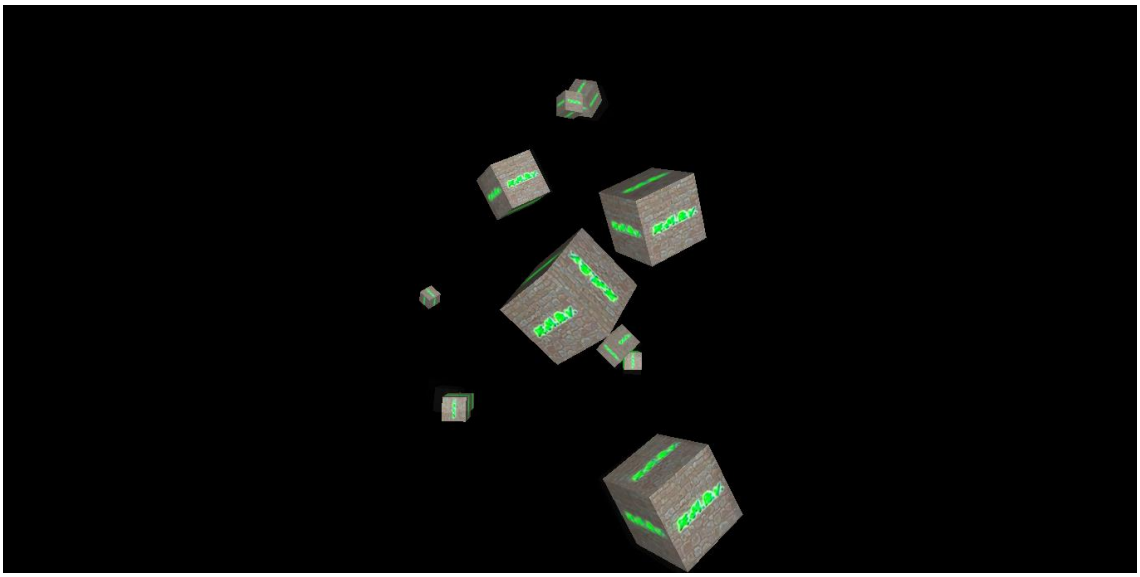
```
void keyboardFunc(unsigned char key, int x, int y){
if (key == 'q' && blurIntensity < 0.4f) blurIntensity += 0.1f;
if (key == 'w' && blurIntensity > -0.4f) blurIntensity -= 0.1f;
if (blurIntensity > 0.4f) blurIntensity = 0.4f;
if (blurIntensity < -0.4f) blurIntensity = -0.4f;
std::cout << blurIntensity << std::endl;
}
```

   La función keyboardFunc recibe la tecla pulsada y las coordenadas del puntero en el momento en el que se pulso la tecla. De esta forma indicaremos (en la segunda línea de código) que si esta pulsada la tecla q y la intensidad del motion blur (blurIntensity) es inferior a 0.4 se aumenta blurIntensity en 0.1, de esta forma acotamos los valores que puede tomar esta variable, para asegurarnos de que la intensidad del motion blur no supera 0.4 tenemos la 4ª línea de código la cual nos indica que si toma un valor superior a 0.4 el valor de blurIntensity es 0.4. En la segunda línea de código indicamos que si esta pulsada la tecla w y la intensidad del motion blur es mayor que -0.4 esta disminuya 0.1, al igual que antes, acotamos la región de valores que puede tomar el motion blur y para asegurarnos de esto tenemos la 5ª línea de código que nos indica que si blurIntensity es menor que -0.4 entonces su valor es -0.4. La última línea de código nos muestra por pantalla cada vez que pulsamos una tecla el valor de blurIntensity.

   De esta forma quedaría implementado el control del Motion Blur por teclado.



blurIntensity = 0.4



blurIntensity = -0.4

2. **Controla los parámetros del DOF por teclado (distancia focal y distancia de desenfoque máximo)**

Para controlar los parámetros del DOF (depth of field) por teclado tenemos que cambiar las variables del postProcessing.v1.frag focalDistance y maxDistanceFactor por variables uniform para poder controlarlas desde el programa principal (main.cpp)
Para ello lo primero que haces es cambiarlas en el archivo postProcessing.v1.frag:

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

Una vez definidas las nuevas variables uniform, pasamos a implementar su control en el programa principal. Lo primero que tenemos que hacer es declararlas e inicializarlas:

```
int uFocalDistance;
int uDistanceFactor;
float focalDistance = -25.0f;
float maxDistanceFactor = 5.0f;
```

uFocalDistance y uDistanceFactor serán nuestras dos variables uniform con las cuales cambiaremos su valor en el shader de postproceso de fragmentos, mientras que focalDistance y maxDistanceFactor serán nuestras variables auxiliares para trabajar con ellas dentro del main. Para ello asignaremos el valor de focalDistance y maxDistanceFactor a uFocalDistance y uDistanceFactor respectivamente, para ello en initShaderPP (inicialización del shader de postproceso) indicamos la localización de las variables unifom:

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

uDistanceFactor = glGetUniformLocation(postProccesProgram,
"maxDistanceFactor");
```

A continuacion en renderFunc() indicamos que se modifican las nuevas variables uniform del shader:

```
if (uFocalDistance != -1){
	glUniform1f(uFocalDistance, focalDistance);
}
if (uDistanceFactor != -1) {
	glUniform1f(uDistanceFactor, 1.0f / maxDistanceFactor);
}

if (uMask != -1) {
	glUniform1fv(uMask, 25, masks[currentMask]);
}
```

Por ultimo solo nos queda controlar las dos variables por teclado, para ello, en keyboardFunc() pondremos el siguiente fragmento de código:

```
if (key == 'q') focalDistance += 1;
if (key == 'w') focalDistance -= 1;
if (key == 'a') maxDistanceFactor += 1;
if (key == 's') maxDistanceFactor -= 1;
```

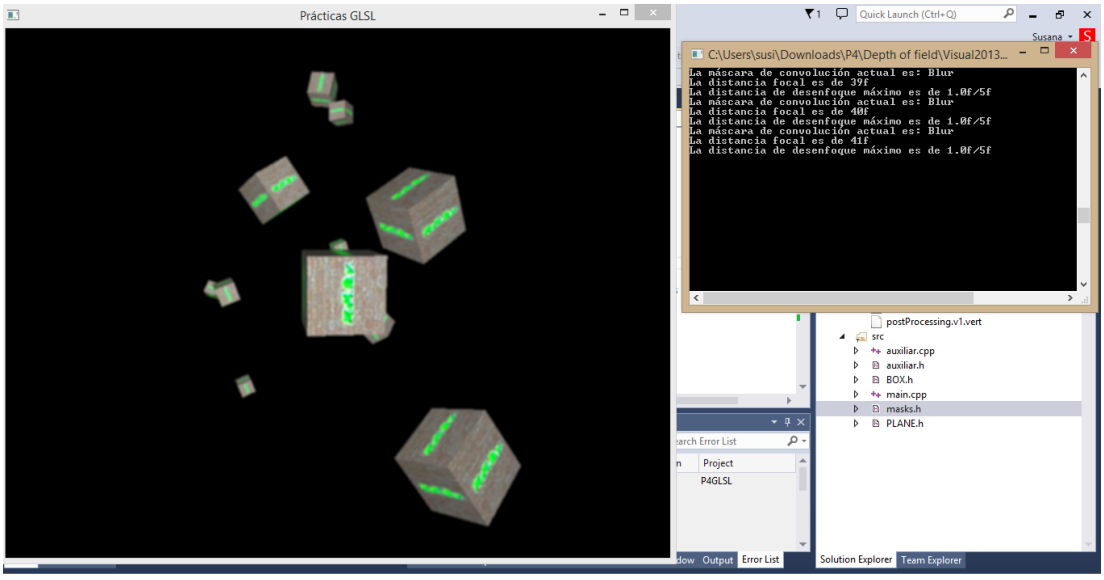
De esta forma indicamos que si esta pulsada la tecla q se aumentará la distancia focal y si esta pulsada la tecla w disminuirá. Indicaremos también el control del desenfoque máximo mediante las teclas a y s las cuales lo aumentaran y disminuirán respectivamente.
Por otra parte hemos añadido en la función KeyboardFunc () unas líneas de código que nos muestran por pantalla la distancia focal y la distancia de desenfoque máximo en cada momento.

```
std::cout << "La distancia focal es de " << focalDistance << "f" <<
std::endl;

std::cout << "La distancia de desenfoque máximo es de 1.0f/" <<
maxDistanceFactor << "f" << std::endl;
```

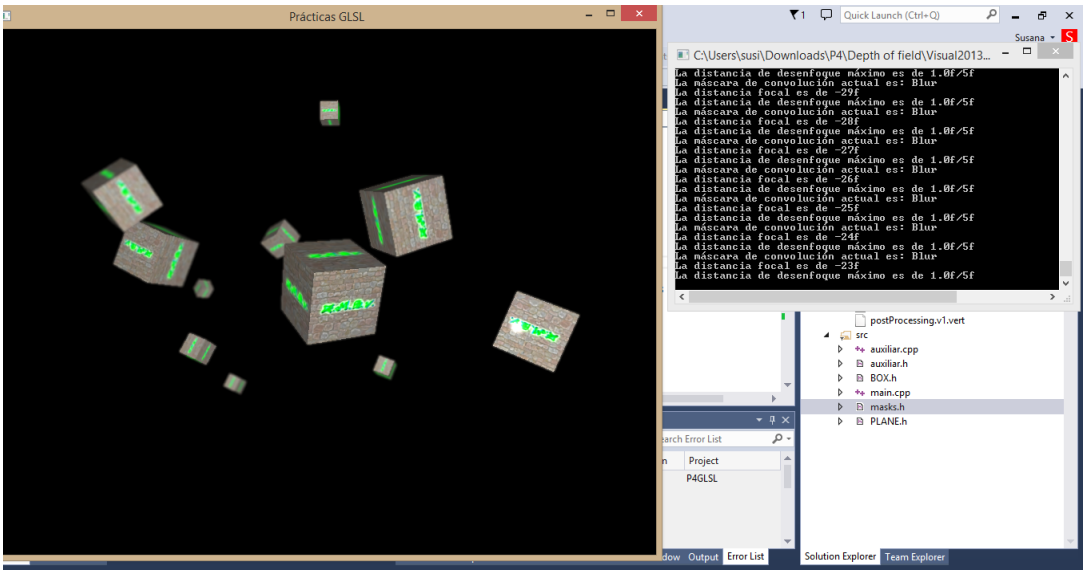
De esta misma forma hemos añadido en main() unas salidas por pantalla que nos indican los controles básicos de la aplicación:

```
std::cout << "Controles básicos:" << std::endl;

std::cout << "Usa Q y W para modificar la distancia focal (Q ampliar, W
reducir)" << std::endl;

std::cout << "Usa A y S para modificar la distancia de desenfoque máxima
(A ampliar, S reducir)" << std::endl;
```

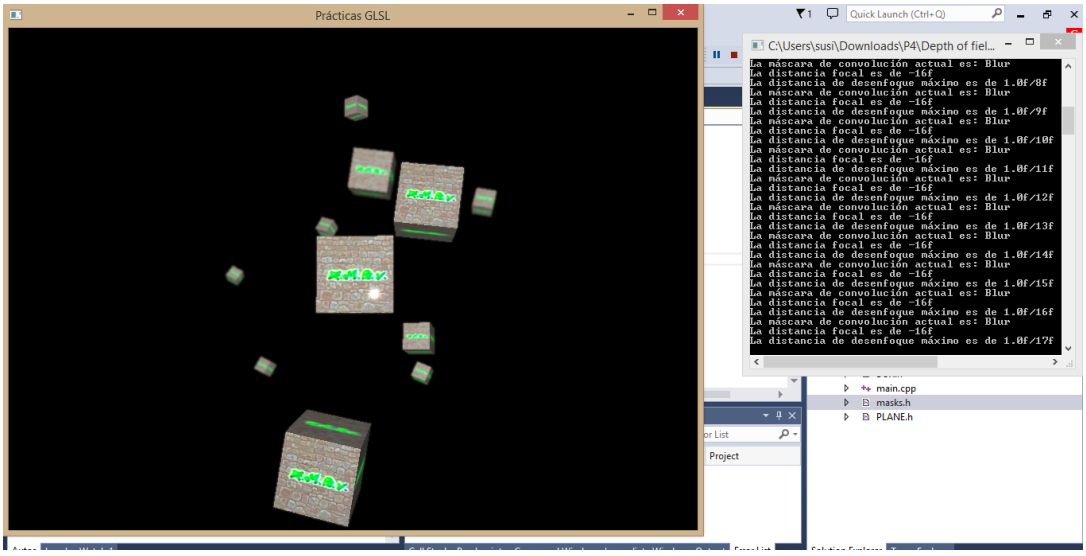
De esta forma quedaría implementado el control de la distancia focal y la distancia de desenfoque máximo por teclado.



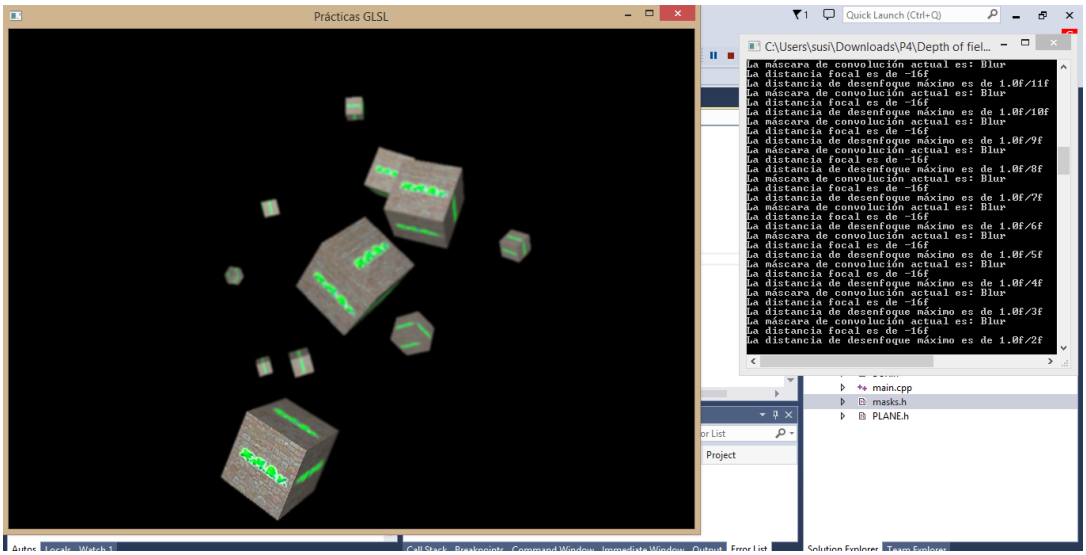
Aumento de la Distancia Focal



Disminución de la Distancia Focal



Aumento del desenfoque máximo



Disminución del Desenfoque Máximo

3. Utiliza el buffer de profundidad para controlar el DOF

Para controlar el DOF mediante un buffer de profundidad tenemos que cambiar la forma mediante la cual se calcula en el shader de fragmentos del postproceso.
Es decir, tenemos que cambiarlo por:

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

Para poder calcular depthTex contamos con la fórmula:

$$Zc = \frac{-near * far}{(zb * (near - far) + far)}$$

De esta forma pasamos a tener la siguiente línea de código que será la encargada de controlar el DOF mediante un buffer de profundidad:

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

Para implementar esta linea de codigo necesitamos 3 variables uniform nuevas depthTex (que nos indicará la profundidad pasada desde el programa principal), variable_near (con la cual obtendremos el valor del plano near desde el programa principal) y la varible_far(nos proporcionará el valor del plano far)

Lo primero que hacemos es declarar estas nuevas variables en el shader de postproceso de fragmentos:

```
uniform sampler2D depthTex;
uniform float variable_near;
uniform float variable_far;
```

También podemos cambiar la forma de calcular el DOF en el main() del shader:

```
void main()
{
(…)
//ORIGINAL
//float dof = abs(texture(vertexTex,texCoord).z -focalDistance)*
maxDistanceFactor;

//APARTADO 3
      float dof = abs((-
variable_near*variable_far/(texture(depthTex,texCoord).r*(variable_near-
variable_far) + variable_far)) - focalDistance) * maxDistanceFactor;

(…)
}
```

Una vez que tenemos las variables uniform que vamos a necesitar declaradas en el shader pasamos a "enlazarlas" con el programa principal. Para ello, al igual que siempre, lo primero que tendremos que hacer es declarar las nuevas variables uniform y sus correspondientes punteros:

```
///buffer
unsigned int depthBufferTexId;
unsigned int uDepthTexPP;
///variables near y far
unsigned int uFar;
unsigned int uNear;
float Far = 50.0;
float Near = 1.0;
```

A continuación pasamos a la función de renderizado, renderFunc(), en la cual indicaremos los valores que van a tomar las variables uniform far y near:

```
float variable_near = Near;
float variable_far = Far;
```

También dentro de la función de renderizado indicaremos lo siguiente:

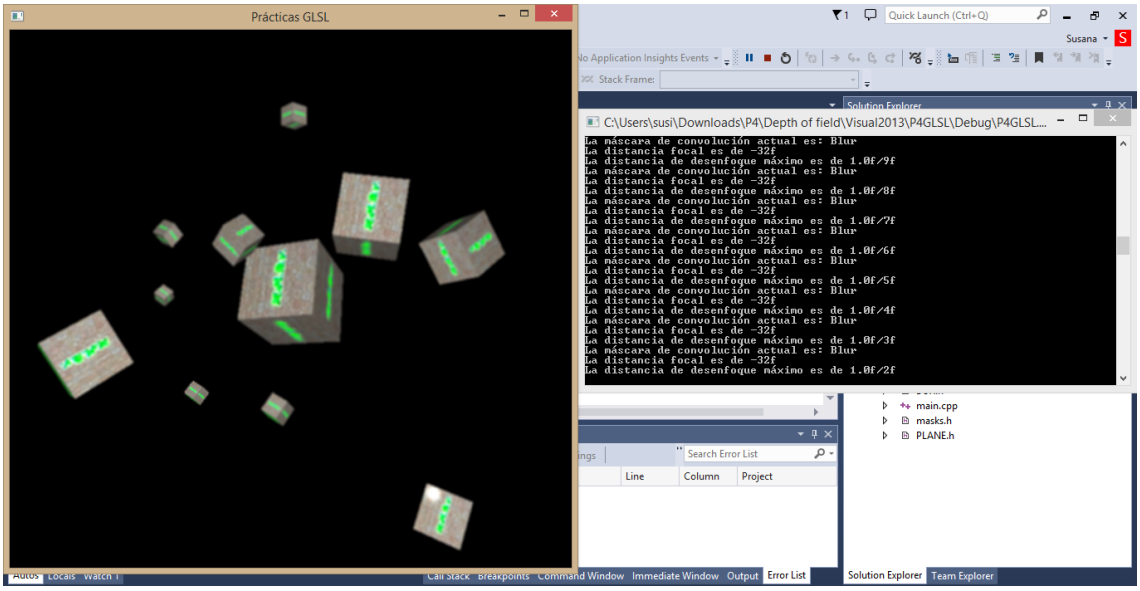
```
if (uFar != -1){
	glUniform1f(uFar, Far);
}
if (uNear != -1){
	glUniform1f(uNear, Near);
}

if (uDepthTexPP != -1)
{
	glActiveTexture(GL_TEXTURE0 + 2);
	glBindTexture(GL_TEXTURE_2D, depthBuffTexId);
	glUniform1i(uDepthTexPP, 2);
}
```

Definiendo así las variables.

Por ultimo en la inicialización del shader de postproceso referenciamos las variables:

```
///buffer
uDepthTexPP = glGetUniformLocation(postProccesProgram, "depthTex");
inPosPP = glGetAttribLocation(postProccesProgram, "inPos");

///far y near
uNear = glGetUniformLocation(postProccesProgram, "variable_near");
uFar = glGetUniformLocation(postProccesProgram, "variable_far");
```

De esta forma quedaría implementado el DOF controlado por un buffer de profundidad.


Prácticas GLSL
Quick Launch (Ctrl+Q)
Susana
No Application Insights Events
Stack Frame:
Solution Explorer
C:\Users\susi\Downloads\P4\Depth of field\Visual2013\P4GLSL\Debug\P4GLSL....
La máscara de convolución actual es: Blur
La distancia focal es de -32f
La distancia de desenfoque máximo es de 1.0f/9f
La máscara de convolución actual es: Blur
La distancia focal es de -32f
La distancia de desenfoque máximo es de 1.0f/8f
La máscara de convolución actual es: Blur
La distancia focal es de -32f
La distancia de desenfoque máximo es de 1.0f/7f
La máscara de convolución actual es: Blur
La distancia focal es de -32f
La distancia de desenfoque máximo es de 1.0f/6f
La máscara de convolución actual es: Blur
La distancia focal es de -32f
La distancia de desenfoque máximo es de 1.0f/5f
La máscara de convolución actual es: Blur
La distancia focal es de -32f
La distancia de desenfoque máximo es de 1.0f/4f
La máscara de convolución actual es: Blur
La distancia focal es de -32f
La distancia de desenfoque máximo es de 1.0f/3f
La máscara de convolución actual es: Blur
La distancia focal es de -32f
La distancia de desenfoque máximo es de 1.0f/2f
main.cpp
masks.h
PLANE.h
Search Error List
Line
Column
Project
Autos
Locals
Watch 1
Call Stack
Breakpoints
Command Window
Immediate Window
Output
Error List
Solution Explorer
Team Explorer

4. **Sube nuevas máscaras de convolución a través de variables uniform. Selecciona entre varias máscaras a través de teclado:**
Para poder controlar las máscaras por teclado lo primero que tenemos que hacer es crear una variable uniform mask que comunique el shader de fragmentos de postproceso con el programa principal.
Para ello, lo primero que tenemos que hacer es declarar la nueva variable uniform en el shader:
Primero definimos el tamaño de la máscara de convolución:

```
const vec2 texIdx[MASK_SIZE] = vec2[](
vec2(-2.0,2.0f), vec2(-1.0,2.0f), vec2(0.0,2.0f), vec2(1.0,2.0f),
vec2(2.0,2.0),vec2(-2.0,1.0f), vec2(-1.0,1.0f), vec2(0.0,1.0f),
vec2(1.0,1.0f), vec2(2.0,1.0),vec2(-2.0,0.0f), vec2(-1.0,0.0f),
vec2(0.0,0.0f), vec2(1.0,0.0f), vec2(2.0,0.0),vec2(-2.0,-1.0f), vec2
(-1.0,-1.0f), vec2(0.0,-1.0f), vec2(1.0,-1.0f), vec2(2.0,-1.0),
vec2(-2.0,-2.0f), vec2(-1.0,-2.0f), vec2(0.0,-2.0f), vec2(1.0,-2.0f),
vec2(2.0,-2.0));

const float maskFactor = float (1.0/65.0);
```

Declaramos la variable uniform mask:

```
uniform 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);
```

A continuación, pasamos a declarar las variables uniform en main.cpp:

```
int uMask;
int currentMask = 0;
```

mediante uMask nos comunicaremos con el shader de postproceso y con currentMask trabajaremos en el programa principal. Para ello asignaremos el valor de currentMask a uMask mediante el siguiente fragmento de código en la inicialización del shader de postproceso (initShaderPP()):

```
uMask = glGetUniformLocation(postProccesProgram, "mask");
```

De esta forma indicamos la localización de las variables uniform.
A continuación en renderFunc() indicamos la nueva variable que vamos a modificar:

```
if (uMask != -1) {
	glUniform1fv(uMask, 25, masks[currentMask]);
}
```

Por ultimo solo nos queda implementar el control por teclado de las máscaras en la función keyboardFunc() mediante el siguiente fragmento:

```
if (key == 'm'){
	currentMask += 1;
	if (currentMask == sizeof(masks) / sizeof(masks[0]))
currentMask=0;
	}
```

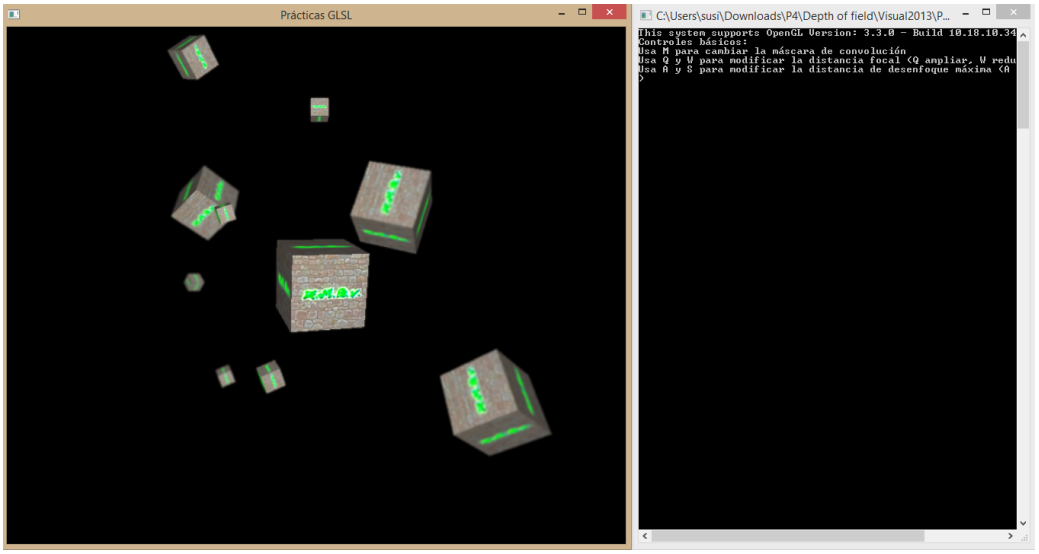
De esta forma si se pulsa la tecla m cambiamos de máscara y si ya no hay mas mascaras en el archivo mask.h, creado para almacenar las máscaras de convolución, se vuelve a la primera máscara.
Al igual que en el apartado anterior añadimos una salida por pantalla en keyboardFunc() que nos indica la máscara en la que nos encontramos en cada momento.

```
std::cout << "La máscara de convolución actual es: " <<
maskNames[currentMask] << std::endl;
```

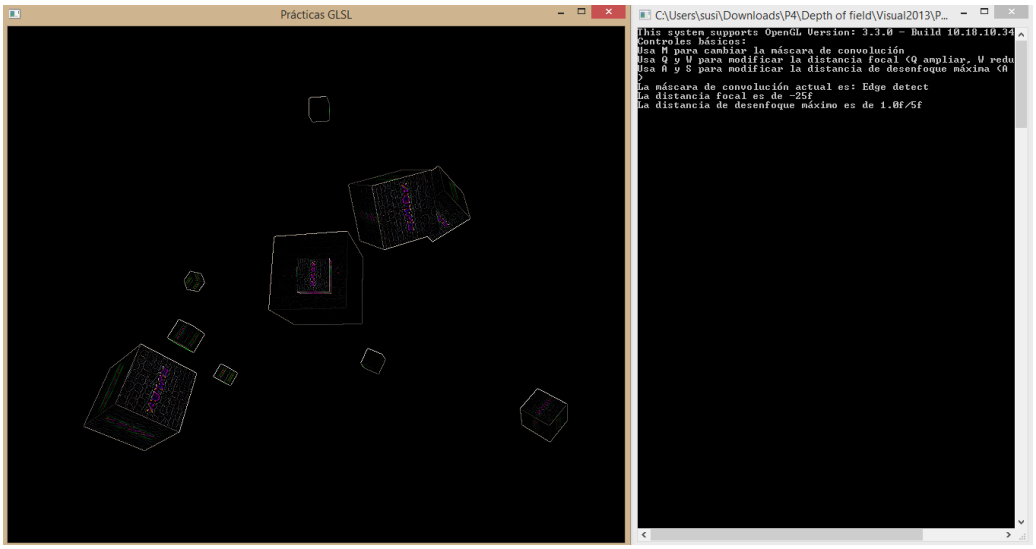
De la misma forma añadimos una salida por pantalla en main() para indicar el control de la máscara de convolución:

```
std::cout << "Usa M para cambiar la máscara de convolución" <<
std::endl;
```

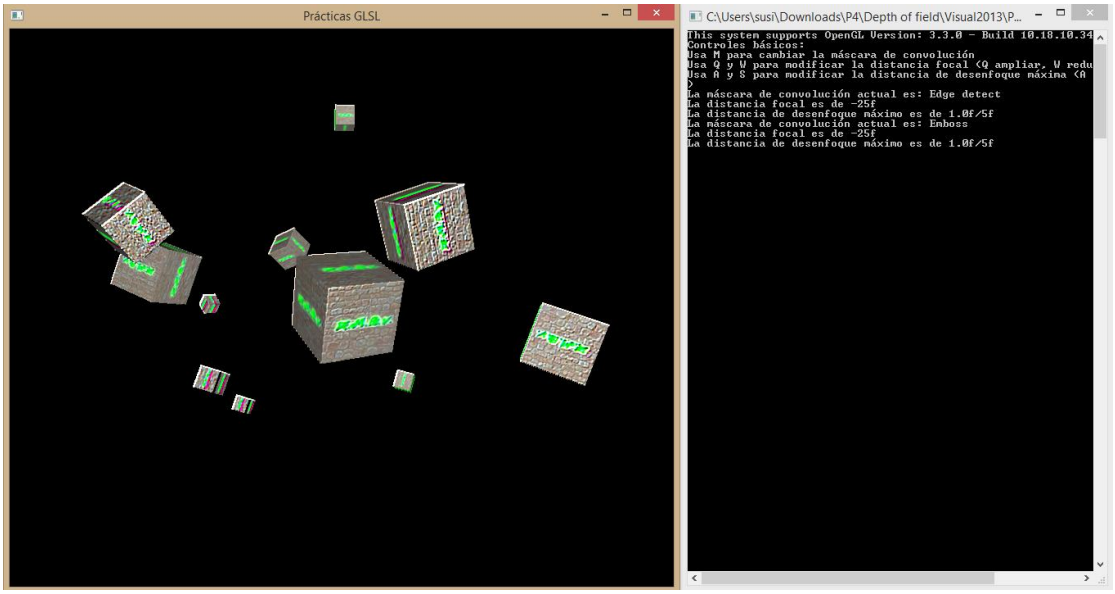
De esta forma quedaría implementado el control de las máscaras de convolución por teclado.



Máscara de convolución blur



Máscara de convolución Edge Detect



Máscara de convolución Emboss