

# Jugando con Shaders en WebGL

# ¿Quién soy?

Felipe Alfonso

*@bitnenfer*

# ¿Quién soy?

Felipe Alfonso  
*@bitnenfer*

```
180 LoadDMAWait::
181     ; Copy Instruction into high ram ($FF80)
182     ld bc, ShadowOAM_StartDMA
183     ld hl, $FF80
184     ld d, (ShadowOAM_EndDMA - ShadowOAM_StartDMA)
185 .LoadLoop:
186     ld a, [bc]
187     ld [hl], a
188     inc bc
189     inc hl
190     dec d
191     ld a, d
192     cp $00
193     jr nz, .LoadLoop
194     ret
195
```

Programador de  
Videojuegos

# ¿Quién soy?

Felipe Alfonso  
*@bitnenfer*

```
180 LoadDMAWait::  
181     ; Copy Instruction into high ram ($FF80)  
182     ld bc, ShadowOAM_StartDMA  
183     ld hl, $FF80  
184     ld d, (ShadowOAM_EndDMA - ShadowOAM_StartDMA)  
185 .LoadLoop:  
186     ld a, [bc]  
187     ld [hl], a  
188     inc bc  
189     inc hl  
190     dec d  
191     ld a, d  
192     cp $00  
193     jr nz, .LoadLoop  
194     ret  
195
```



Programador de  
Videojuegos

Ilustrador

# ¿Quién soy?

Felipe Alfonso  
*@bitnenfer*

```
180 LoadDMAWait::  
181     ; Copy Instruction into high ram ($FF80)  
182     ld bc, ShadowOAM_StartDMA  
183     ld hl, $FF80  
184     ld d, (ShadowOAM_EndDMA - ShadowOAM_StartDMA)  
185 .LoadLoop:  
186     ld a, [bc]  
187     ld [hl], a  
188     inc bc  
189     inc hl  
190     dec d  
191     ld a, d  
192     cp $00  
193     jr nz, .LoadLoop  
194     ret  
195
```

Programador de  
Videojuegos



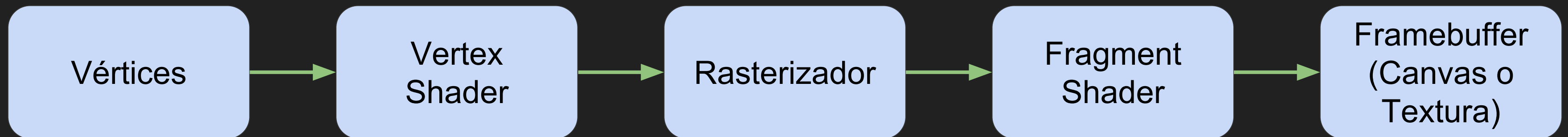
Ilustrador



Phaser v3.0

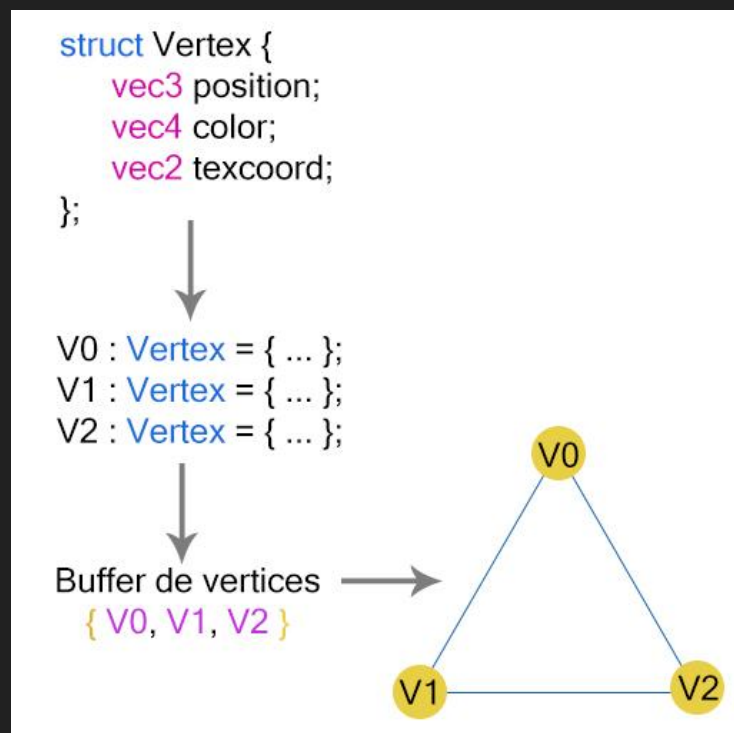
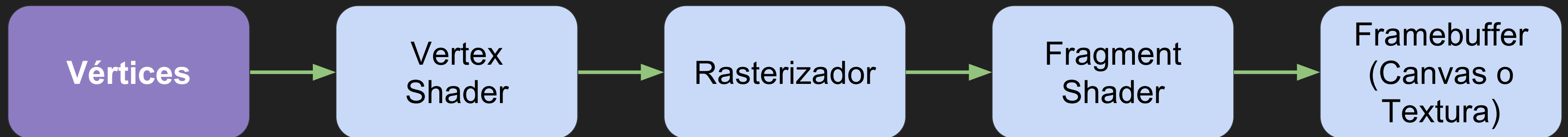
# Resumen

## Pipeline Gráfico de WebGL (simplificado)



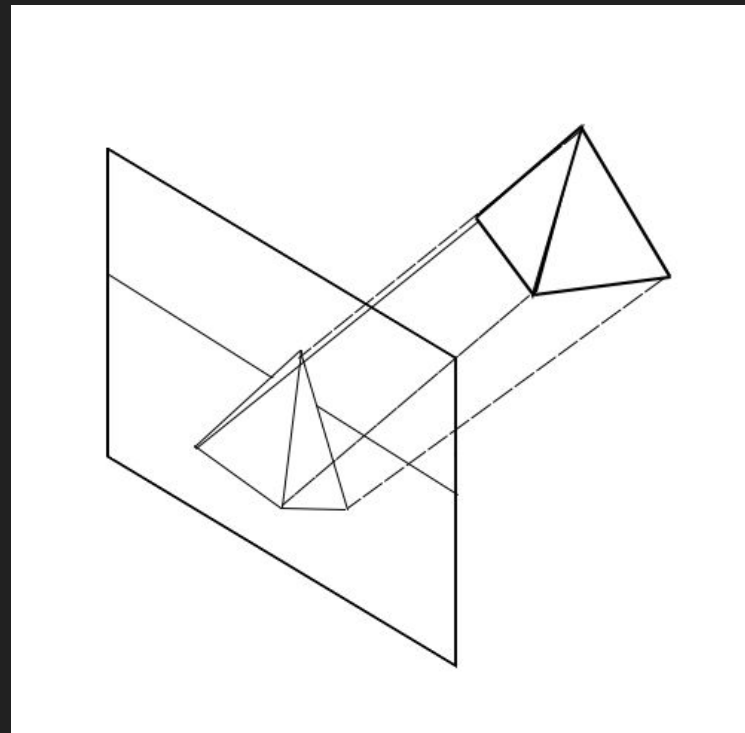
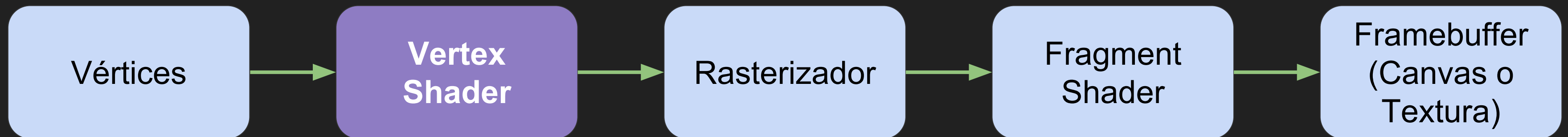
# Resumen

## Pipeline Gráfico de WebGL (simplificado)



# Resumen

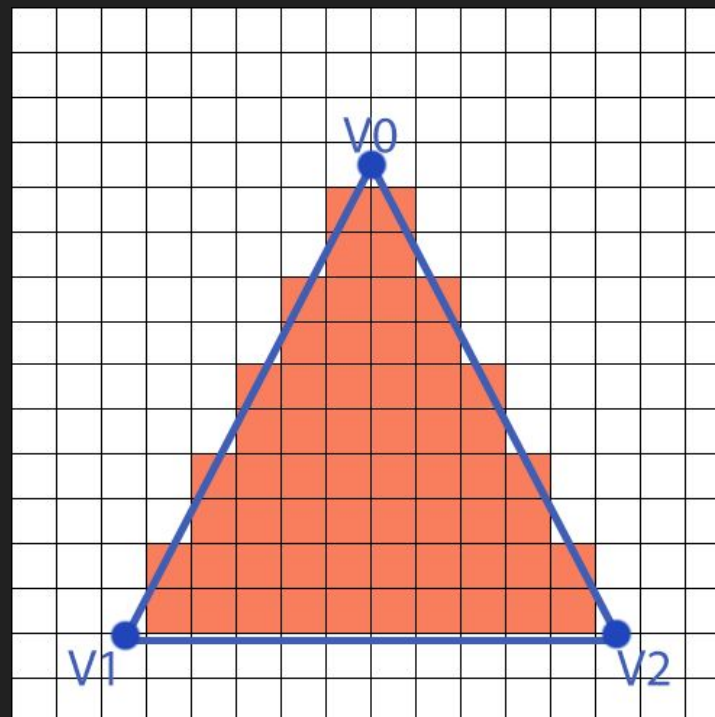
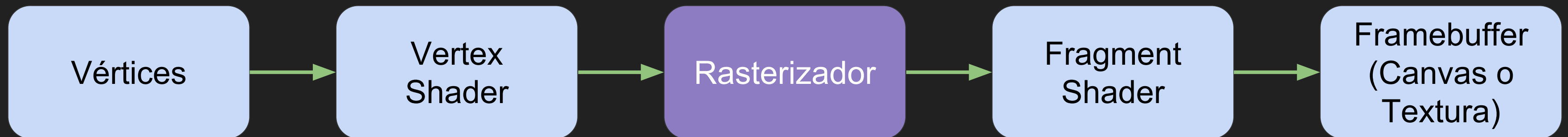
## Pipeline Gráfico de WebGL (simplificado)





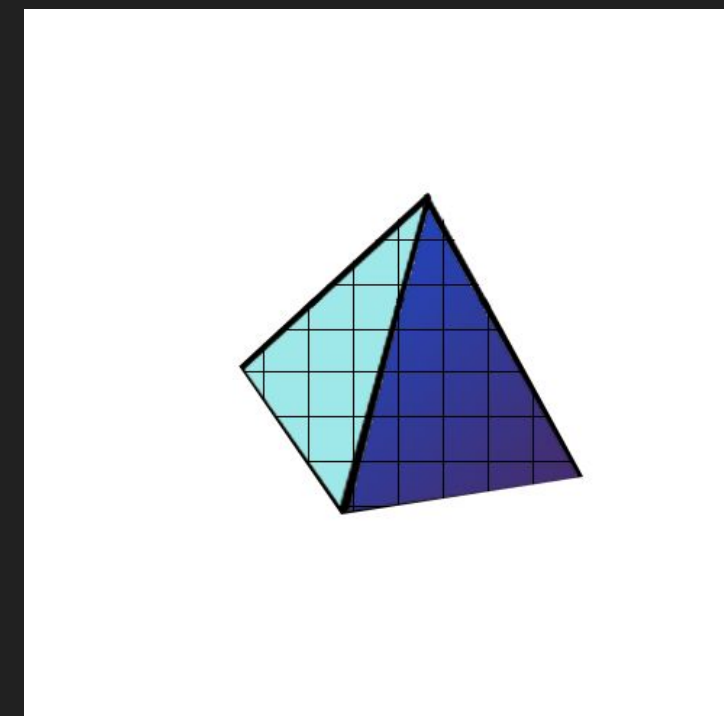
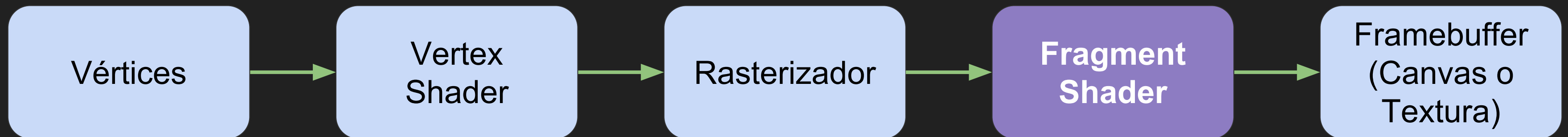
# Resumen

## Pipeline Gráfico de WebGL (simplificado)



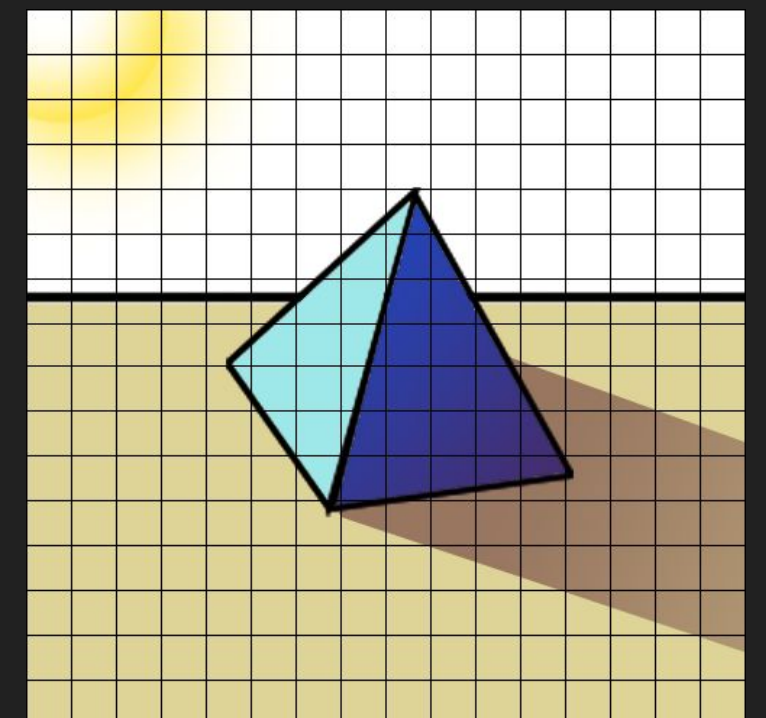
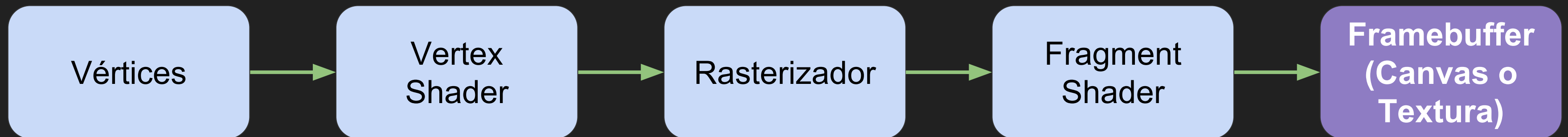
# Resumen

## Pipeline Gráfico de WebGL (simplificado)



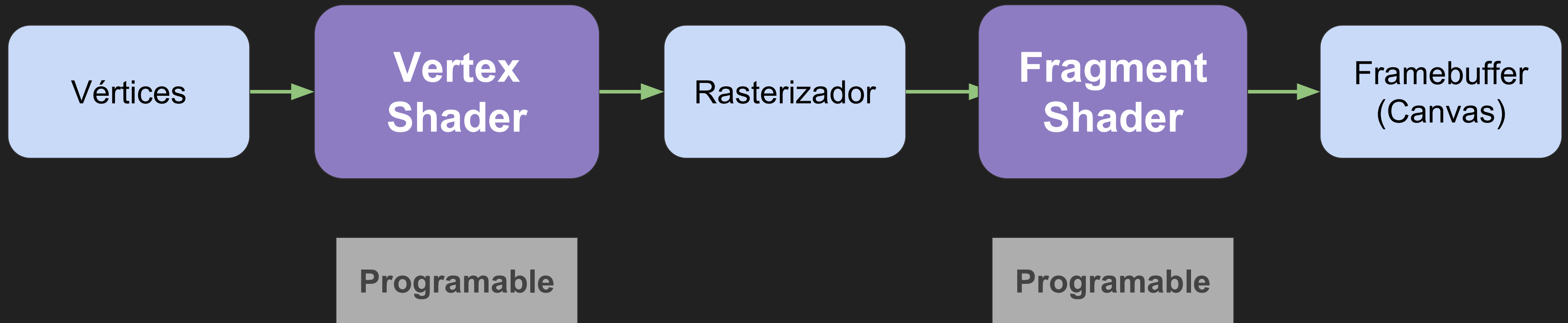
# Resumen

## Pipeline Gráfico de WebGL (simplificado)



# Resumen

Pipeline Gráfico de WebGL (simplificado)



¿Qué son los shaders?

# ¿Qué son los shaders?

- Programas que se ejecutan en la GPU.

# ¿Qué son los shaders?

- Programas que se ejecutan en la GPU.
- Escritos en GLSL (similar a C), compilados por el backend gráfico del navegador.

# ¿Qué son los shaders?

```
1  
2 void main()  
3 {  
4     gl_Position = vec4(0.0);  
5 }  
6  
7
```

Vertex Shader

```
2  
3 void main()  
4 {  
5     gl_FragColor = vec4(1.0);  
6 }  
7
```

Fragment Shader



# Tipos de datos

## Escalar

*bool, int, y float*

# Tipos de datos

## Escalar

*bool, int, y float*

## Vectores

*vec2, vec3, vec4, bvec2, bvec3, bvec4, ivec2, ivec3 y ivec4*

## Matrices

*mat2, mat3 y mat4*

# Tipos de datos

## Escalar

*bool, int, y float*

## Vectores

*vec2, vec3, vec4, bvec2, bvec3, bvec4, ivec2, ivec3 y ivec4*

## Matrices

*mat2, mat3 y mat4*

## Texturas

*sampler2D y samplerCube*

# Tipos de variables

# Tipos de variables

- *uniform ( VS y FS )*

Variable constante durante la ejecución de ambos shaders.

# Tipos de variables

- *uniform ( VS y FS )*

Variable constante durante la ejecución de ambos shaders.

- *attribute ( sólo VS )*

Variable de entrada al vertex shader. Datos de vértices directos desde la aplicación.

# Tipos de variables

- ***uniform ( VS y FS )***

Variable constante durante la ejecución de ambos shaders.

- ***attribute ( sólo VS )***

Variable de entrada al vertex shader. Datos de vértices directos desde la aplicación.

- ***varying ( VS y FS )***

Variable de salida del vertex shader y de entrada al fragment shader.  
Canal de comunicación entre ambos shaders.

# Variable predefinidas



# Variable predefinidas

## Vertex Shader:

output: *gl\_Position* (vec4)  
*gl\_PointSize* (float)

## Fragment Shader:

input: *gl\_FragCoord* (vec4)  
*gl\_FrontFacing* (bool)  
*gl\_PointCoord* (vec2)

output: *gl\_FragColor* (vec4)  
*gl\_FragData* (vec4)

# Variable predefinidas

## Vertex Shader:

output: ***gl\_Position*** (vec4)  
*gl\_PointSize* (float)

## Fragment Shader:

input: *gl\_FragCoord* (vec4)  
*gl\_FrontFacing* (bool)  
*gl\_PointCoord* (vec2)

output: ***gl\_FragColor*** (vec4)  
*gl\_FragData* (vec4)

# Bibliografía

- Documentación de GLSL  
<http://www.shaderific.com/glsl/>
- Tutorial de Shaders en WebGL  
<https://thebookofshaders.com/>
- WebGL Cheatsheet  
[https://www.khronos.org/files/webgl/webgl-reference-card-1\\_0.pdf](https://www.khronos.org/files/webgl/webgl-reference-card-1_0.pdf)
- Fragment Shader Workshop (SDF + Raymarching)  
<http://hughsk.io/fragment-foundry/>

# runDemo();

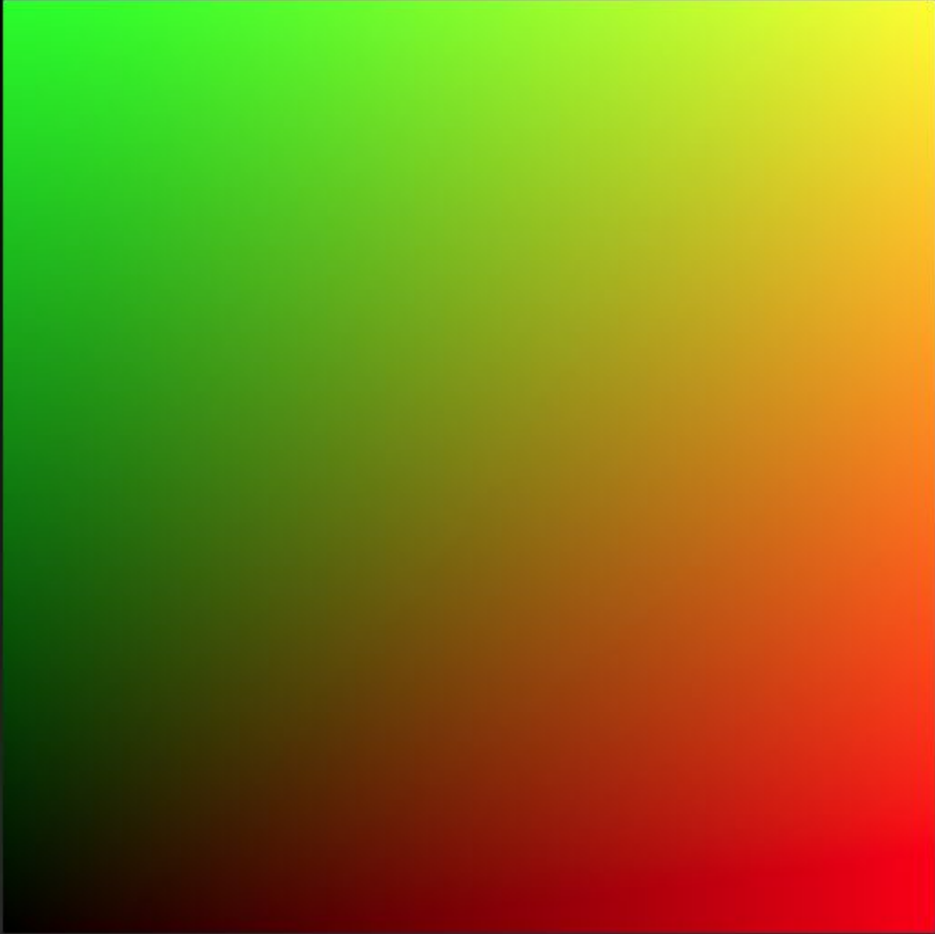
NodersGL - Editor

localhost/noders-gl/#clear

Close Controls

Compilation Time: 1.92ms

frame time: 16.66ms  
avg.: 16.67ms  
music by NNNNNNNNNN



Vert

Frag

```
1 precision mediump float;
2
3 attribute vec3 inPosition;
4
5 void main()
6 {
7     gl_Position = vec4(inPosition, 1.0);
8 }
9
```

Shader

shader

Playground

postprocessId

Convolution Matrix

postprocess

passCount

1

Model

Material

Light Direction

Texture

Background

Media

Editor