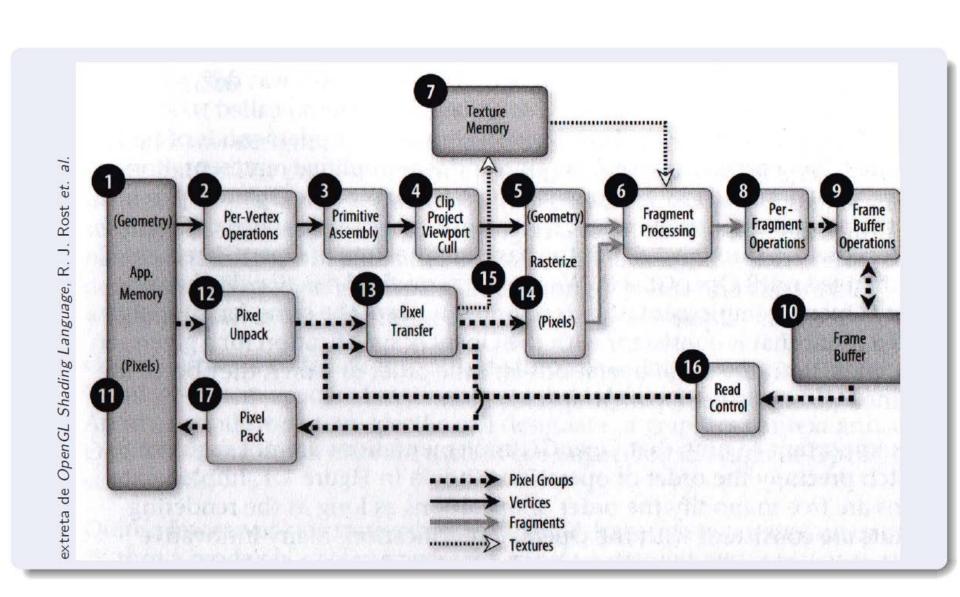
Vertex shaders i Fragment shaders Entorn per desenvolupar shaders (viewer)

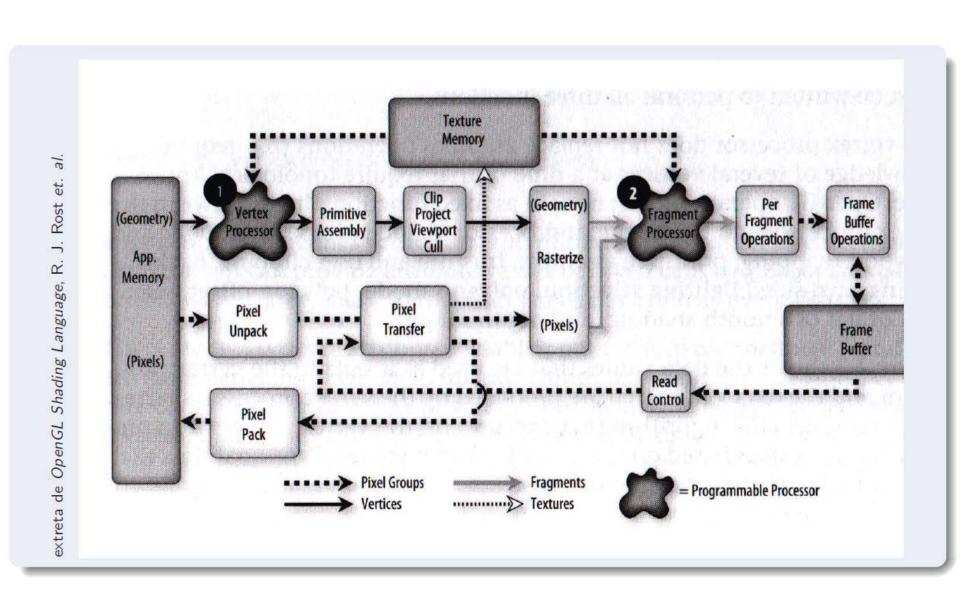
Professors de Gràfics Set 2019

PIPELINE PROGRAMABLE

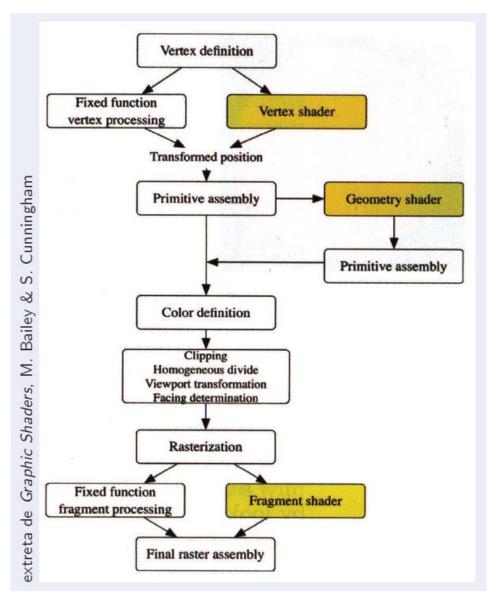
Pipeline fix



Pipeline programable



Pipeline programable (amb GS)



Llenguatges de programació shaders

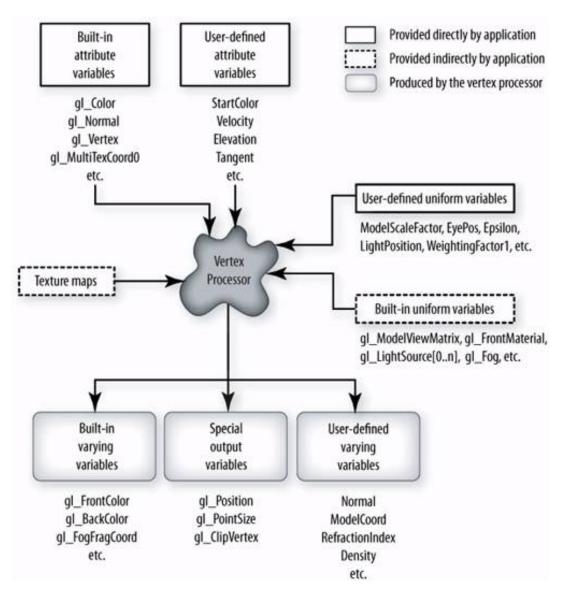
- Cg (C per gràfics) Llenguatge desenvolupat per Nvidia. Col·laboració amb Microsoft. Basat en C.
- HLSL (*High-Level Shader Language*) Llenguatge desenvolupat per Microsoft. Col·laboració amb Nvidia. Basat en C.
- GLSL (GL Shader Language Llenguatge estandaritzat pel OpenGL Architecture Board a partir del release 2.0.

Versions

Versions

Versió	Vers. OGL	data	incorpora
1.10	2.0	2004	vertex i fragment shaders
1.20	2.1	2006	
1.30	3.0	2008	Core and Compatibility
			profiles, in, out, inout
1.40	3.1	2009	
1.50	3.2	2009	geometry shaders
3.3		2010	
4.0		2010	tesselation shaders
	4.3	2012	compute shaders

Vertex shader (compatibility)



Vertex shader (3.3 core)

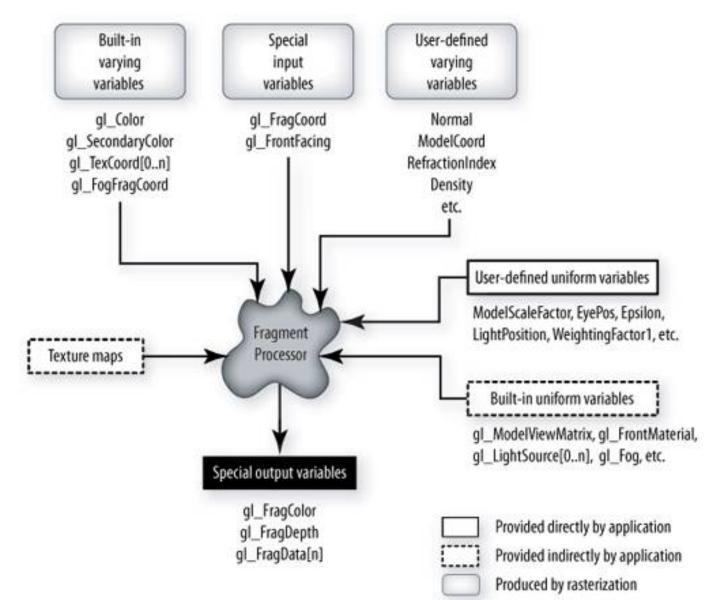
Attributes (user-defined)

```
vec3 vertex; // object space
vec3 normal;
vec3 color;
vec2 texCoord; ...
                                 Uniforms (user-defined, read-only)
                                mat4 modelViewMatrix;
                     Vertex
 Texture maps
                    Processor
                                mat3 normalMatrix;
                                vec4 lightAmbient;
```

vec4 gl_Position; // predefinit; usualment en clip space
vec4 frontColor;

Outputs

Fragment shader (compatibility)

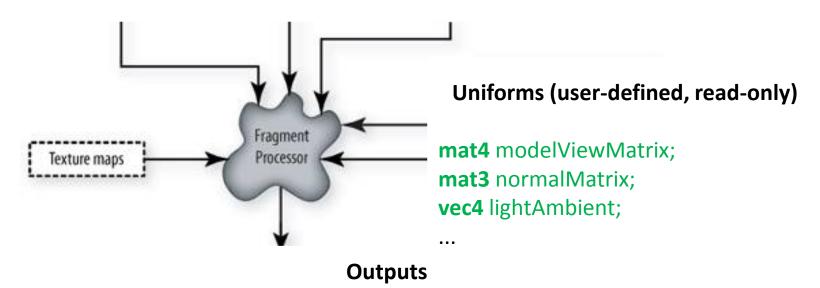


Fragment shader (3.3 core)

Inputs

vec4 gl_FragCoord; // window space
bool gl_FrontFacing;

vec4 frontColor; ...

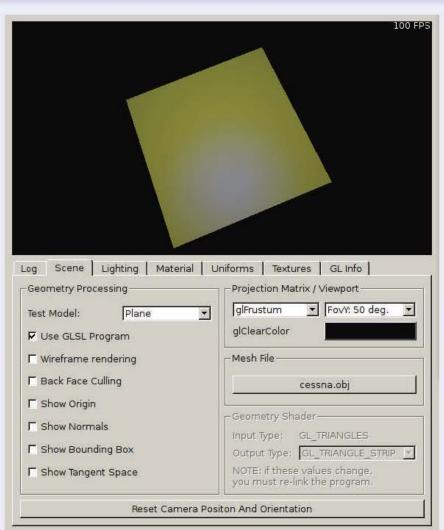


float gl_FragDepth; // z in window space

vec4 fragColor;

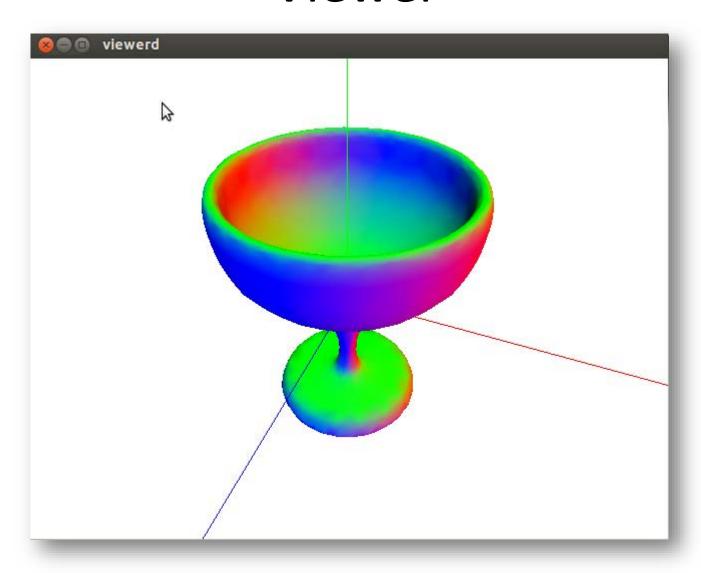
ShaderMaker

Exemple d'una plataforma per a experimentar



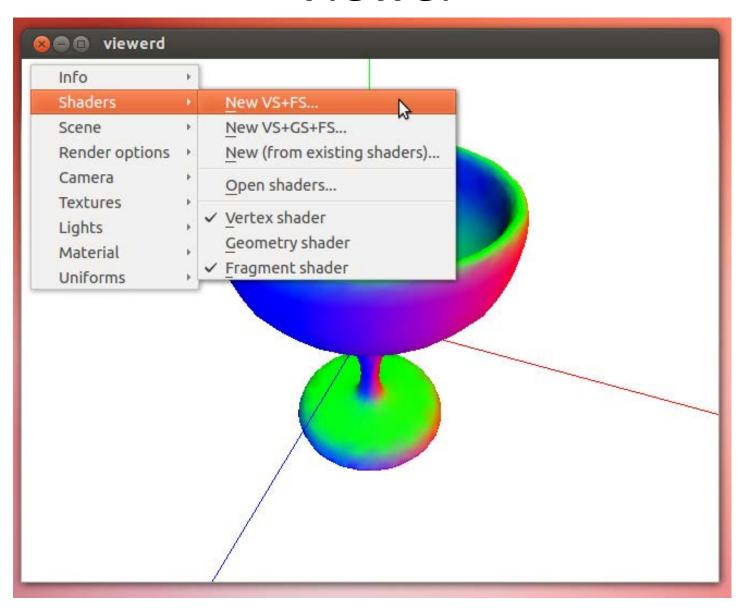
No permet provar shaders en core profile

```
File Edit Shader View ?
 Vertex Shader
             Fragment Shader
  /* lighting.frag - per fragment lighting */
// switch between vertex and fragment lighting.
 uniform bool disablePerFragmentLighting;
// use toon sahding
 uniform bool useToonShading;
 // wether the eye is located in the origin (true
 // or at (0,0, +infinity) (false)
 uniform bool eyeAtOrigin;
 // input
 varying vec3 normal; // fragment normal in eye s
 varying vec3 position; // fragment position in a
Attach to program
                             Compile and Link (F5)
```



- Funciona en linux32 i linux64
- És recomanable crear una carpeta amb els shaders que anireu creant:
 - mkdir shaders (on vulgueu)
 - cd shaders/
 - /assig/grau-g/viewer

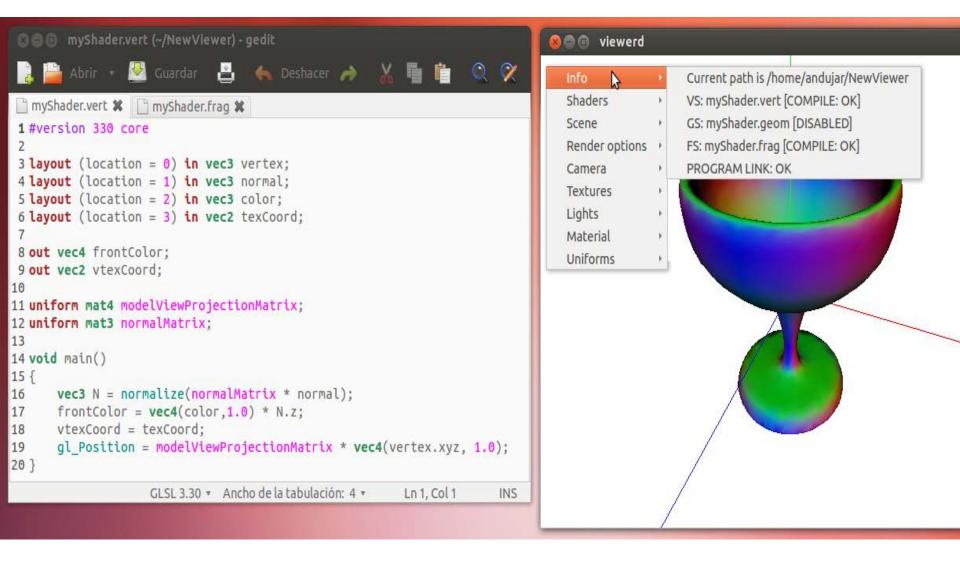
Premeu [SPACE] per accedir al memu



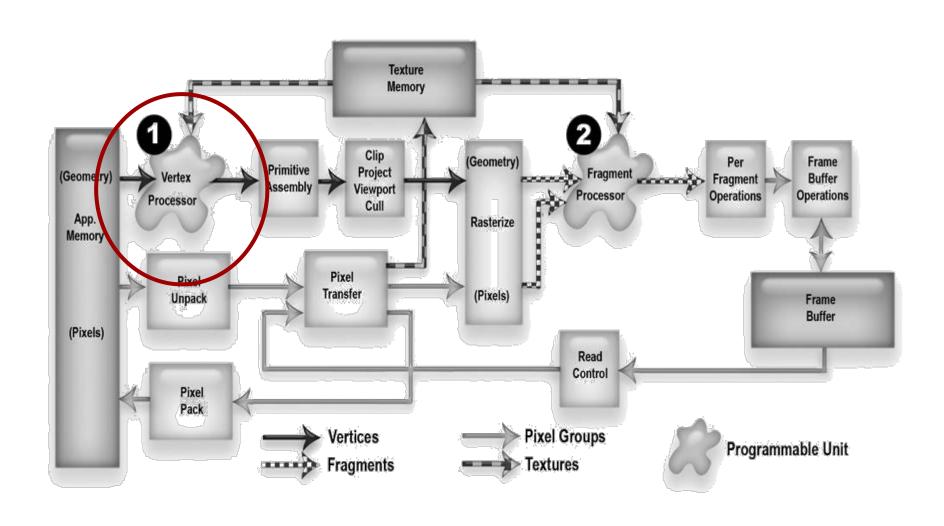
Workflow per resoldre cada exercici:

- cd shaders/
- /assig/grau-g/viewer
- Crear/obrir els shaders:
 - Crear shaders des de l'inici:
 Shaders → New VS+FS... (usa una plantilla per defecte).
 - Crear els shaders basant-se en shaders existents:
 Shaders → New (from existing shaders)...
 - Obrir shaders existents:
 Shaders → Open shaders...

Cada cop que guardeu un shader, es carregarà automàticament



VERTEX SHADERS



Vertex shader (3.3 core)

Attributes (user-defined)

```
vec3 vertex; // object space
vec3 normal;
vec3 color;
vec2 texCoord; ...
                                 Uniforms (user-defined, read-only)
                               mat4 modelViewMatrix;
                    Vertex
 Texture maps
                    Processor
                               mat3 normalMatrix;
                               vec4 lightAmbient;
                  Outputs
```

vec4 gl_Position; // predefinit; usualment en clip space
vec4 frontColor;

- Attribute variables: són variables que representen els atributs d'un vèrtex. Poden canviar de valor per cada vèrtex d'una mateixa primitiva. Pel VS són d'entrada.
- Attributes definits pel viewer (cal declarar-los):

```
layout (location = 0) in vec3 vertex; // similar a gl_Vertex (però 3D) layout (location = 1) in vec3 normal; // idèntic a gl_Normal layout (location = 2) in vec3 color; // similar a gl_Color (però RGB) layout (location = 3) in vec2 texCoord; // similar a gl_MultiTexCoord0
```

 Uniform variables: són variables que canvien amb poca freqüència. Com a molt poden canviar un cop per cada primitiva (però no pas per cada vèrtex de la primitiva).

Variables uniform que envia el viewer (cal declarar-les)

```
uniform mat4 modelMatrix;
uniform mat4 viewMatrix;
uniform mat4 projectionMatrix;
uniform mat4 modelViewMatrix;
uniform mat4 modelViewProjectionMatrix;
```

```
uniform mat4 modelMatrixInverse;
uniform mat4 viewMatrixInverse;
uniform mat4 projectionMatrixInverse;
uniform mat4 modelViewMatrixInverse;
uniform mat4 modelViewProjectionMatrixInverse;
```

uniform mat3 normalMatrix;

Variables uniform que envia el viewer:

```
uniform vec4 lightAmbient;
uniform vec4 lightDiffuse;
uniform vec4 lightSpecular;
uniform vec4 lightPosition;
```

uniform vec4 matAmbient;

uniform vec4 matDiffuse;

uniform vec4 matSpecular;

uniform float matShininess;

```
// (sempre estarà en eye space)

// similar a gl_FrontMaterial.ambient
// similar a gl_FrontMaterial.diffuse
// similar a gl_FrontMaterial.specular
// similar a gl_FrontMaterial.shininess
```

// similar a gl_LightSource[0].ambient

// similar a gl_LightSource[0].specular

// similar a gl LightSource[0].position

// similar a gl_LightSource[0].diffuse

Variables uniform que envia el viewer:

Output variables:

- out vec4 gl_Position (predeclarada)
- Variables "varying": el VS les passa al FS
 - Pel VS són de sortida.
 - Pel FS són d'entrada, i <u>es calculen per interpolació</u>.
 - Exemples típics (depenen de l'aplicació): color, normal, coordenades del vèrtex, coordenades de textura...

Un vertex shader sempre ha d'escriure a

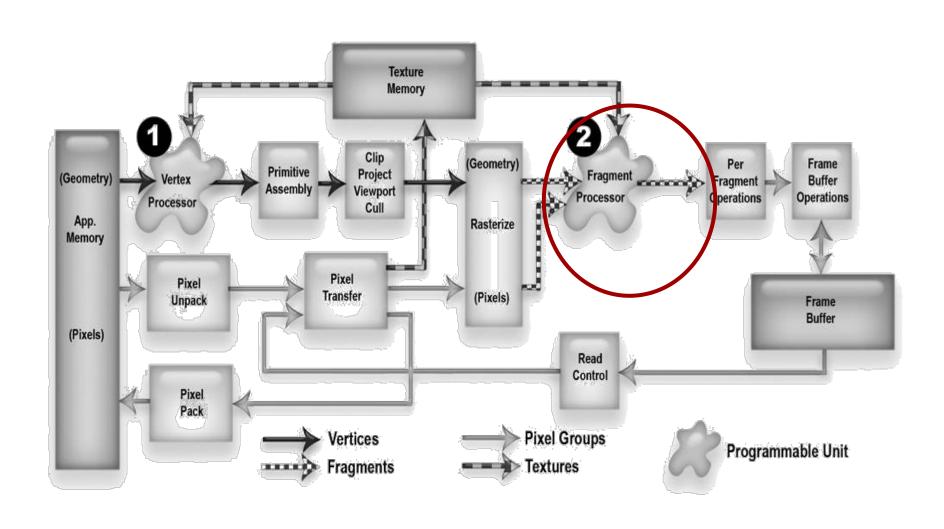
vec4 gl_Position

(usualment les coordenades del vèrtex en clip space).

Normalment ho farà multiplicant el vèrtex per la matriu modelViewProjectionMatrix.

- El VS s'executa per cada vèrtex que s'envia a OpenGL.
- Les tasques habituals d'un VS són:
 - Transformar el vèrtex (object space → clip space)
 - Transformar i normalitzar la normal (eye space)
 - Calcular la il·luminació del vèrtex
 - Generar o passar les coords de textura pel vèrtex

FRAGMENT SHADERS

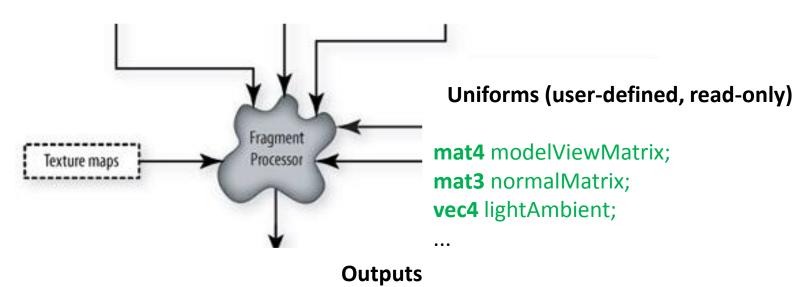


Fragment shader (3.3 core)

Inputs

vec4 gl_FragCoord; // window space
bool gl_FrontFacing;

vec4 frontColor; ...



float gl_FragDepth; // z in window space

vec4 fragColor;

Special input variables: calculats per OpenGL de forma automàtica; es poden llegir al fragment shader:

```
vec4 gl_FragCoord; // coordenades del fragment (window space)
bool gl_FrontFacing; // true si el fragment és d'un polígon frontface
```

- Varying variables: són variables que es calculen al vertex shader i arriben interpolades al fragment shader.
- Exemple (core profile):

in vec4 frontColor;

Output variables:

• Predefinides:

float gl_FragDepth // depth final del fragment (pel z-buffer)

Definides per l'usuari:

out vec4 **fragColor** // color del fragment

- Un fragment shader s'executa per cada fragment que produeix cada primitiva.
- Les tasques habituals d'un fragment shader són:
 - Accedir a textura
 - Incorporar el color de la textura
 - Incorporar efectes a nivell de fragment (ex. boira).
- I el que no pot fer un fragment shader:
 - Canviar les coordenades del fragment (sí pot canviar gl_FragDepth)
 - Accedir a informació d'altres fragments (tret de dFdx, dFdy)

EXEMPLES

VS per defecte al viewer

```
#version 330 core
layout (location = 0) in vec3 vertex;
layout (location = 1) in vec3 normal;
layout (location = 2) in vec3 color;
layout (location = 3) in vec2 texCoord;
out vec4 frontColor;
out vec2 vtexCoord;
uniform mat4 modelViewProjectionMatrix;
uniform mat3 normalMatrix;
void main() {
 vec3 N = normalize(normalMatrix * normal);
 gl_Position = modelViewProjectionMatrix * vec4(vertex.xyz, 1.0);
 frontColor = vec4(color, 1.0) * N.z;
 vtexCoord = texCoord; }
```

FS per defecte al viewer

```
#version 330 core
in vec4 frontColor;
out vec4 fragColor;
void main()
     fragColor = frontColor;
```

LLENGUATGE GLSL

Elements del llenguatge GLSL

Tipus bàsics

```
Escalars
int, float, bool
```

```
Vectorials
vec2, vec3, vec4, mat2, mat3, mat4, ivec3, bvec4,...
```

```
Constructors

Hi ha arrays: mat2 mats[3];
i també structs:

1    struct light{
2    vec3 color;
3    vec3 pos;
4  };

que defineixen implícitament constructors: light l1(col,p);
```

Elements del llenguatge GLSL

Functions

N'hi ha moltes, especialment en les àrees que poden interessar quan tractem geometria o volem dibuixar. Per exemple, radians(), degrees(), sin(), cos(), tan(), asin(), acos(), atan() (amb un o amb dos paràmetres), pow(), log(), exp(), abs(), sign(), floor(), min(), max(), length(), distance(), dot(), cross(), normalize(), noise1(), noise2(), ...

OpenGL Quick Reference card

https://www.khronos.org/files/opengl-quick-reference-card.pdf

	hading Language 1.50	_								
The OpenGL= Shading Language is several closely- related languages which are used to create shaders for each of the programmable processors contained in the		Preprocessor	Preprocessor (2.2) Preprocessor Operators			Preprocessor Directives Each number sign (#) can be preceded in its line only by space or horizontal tabs.				
OpenGL processing pipeline. [n.v.n] and [Table n.n] refer to sections and tables in the		Preprocessor operators follow C++ standards. Preprocessor expressions are exaluated according to the behavior of the host processor, not the processor targeted by the shades.			5710 5710	er n	clor	Fundef Fig. 60 Fiveralan Fi	r ep knoff for los	
specification at www. Content shown in blue	opengl.org/registry Its removed from the OpenGL 3.2:	Frenzion 150		Wender 150" k	epire (belupe	shaders us	ingversion 1.50	of the language	. Fversion	
core profile and present only in the OpenGL3.3 compatibility profile.		Westion 150 compatibility must occur in a silicon patibility in compatibility in the compatib			thader before anything also other than white space or comments. Use a scene feature in the compatibility profile.					
					uire, emittle, warm, distrile me: the extension supported by the compiler, or "all"					
Types (4.1.1-4.1.1) tanapatent Types	al .	Predefined N		. someofren	4.00 BIO	enson supp	0780 b) the co	mple; or all		
vold	no function return value									
bool	Spolean	UNE	_RLS Decimal integer of	protents		1585	10N D	ecimal integer, e	g::150	
Int, sint	signed and unsigned integers									
float	floating scalar	_								
vec2, vec3, rec4	Floating point vector	Qualifiers			Interes	eletion ((unlifier (4.)	3.91		
brec2, brec3, brec4	Spolesn vector	Qualifiers						er and inputs to	fragment str	
Ivec2, Ivec2, Ivec3	signed and unsigned integer vector	Storage Qual	Storage Qualifiers [4.2]		smooth perspective correct interpolation					
uveci, uveci, uveci	c2, uvec2, uvec9 Variable d		stans may have one storage qu		flat no interpolation					
mat2, mat3, mat4	3x2, 3x2, 4x4 float matrix	none	(default) local read/hyrbamenory o	Input parameter			heer interpolat			
	3-column float matrix with 2, 2, or 4 rows	const	complied me constant, or read-only	Unction parameter	The foll	bying pred	declared warful	bles can be red	leclared with	
	3-column float matrix with 2, 3, or 4 rows	lo.	Interestrice shader from one-loads		Interpolation qualifier:			el PrentColor		
	4-column float matrix with 2, 2, or 4 rows	oentrold in	Integer room shaper from previous a integer with centrold based intercold		Vertec	autratic	gl_Rror gl_Bad			
Floating-Point Sampler	Tunar (Connua)						gi Pro	xCalor ntSecondaryColo	7	
rost ng-roint sampler ramole11.120	score (Uprepie)	out centrold out	Integrout of a shader to subsequent integration controld based interpole	t stage (popled out)			gl_Red	kSecondaryColor		
					Programt language:		c gl_Calc	gl_Calor		
samplerCube	access cube magped texture	uniform	Integration community CounCl., a	ndithe application			gi_Seco	andaryColor		
sampler20Red:	access rectangular facture				Paner	eter O-e	differa (4.4)			
empler[1,2]DShedow score: 10 or 10 depth texture/comparison		Uniform (4.3)	5]		Input v	alues are so	opled in at fun	notion call time	, putput valu	
sampler20Red5hadow	access rectangular texture/comparison		Use to declare global variables with the same values across the entire primitive being propertied. Uniform variables are		Input values are copied in at function call time, output value are copied out at function return time.					
carcoler11.2 DArray	y access 1D or 20 area texture need		the entire primitive being processed. Uniform veriables are read-only. Doe uniform qualifiers with any basic data types or			(default) car	meas in			
	access 10 or 10 array depth texture/	array of these,	or when declaring a variable wit	case type it a	In			sed into a function		
manhani hirihada sakasana	comparison	structure, e.g.: uniform yead light/Position:			out	for function parameters passed back out of a function, but				
comparison sample-Suffer sections		Uniform sec4	igno-serior;		not initialized for use when passed in					
		Leyout Quality	Leyout Qualifiers (4.2.6)			Input for function parameters passed both into and out of a function				
complex104/5 access 20 multi-comple texture complex104/5Array access 20 multi-complex array texture		Inpout) (ayout-qual(films) black-declaration			Precision and Precision Qualifiers (4.5)					
samplers solidarray	access 20 multi-cample array testure		t-qual(firs) In/out/uniform		Precisio	n qualifiers	have no affect	ction precision;	they ald code	
integer Sampler Types (Cosmuel		t-qualifiers) in/out/uniform de:	denetten			enGLES. They	(carec		
izampler(1,1,2 D	access houser 10, 10, or 20 testure	Input Levout C				p, medium				
IsamplerCube	access integer cube mapped feature	Layout qualifier	Identifiers for geometry shader	Inputs:	Precisio	e qualifiers	precede a floa	eting point or in	tagar dadarat	
Izample:109ect score: https://doi.org/10.rectorgular-texture		points, lines, lines_adjacency, triangles, triangles_adjacency				float color				
			es can have an input layout only				ent sets a defa	suit for subsequ	uent declarati	
izempler[1,1]DArray izemplerGuffer	access integer 30 or 20 array texture	the built-in raris	ble gl_FragCoord with the layour			g Int;				
	acoscs hittiger buffer texture	Identifiers:			Imari	ent.Quali	fiers Exemp artir(all)	ples [4.6]		
lompler20HS	access hitager 20 multi-cample texture	origin_upper	_left, plasi_center_integer		Apragn	a STOGL Inv	ariant(all)	force all outpo	ut rariables to	
lompler20HSAme;	access int. 30 multi-cample array texture	Output I suppl	Output Layout Qualiflers		Invariant at Position:		Invariant modifica mod	loughy declared		
Unalgred Integer Sampl	ler Tipec (Closque)	Layout qualifier	Layout Qualifier Identifiers for geometry chader outputs:		inianant g_rosmon;		rarable			
usampler[1,2,3]D	access unsigned int 10, 30, or 20 testure		drip, triangle_drip,		Invarian	nt centrold o	out vec3 Color;	quality as part	t of a variable	
usamplerCube	accept undered introde macced feature	777	o_vertices - integer-constant				-	declaration		
unimpercuse access unagree in suce mapped secure unimple/20fact access unagreed int rectangular texture		Hallerm West	Laurent Confedera		Order of Qualification [4.7] When multiple qualifications are present, they must follow:					
		Language West	Uniform-Block Layout Qualifiers Layout qualifier Identifiers for uniform blocks:			nultiple qu	allfications an	e present, they	must follow	
usampler[1,2]DArray		Layout qualifier identifiers for uniform blocks: shared, packed, std140, row_major, column_major				rder. This or	rder is as folio	PAGE.		
usamplerBuffer scoess unsigned integer buffer texture		crareo, pacced, stos 40, rovi_major, column_major			invariant, interpolation, storage, precision storage, peremeter, precision					
usempler20MS	access wint 20 multi-cample texture				ato	тада, дага	meter, precis	100		
usampler20MSArray	access sint 20 multi-cample array texture									
mollelt Conventions 14	I others must use constructors)	Oneseter	and Expressions		14		and and the start			
Expression type	Implicitly converted to type				14.	- 11	ogical inclusive:			
int, wint	fact	Operators p	5.1) Numbered in order of pre- quality operators > < < >	cedence. The	15.		election (Select o calect Individual	ts one entire ope ual components	rand, Usernik of vectors.)	
re, une Vec2, unec2	vec1	Boolean, To con	r pare vectors component-vilse,	use functions		**				
vecz, unecz vecă, unecă	vesi	such as lessThan	r[], aqual[], etc.			10 A	exignment			
lyeck useck	veci	1 []	parenthiltical grouping		16.	Page Accessor	of threatis auxilia	nments		
	1.00	- 11	array subscript			La de ja				
		. 0	function call & constructor struct. field or method selector, switcher	-	17.		equence			
aggregation of Basic Ty	280	2 11								
Aggregation of Basic Ty Arrays float(2) foc	pas	4 7	postfix increment and decrement		Vector Components (5.5) In addition to array numeric subscript syntax (e.g.: x(0), x(0)).					
Iggregation of Basic Ty Arrays float [3] foo float too[3]:		1 #=	postfix increment and decrement prefix increment and decrement		Vector	fan in s	annual (aca)	andat mater to	and the same	
Iggregation of Basic Ty Amays fleet(2) foo; fleet to(2) "enutur only 1-d	sc and blocks can be arrays Inwestonal arrays supported	1 #=	poetfix increment and decrement prefix increment and decrement unary.		In addit	fon to any	y numeric sub-			
Iggregation of Basic Ty Amays fleet(2) foo; fleet to(2) "enutur only 1-d		1	positiv increment and decrement prefix increment and decrement unary multiplicative		In addit	fon to array of vector co	y numeric sub emponents en	ecript syntax je e denoted by a d replicated, e.	single letter.	
Agregation of Basic Ty Arrays fleet(3) for fleet to(3) stocked stocked stocked stocked stocked stocked	es and blocks can be arrays Imersional arrays supported a members can be arrays	2 +1 4 */5 5 +-	postfix increment and discrement praffix increment and discrement unary, multiplicative addition bloods with		In addit	fon to array of vector co nents can b ref. Use wh	y numeric sub emponents an se swissled an en accessing ve	e denoted by a	zingle letter. g.: poz.ux, po	
Iggregation of Basic Ty Arrays fleet(3) for fleet to (3) direction direc	es and blocks can be arrays (mensional arrays supported a members can be arrays norms)	2 +1 4 */5 5 +-	postfix increment and discrement praffix increment and discrement unary, multiplicative addition bloods with		In addit	don to arre of westor co nents can b	y numeric sub emponents an se swissled an en accessing ve	e denoted by a direplicated, e,	zingle letter. g.: poz.ux, po	
Agregation of Basic Ty Arrays fleet(3) for fleet to(3) stocked stocked stocked stocked stocked stocked	es and blocks can be arrays financiars) arrays supported a members can be arrays some some ((i) (i) episeus servicia desirables,	++- 2 +1 4 */5 5 + 5 40.00 7 00 00 00	pooffs, increment and placement profits increment and decrement strary multiplicative additive stratus antiferrational residential decalling		In addit names Compo dx y. z.	fon to array of vector co nents can b ref. Use who normals	y numeric sub imponents an a switcled an an accessing ve	e denoted by a direplicated, e,	zingle letter. g.: poz.ax, po ert points or	
Iggregation of Basic Ty Arrays float(3) foo; float foo(3) structure only 1-d structure struct type- members struct type- members struct type-	se and biddiscan be arrays invariant languiscoperad invariant languiscoperad invariant con be arrays invariant invar	++- 1 ++- 4 */5 5 +- 6 *(5) 7 <> 4 %	costs, however, and decrement prefix however, and decrement stary multiplicative addition aboves with readonal according aboves and		in additionames Compo dx y. c. (b; y. c.	fon to array of vector co nects can b ref. Use who normals	y numeric sub emponents an se covitated and en accessing ve t en accessing ve	e denoted by a d replicated, a, ctors that repre- ctors that repre-	zingle letter: g.: per.ax, pe ent points or ent colors	
Iggregation of Basic Ty Arrays float(3) foo; float foo(3) structure only 1-d structure struct type- members struct type- members struct type-	es and blocks can be arrays invariant arrays supported invariant arrays some { (i)	4 - 1 4 - 7 / 5 5 - 45 / 30 7 - 52 - 67 / 30 8 - 8	postfix increment and decrement prefix increment and decrement unary multiplicative additive shows with reactional equality throwise and unline or		in additionames Compo dx y. c. (b; y. c.	fon to array of vector co nects can b ref. Use who normals	y numeric sub- emponents on se caylooled on an accessing ve t an accessing ve an accessing ve	e denoted by a direplicated, e, ctors that repre-	zingle letter: g.: per.ax, pe ent points or ent colors	
Agregation of Basic Ty Arrays fleet(s) for, fleet fool(s) structure stru	se and block can be array; Investoral a ray outported an ember can be array; (a) (replaced worklin deshedance (b) (replaced worklin deshedance (c) (replaced workling by (c) (replaced workling by (c) (replaced workling)	2 +4- 2 +4- 4 *7/3 5 *7/3 6 *6.50 7 *5/2 *6' 28' 28' 28' 28' 28' 28' 28' 28' 28' 28	posfit incorrect and decrement parts incorrect and decrement unary multiplicative additive situations and exactive thouse and exactive thouse and situative thouse and control of thouse and control of thouse and control of		in additionames Compo dx y. c. (b; y. c.	fon to array of vector co nector can b self Use who comple of Use who of Use who	y numeric sub- emponents on se caylooled on an accessing ve t an accessing ve an accessing ve	e denoted by a d replicated, a, ctors that repre- ctors that repre-	zingle letter: g.: per.ax, pe ent points or ent colors	
Aggregation of Basic Tyy Arrays fleet(3) for, fleet foo(3) fleet foo(3) fleet foo(3) fleet foo(3) fleet foo(3) fleet fleet fleet fleet fleet fleet fleet fleet fleet fleet	se and block can be array; Invariant a ray; appoint an array; appoint an array; (a) (b) (c) (c) (c) (c) (c) (c) (c	2 +4- 2 +4- 4 * 7/5 5 +5 * 20 6 * 50 * 20 * 20 8 * 20 * 20 * 20 * 20 10 * 20 * 20 * 20 * 20 * 20 * 20 * 20 *	coefficients and decrement participation of decrement control of the control of t		in additionames Compo dx y. c. (b; y. c.	fon to array of vector co nector can b self Use who comple of Use who of Use who	y numeric sub- emponents on se caylooled on an accessing ve t an accessing ve an accessing ve	e denoted by a d replicated, a, ctors that repre- ctors that repre-	zingle letter: g.: per.ax, pe ent points or ent colors	
Agregation of Basic Ty Arrays fleet(s) for, fleet fool(s) structure stru	se and block can be array; Investoral a ray outported an ember can be array; (a) (replaced worklin deshedance (b) (replaced worklin deshedance (c) (replaced workling by (c) (replaced workling by (c) (replaced workling)	2 +4- 2 +4- 4 *7/3 5 *7/3 6 *6.50 7 *5/2 *6' 28' 28' 28' 28' 28' 28' 28' 28' 28' 28	posfit incorrect and decrement parts incorrect and decrement unary multiplicative additive situations and exactive thouse and exactive thouse and situative thouse and control of thouse and control of thouse and control of		in additionames Compo dx y. c. (b; y. c.	fon to array of vector co nector can b self Use who comple of Use who of Use who	y numeric sub- emponents on se caylooled on an accessing ve t an accessing ve an accessing ve	e denoted by a d replicated, a, ctors that repre- ctors that repre-	zingle letter: g.: per.ax, pe ent points or ent colors	
Aggregation of Basic Ty Arrays fleetig for fleet toolig structure	se and block can be array; Invariant a ray; appoint an array; appoint an array; (a) (b) (c) (c) (c) (c) (c) (c) (c	2 +4- 2 +4- 4 * 7/5 5 +5 * 20 6 * 50 * 20 * 20 8 * 20 * 20 * 20 * 20 10 * 20 * 20 * 20 * 20 * 20 * 20 * 20 *	coefficients and decrement participation of decrement control of the control of t		in additionames Compo dx y. c. (b; y. c.	fon to array of vector co nector can b self Use who comple of Use who of Use who	y numeric sub- emponents on se caylooled on an accessing ve t an accessing ve an accessing ve	e denoted by a d replicated, a, ctors that repre- ctors that repre-	zingle letter: g.: per.ax, pe ent points or ent colors	

EXEMPLE: PHONG SHADING (VS)

VS (1/3)

```
#version 330 core
```

```
layout (location = 0) in vec3 vertex;
layout (location = 1) in vec3 normal;
layout (location = 2) in vec3 color;
layout (location = 3) in vec2 texCoord;
out vec4 frontColor;
uniform mat4 modelViewProjectionMatrix;
uniform mat4 modelViewMatrix;
uniform mat3 normalMatrix;
uniform vec4 matAmbient, matDiffuse, matSpecular;
uniform float matShininess;
uniform vec4 lightAmbient, lightDiffuse, lightSpecular, lightPosition;
```

VS (2/3)

```
vec4 light(vec3 N, vec3 V, vec3 L)
  N=normalize(N);
  V=normalize(V);
  L=normalize(L);
  vec3 R = normalize( 2.0*dot(N,L)*N-L );
  float NdotL = max(0.0, dot(N,L));
  float RdotV = max(0.0, dot(R,V));
  float Idiff = NdotL;
  float Ispec = 0;
  if (NdotL>0) Ispec=pow( RdotV, matShininess );
             matAmbient * lightAmbient +
  return
    matDiffuse * lightDiffuse * Idiff +
    matSpecular * lightSpecular * Ispec;
```

VS (3/3)

```
void main()
  vec3 P = (modelViewMatrix * vec4(vertex.xyz, 1.0)).xyz;
  vec3 N = normalize(normalMatrix * normal);
  vec3 V = -P;
  vec3 L = (lightPosition.xyz - P);
  frontColor = light(N, V, L);
  gl_Position = modelViewProjectionMatrix * vec4(vertex.xyz, 1.0);
```

MISC

Configuració de **gedit**

Activar syntax highlighting per GLSL 3.30:

```
mkdir ~/.local/share/gtksourceview-3.0/
mkdir ~/.local/share/gtksourceview-3.0/language-specs
cp /assig/grau-g/glsl330.lang ~/.local/share/gtksourceview-3.0/language-specs/
mkdir ~/.config/
mkdir ~/.config/gedit/
mkdir ~/.config/gedit/snippets/
cp /assig/grau-g/glsl.xml ~/.config/gedit/snippets/glsl.xml
```

o directament:

/assig/grau-g/gedit-config

- Activar el plugin "snippets" del gedit (Preferences -> Plugins -> Snippets)
- El plugin del gedit fa que defs[TAB] s'expandeixi a les declaracions de tots els uniforms que envia el viewer