# Introduction to Mesa

## The open-source graphics API implementation library

Samuel Iglesias Gonsálvez
siglesias@igalia.com

Ubucon Europe 2018
Gijón, Spain

# About Me

## Who am I?

- BSc and MSc on Telecommunications Engineering by University of Oviedo.
- Member of the Graphics team at Igalia, an open source consultancy.
- Contributor to Mesa, focusing on Intel GPU drivers for OpenGL and Vulkan.
- Contributor to Khronos's Vulkan conformance test suite and piglit, an open-source OpenGL driver testing framework.

# Introduction

## About GPUs

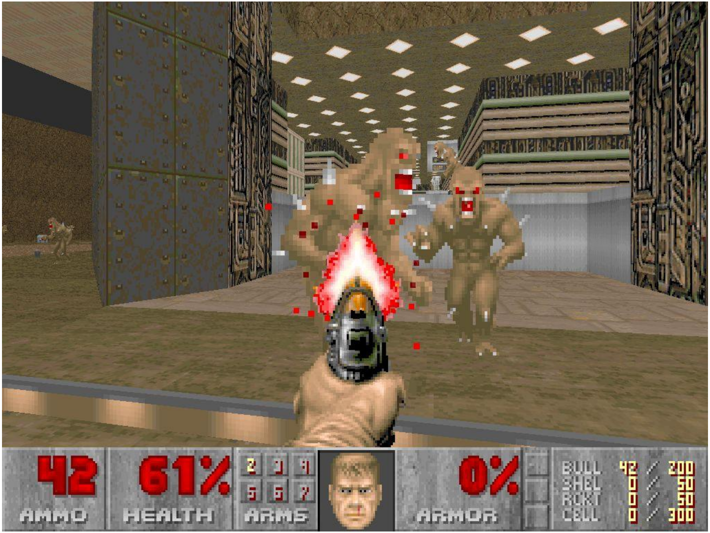
- GPU: graphics processing unit
- *It is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device.* Wikipedia.
- It can also run shaders (code) that has specific inputs/outputs.

# Introduction

## About OpenGL

- OpenGL 1.0 was released in January 1992 by Silicon Graphics (SGI).
  - It was released 26 years ago!
- It was originally based on the SGI's Iris GL API.
- Nowadays, it is maintained by Khronos Group, a consortium of different companies.
- The current version is 4.6, released in July 2017.
- It has extensions that can be optionally supported by the drivers.
- The applications do OpenGL function calls and provide GLSL shaders to do the rendering/computing

# Introduction



# Introduction

# Introduction

## About Vulkan

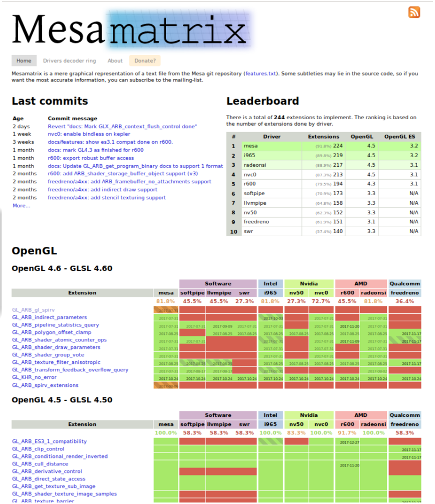
- Vulkan 1.0 was released in February 2016 by Khronos Group.
  - Current version is 1.1, released in March 2018.
- It is based on AMD's Mantle API.
- It was designed to be a considerably lower level API and offering parallel tasking.
  - Vulkan offers lower overhead (so lower CPU usage), more direct control over the GPU.
- The applications do Vulkan function calls and provide SPIR-V shaders to do the rendering/computing.

# Introduction

## About Mesa

- Open-source implementation of the OpenGL and Vulkan specifications for a variety of hardware on user-space as a library.
- The Mesa project was originally started by Brian Paul.
  - Version 1.0 released in February 1995.
  - Current version is 18.0.
- There are drivers for:
  - Intel (i965, i915, anv)
  - AMD (radv, radeonsi, r600)
  - NVIDIA (nouveau)
  - Imagination Technologies (imx)
  - Broadcom (vc4, vc5)
  - Qualcomm (freedreno)
  - Software renderers (classic swrast, softpipe, llvmpipe, OpenSWR)
  - VMware virtual GPU
  - Etc

# Mesa



Mesamatrix

Home | Drivers decoder ring | About | Donate?

Mesamatrix is a mere graphical representation of a text file from the Mesa git repository (features.txt). Some subtleties may lie in the source code, so if you want the most accurate information, you can subscribe to the mailing-list.

## Last commits

| Age | Commit message |
|---|---|
| 2 days | Revert "docs: Mark GLX_ARB_context_flush_control done" |
| 1 week | nvc0: enable bindless on kepler |
| 3 weeks | docs/features: show es3.1 compat done on r600. |
| 1 month | docs: mark GL4.3 as finished for r600 |
| 1 month | r600: export robust buffer access |
| 1 month | docs: Update GL_ARB_get_program_binary docs to support 1 format |
| 2 months | r600: add ARB_shader_storage_buffer_object support (v3) |
| 2 months | freedreno/a4xx: add ARB_framebuffer_no_attachments support |
| 2 months | freedreno/a4xx: add indirect draw support |
| 2 months | freedreno/a4xx: add stencil texturing support |

More...

## Leaderboard

There is a total of **244** extensions to implement. The ranking is based on the number of extensions done by driver.

| # | Driver | Extensions | OpenGL | OpenGL ES |
|---|---|---|---|---|
| 1 | mesa | (91.8%) 224 | 4.5 | 3.2 |
| 2 | i965 | (89.8%) 219 | 4.5 | 3.2 |
| 3 | radeonsi | (88.9%) 217 | 4.5 | 3.1 |
| 4 | nvc0 | (87.3%) 213 | 4.5 | 3.1 |
| 5 | r600 | (79.5%) 194 | 4.3 | 3.1 |
| 6 | softpipe | (70.9%) 173 | 3.3 | N/A |
| 7 | llvmpipe | (64.8%) 158 | 3.3 | N/A |
| 8 | nv50 | (62.3%) 152 | 3.3 | N/A |
| 9 | freedreno | (61.9%) 151 | 3.1 | N/A |
| 10 | swr | (57.4%) 140 | 3.3 | N/A |

## OpenGL

### OpenGL 4.6 - GLSL 4.60

| | | Software | | | Intel | Nvidia | | AMD | | Qualcomm |
|---|---|---|---|---|---|---|---|---|---|---|
| Extension | mesa | softpipe | llvmpipe | swr | i965 | nv50 | nvc0 | r600 | radeonsi | freedreno |
| | 81.8% | 45.5% | 45.5% | 27.3% | 81.8% | 27.3% | 72.7% | 45.5% | 81.8% | 36.4% |
| GL_ARB_gl_spirv | 2017-07-31 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 |
| GL_ARB_indirect_parameters | 2017-07-31 | 🟥 | 🟥 | 🟥 | 2017-10-03 | 🟥 | 2017-07-31 | 🟥 | 2017-07-31 | 🟥 |
| GL_ARB_pipeline_statistics_query | 2017-07-31 | 2017-07-31 | 2017-09-09 | 2017-07-31 | 2017-07-31 | 🟥 | 2017-07-31 | 2017-11-20 | 2017-07-31 | 🟥 |
| GL_ARB_polygon_offset_clamp | 2017-08-25 | 🟥 | 2017-08-25 | 2017-08-25 | 2017-08-25 | 2017-08-25 | 2017-08-25 | 2017-08-25 | 2017-08-25 | 2017-11-17 |
| GL_ARB_shader_atomic_counter_ops | 2017-07-31 | 2017-07-31 | 🟥 | 🟥 | 2017-07-31 | 🟥 | 2017-07-31 | 2017-11-09 | 2017-07-31 | 2017-11-17 |
| GL_ARB_shader_draw_parameters | 2017-07-31 | 🟥 | 🟥 | 🟥 | 2017-07-31 | 🟥 | 2017-07-31 | 🟥 | 2017-07-31 | 🟥 |
| GL_ARB_shader_group_vote | 2017-07-31 | 🟥 | 🟥 | 🟥 | 2017-07-31 | 🟥 | 2017-07-31 | 🟥 | 2017-07-31 | 🟥 |
| GL_ARB_texture_filter_anisotropic | 2017-08-25 | 2017-08-25 | 2017-08-25 | 🟥 | 2017-08-25 | 2017-08-25 | 2017-08-25 | 2017-08-25 | 2017-08-25 | 2017-11-17 |
| GL_ARB_transform_feedback_overflow_query | 2017-07-31 | 2017-08-17 | 2017-08-17 | 🟥 | 2017-07-31 | 🟥 | 🟥 | 🟥 | 2017-08-02 | 🟥 |
| GL_KHR_no_error | 2017-10-24 | 2017-10-24 | 2017-10-24 | 2017-10-24 | 2017-10-24 | 2017-10-24 | 2017-10-24 | 2017-10-24 | 2017-10-24 | 2017-10-24 |
| GL_ARB_spirv_extensions | 2017-08-06 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 |

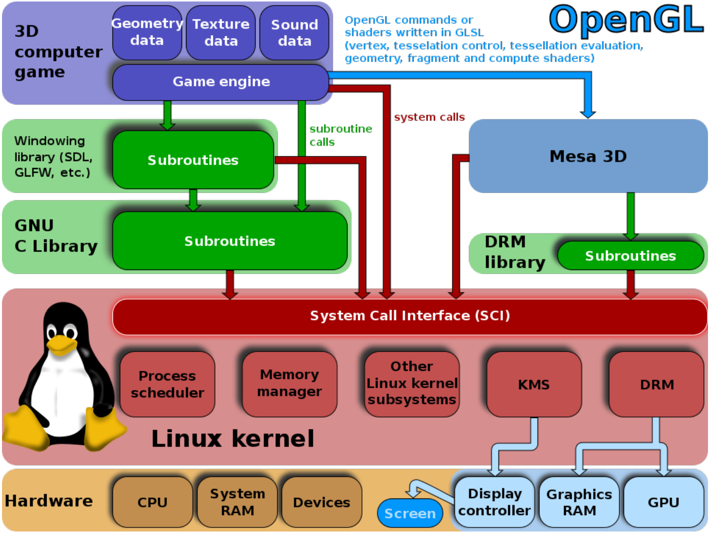
### OpenGL 4.5 - GLSL 4.50

| | | Software | | | Intel | Nvidia | | AMD | | Qualcomm |
|---|---|---|---|---|---|---|---|---|---|---|
| Extension | mesa | softpipe | llvmpipe | swr | i965 | nv50 | nvc0 | r600 | radeonsi | freedreno |
| | 100.0% | 58.3% | 58.3% | 58.3% | 100.0% | 83.3% | 100.0% | 91.7% | 100.0% | 58.3% |
| GL_ARB_ES3_1_compatibility | 🟩 | 🟥 | 🟥 | 🟥 | 🟩 | 🟥 | 🟩 | 2017-12-27 | 🟩 | 🟥 |
| GL_ARB_clip_control | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 2017-11-17 |
| GL_ARB_conditional_render_inverted | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 2017-11-17 |
| GL_ARB_cull_distance | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 2017-11-20 | 🟩 | 🟥 |
| GL_ARB_derivative_control | 🟩 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 |
| GL_ARB_direct_state_access | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| GL_ARB_get_texture_sub_image | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| GL_ARB_shader_texture_image_samples | 🟩 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 |
| GL_ARB_texture_barrier | 🟩 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 2017-11-17 |

## About Mesa

- It supports up to OpenGL 4.6, OpenGL ES 3.2 and Vulkan 1.1.

# Introduction to the Linux Graphics Stack



# How Mesa works internally

## Loading the right driver: OpenGL

- Mesa has a loader that selects the driver by asking vendor id, chip id... to the kernel driver via DRM.
- There is a map of PCI IDs and user-space Mesa drivers.
- When it is found, Mesa loads the respective driver and see if the driver successes; first trying the TLS version, then the non-TLS version.
- In case of failure, the loader tries software renderers.
- It is possible to force software renderer
  - LIBGL_ALWAYS_SOFTWARE=1

# How Mesa works internally

## Loading the right driver: OpenGL

- Mesa has a loader that selects the driver by asking vendor id, chip id... to the kernel driver via DRM.
- There is a map of PCI IDs and user-space Mesa drivers.
- When it is found, Mesa loads the respective driver and see if the driver successes; first trying the TLS version, then the non-TLS version.
- In case of failure, the loader tries software renderers.
- It is possible to force software renderer
  - LIBGL_ALWAYS_SOFTWARE=1

# How Mesa works internally

## Loading the right driver: OpenGL

- Mesa has a loader that selects the driver by asking vendor id, chip id... to the kernel driver via DRM.
- There is a map of PCI IDs and user-space Mesa drivers.
- When it is found, Mesa loads the respective driver and see if the driver successes; first trying the TLS version, then the non-TLS version.
- In case of failure, the loader tries software renderers.
- It is possible to force software renderer
  - LIBGL_ALWAYS_SOFTWARE=1

# How Mesa works internally

## Loading the right driver: OpenGL

- Mesa has a loader that selects the driver by asking vendor id, chip id... to the kernel driver via DRM.
- There is a map of PCI IDs and user-space Mesa drivers.
- When it is found, Mesa loads the respective driver and see if the driver successes; first trying the TLS version, then the non-TLS version.
- In case of failure, the loader tries software renderers.
- It is possible to force software renderer
  - LIBGL_ALWAYS_SOFTWARE=1

# How Mesa works internally

## Loading the right driver: OpenGL

- Mesa has a loader that selects the driver by asking vendor id, chip id... to the kernel driver via DRM.
- There is a map of PCI IDs and user-space Mesa drivers.
- When it is found, Mesa loads the respective driver and see if the driver successes; first trying the TLS version, then the non-TLS version.
- In case of failure, the loader tries software renderers.
- It is possible to force software renderer
  - LIBGL_ALWAYS_SOFTWARE=1

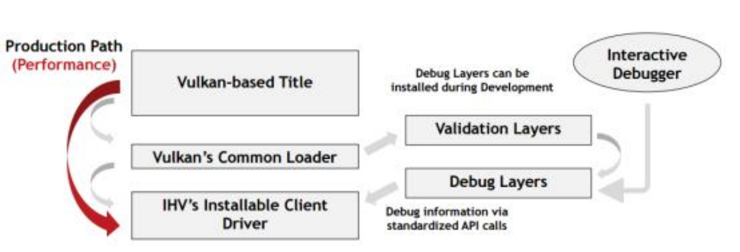
# How Mesa works internally

## Example

```
$ LIBGL_DEBUG=verbose glxgears
libGL: Can't open configuration file /home/siglesias/.drirc: No such file or directory.
libGL: pci id for fd 4: 8086:5917, driver i965
libGL: OpenDriver: trying /home/siglesias/devel/jh-install/lib/dri/tls/i965_dri.so
libGL: OpenDriver: trying /home/siglesias/devel/jh-install/lib/dri/i965_dri.so

...
```

# How Mesa works internally

## Loading the right driver: Vulkan


Production Path
(Performance)
Vulkan-based Title
Vulkan's Common Loader
IHV's Installable Client Driver
Debug Layers can be installed during Development
Interactive Debugger
Validation Layers
Debug Layers
Debug information via standardized API calls

# How Mesa works internally

## Loading the right driver: Vulkan

- Vulkan loader looks for ICD (Installable Client Driver) files in common paths. They tell the loader where the drivers are.
  - It is possible to force other paths:

  export VK_ICD_FILENAMES=$HOME/icd/intel_icd.x86_64.json
- Application asks to the loader for a device enumeration and selects which one(s) it wants to run on.
- It is possible to load directly the driver's library bypassing the loader.

# How Mesa works internally

## Example

```
$ vulkaninfo

...

INFO: [loader] Code 0 : Found ICD manifest file
/usr/share//vulkan/icd.d/radeon_icd.x86_64.json, version "1.0.0"
INFO: [loader] Code 0 : Found ICD manifest file
/usr/share//vulkan/icd.d/intel_icd.x86_64.json, version "1.0.0"

...
```

# How Mesa works internally

## Function hooks, HW limits

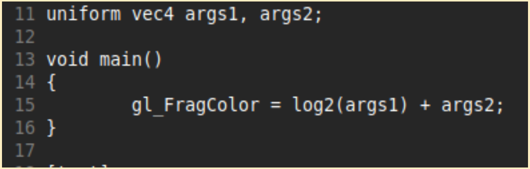
- In case of OpenGL/Vulkan function calls, each driver provides hooks for each of them.
- In some cases, specially on OpenGL, Mesa can provide the same hook for all drivers for functionality that don't need GPU iteraction.
- Each driver provides its own limits (memory size, number of elements of a specific type, etc), although Mesa provides defaults for most of OpenGL limits.

# How Mesa works internally

## Shaders on OpenGL

- On OpenGL, GLSL used to be the language to write them. It is similar to C.
  - On OpenGL, it is driver's duty to compile the GLSL shaders. Mesa provides such GLSL compiler for its drivers and does optimizations to reduce the generated code size.

## GLSL shader example: binary logarithm



```
uniform vec4 args1, args2;

void main()
{
        gl_FragColor = log2(args1) + args2;
}

```

# How Mesa works internally

## Shaders on Vulkan

- On Vulkan, SPIR-V is the binary intermediate language used for shaders.
  - It can be generated from other languages (GLSL, HLSL, others) or written directly in text format and then generate the binary form.
  - SPIR-V has its own compiler provided by Khronos. Drivers don't need to have specific compilers for SPIR-V.
  - Now, OpenGL also supports SPIR-V for shaders through GL_ARB_gl_spirv extension (included in OpenGL 4.6).

# How Mesa works internally

## Shaders on Vulkan

- On Vulkan, SPIR-V is the binary intermediate language used for shaders.
  - It can be generated from other languages (GLSL, HLSL, others) or written directly in text format and then generate the binary form.
  - SPIR-V has its own compiler provided by Khronos. Drivers don't need to have specific compilers for SPIR-V.
  - Now, OpenGL also supports SPIR-V for shaders through GL_ARB_gl_spirv extension (included in OpenGL 4.6).

# How Mesa works internally

## Shaders on Vulkan

- On Vulkan, SPIR-V is the binary intermediate language used for shaders.
    - It can be generated from other languages (GLSL, HLSL, others) or written directly in text format and then generate the binary form.
    - SPIR-V has its own compiler provided by Khronos. Drivers don't need to have specific compilers for SPIR-V.
    - Now, OpenGL also supports SPIR-V for shaders through GL_ARB_gl_spirv extension (included in OpenGL 4.6).

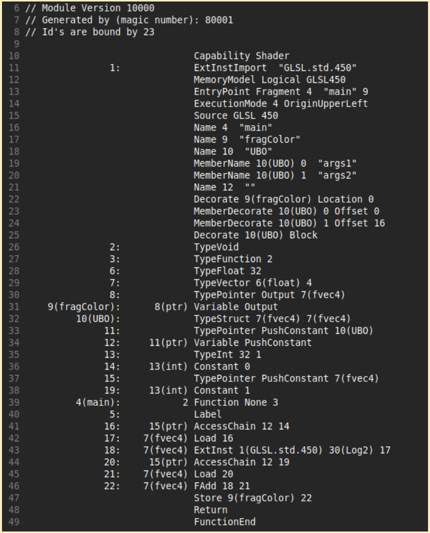
# How Mesa works internally

## Shaders on Vulkan

- On Vulkan, SPIR-V is the binary intermediate language used for shaders.
    - It can be generated from other languages (GLSL, HLSL, others) or written directly in text format and then generate the binary form.
    - SPIR-V has its own compiler provided by Khronos. Drivers don't need to have specific compilers for SPIR-V.
    - Now, OpenGL also supports SPIR-V for shaders through GL_ARB_gl_spirv extension (included in OpenGL 4.6).

# How Mesa works internally

## SPIR-V shader example



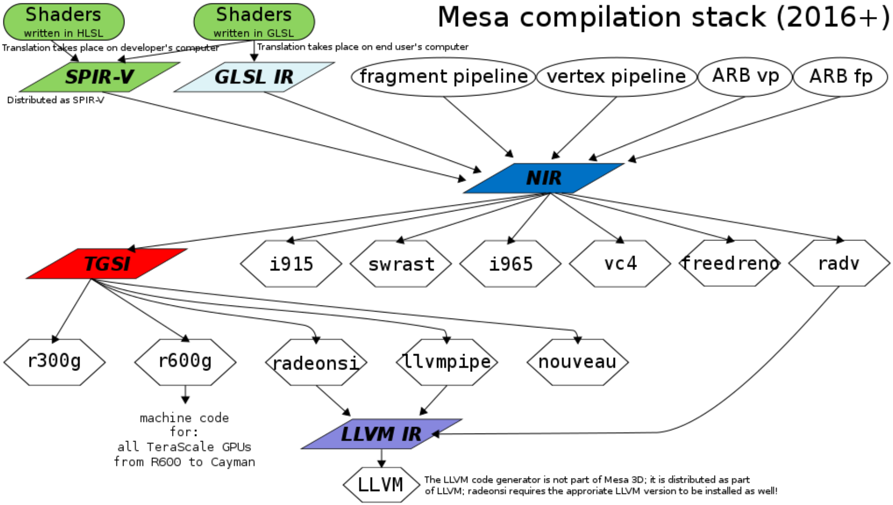
```
// Module Version 10000
// Generated by (magic number): 80001
// Id's are bound by 23

                              Capability Shader
               1:             ExtInstImport  "GLSL.std.450"
                              MemoryModel Logical GLSL450
                              EntryPoint Fragment 4  "main" 9
                              ExecutionMode 4 OriginUpperLeft
                              Source GLSL 450
                              Name 4  "main"
                              Name 9  "fragColor"
                              Name 10  "UBO"
                              MemberName 10(UBO) 0  "args1"
                              MemberName 10(UBO) 1  "args2"
                              Name 12  ""
                              Decorate 9(fragColor) Location 0
                              MemberDecorate 10(UBO) 0 Offset 0
                              MemberDecorate 10(UBO) 1 Offset 16
                              Decorate 10(UBO) Block
               2:             TypeVoid
               3:             TypeFunction 2
               6:             TypeFloat 32
               7:             TypeVector 6(float) 4
               8:             TypePointer Output 7(fvec4)
    9(fragColor):      8(ptr) Variable Output
         10(UBO):             TypeStruct 7(fvec4) 7(fvec4)
              11:             TypePointer PushConstant 10(UBO)
              12:     11(ptr) Variable PushConstant
              13:             TypeInt 32 1
              14:     13(int) Constant 0
              15:             TypePointer PushConstant 7(fvec4)
              19:     13(int) Constant 1
         4(main):           2 Function None 3
               5:             Label
              16:     15(ptr) AccessChain 12 14
              17:    7(fvec4) Load 16
              18:    7(fvec4) ExtInst 1(GLSL.std.450) 30(Log2) 17
              20:     15(ptr) AccessChain 12 19
              21:    7(fvec4) Load 20
              22:    7(fvec4) FAdd 18 21
                              Store 9(fragColor) 22
                              Return
                              FunctionEnd
```

# How Mesa works internally

## Intermediate representations

- SPIR-V: it is a standard created by Khronos and used by Vulkan.
- GLSL IR: it is an internal IR used by Mesa. It represents a list of expression trees.
- NIR: it is an internal IR used by Mesa. It uses SSA which allows to do more optimizations.
- Tungsten Graphics Shader Infrastructure (TGSI) was introduced in 2008 by Tungsten Graphics, used by Gallium drivers (although VC4 and freedreno can consume NIR directly too).
- LLVM IR: it is used by the LLVM compiler. There are LLVM backends to generate assembly code for HW using it as input.

# How Mesa works internally



Mesa compilation stack (2016+)

Shaders
written in HLSL

Shaders
written in GLSL

Translation takes place on developer's computer

Translation takes place on end user's computer

SPIR-V

Distributed as SPIR-V

GLSL IR

fragment pipeline

vertex pipeline

ARB vp

ARB fp

NIR

TGSI

i915

swrast

i965

vc4

freedreno

radv

r300g

r600g

radeonsi

llvmpipe

nouveau

machine code
for:
all TeraScale GPUs
from R600 to Cayman

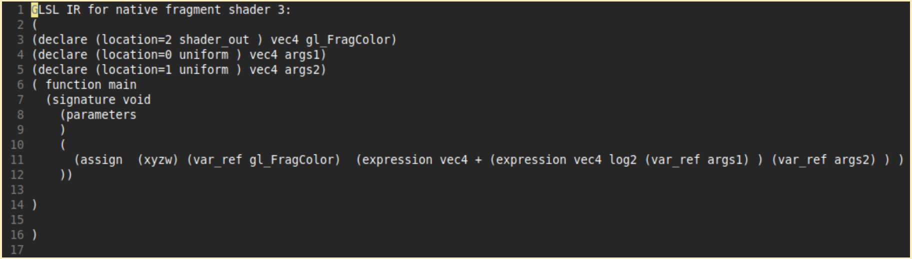
LLVM IR

LLVM

The LLVM code generator is not part of Mesa 3D; it is distributed as part
of LLVM; radeonsi requires the approriate LLVM version to be installed as well!

# How Mesa works internally

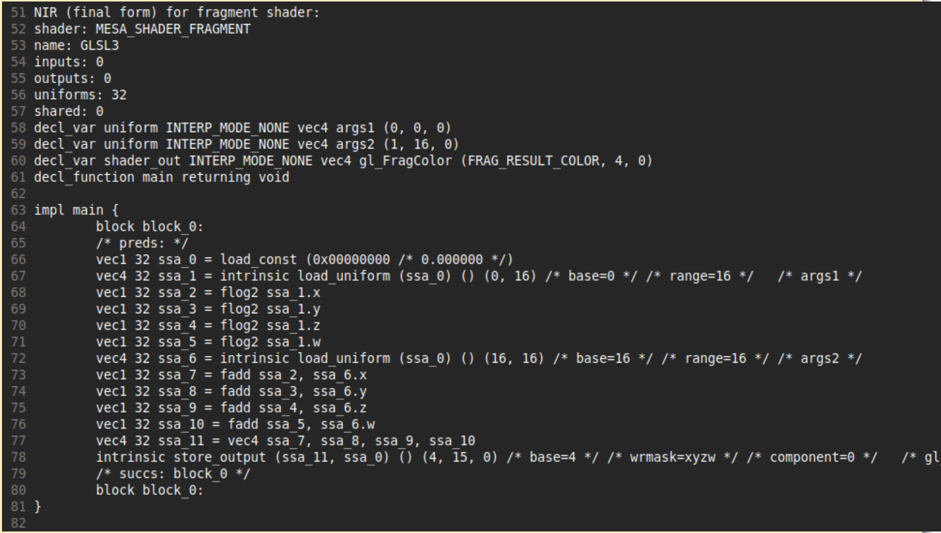
## GLSL IR example



```
GLSL IR for native fragment shader 3:
(
(declare (location=2 shader_out ) vec4 gl_FragColor)
(declare (location=0 uniform ) vec4 args1)
(declare (location=1 uniform ) vec4 args2)
( function main
  (signature void
    (parameters
    )
    (
      (assign  (xyzw) (var_ref gl_FragColor)  (expression vec4 + (expression vec4 log2 (var_ref args1) ) (var_ref args2) ) )
    ))

)

)
```

# How Mesa works internally

## NIR example



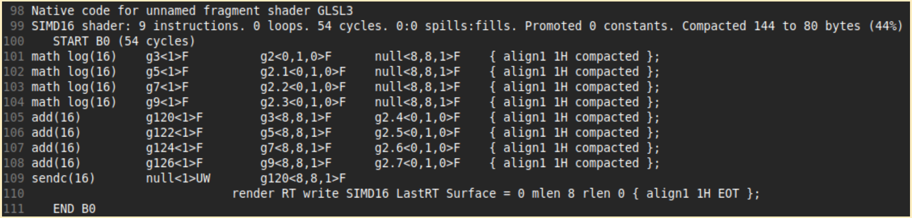
```
NIR (final form) for fragment shader:
shader: MESA_SHADER_FRAGMENT
name: GLSL3
inputs: 0
outputs: 0
uniforms: 32
shared: 0
decl_var uniform INTERP_MODE_NONE vec4 args1 (0, 0, 0)
decl_var uniform INTERP_MODE_NONE vec4 args2 (1, 16, 0)
decl_var shader_out INTERP_MODE_NONE vec4 gl_FragColor (FRAG_RESULT_COLOR, 4, 0)
decl_function main returning void

impl main {
        block block_0:
        /* preds: */
        vec1 32 ssa_0 = load_const (0x00000000 /* 0.000000 */)
        vec4 32 ssa_1 = intrinsic load_uniform (ssa_0) () (0, 16) /* base=0 */ /* range=16 */   /* args1 */
        vec1 32 ssa_2 = flog2 ssa_1.x
        vec1 32 ssa_3 = flog2 ssa_1.y
        vec1 32 ssa_4 = flog2 ssa_1.z
        vec1 32 ssa_5 = flog2 ssa_1.w
        vec4 32 ssa_6 = intrinsic load_uniform (ssa_0) () (16, 16) /* base=16 */ /* range=16 */ /* args2 */
        vec1 32 ssa_7 = fadd ssa_2, ssa_6.x
        vec1 32 ssa_8 = fadd ssa_3, ssa_6.y
        vec1 32 ssa_9 = fadd ssa_4, ssa_6.z
        vec1 32 ssa_10 = fadd ssa_5, ssa_6.w
        vec4 32 ssa_11 = vec4 ssa_7, ssa_8, ssa_9, ssa_10
        intrinsic store_output (ssa_11, ssa_0) () (4, 15, 0) /* base=4 */ /* wrmask=xyzw */ /* component=0 */   /* gl
        /* succs: block_0 */
        block block_0:
}

```

# How Mesa works internally

## GPU assembly code

- The driver takes the IR as input and generates the assembly code that the GPU understands.
- Each manufacturer has its own assembly code. The driver can generate it by itself or, for some gallium drivers, via LLVM.
- When the application wants to draw, the assembly is submitted to the GPU among other things for its execution.

## Example: Intel



```
Native code for unnamed fragment shader GLSL3
SIMD16 shader: 9 instructions. 0 loops. 54 cycles. 0:0 spills:fills. Promoted 0 constants. Compacted 144 to 80 bytes (44%)
   START B0 (54 cycles)
math log(16)    g3<1>F          g2<0,1,0>F      null<8,8,1>F    { align1 1H compacted };
math log(16)    g5<1>F          g2.1<0,1,0>F    null<8,8,1>F    { align1 1H compacted };
math log(16)    g7<1>F          g2.2<0,1,0>F    null<8,8,1>F    { align1 1H compacted };
math log(16)    g9<1>F          g2.3<0,1,0>F    null<8,8,1>F    { align1 1H compacted };
add(16)         g120<1>F        g3<8,8,1>F      g2.4<0,1,0>F    { align1 1H compacted };
add(16)         g122<1>F        g5<8,8,1>F      g2.5<0,1,0>F    { align1 1H compacted };
add(16)         g124<1>F        g7<8,8,1>F      g2.6<0,1,0>F    { align1 1H compacted };
add(16)         g126<1>F        g9<8,8,1>F      g2.7<0,1,0>F    { align1 1H compacted };
sendc(16)       null<1>UW       g120<8,8,1>F
                            render RT write SIMD16 LastRT Surface = 0 mlen 8 rlen 0 { align1 1H EOT };
   END B0
```

# How does development work?

## Community

- Volunteers!
- Companies
  - AMD
  - Collabora
  - Feral Interactive
  - Google
  - Intel
  - Igalia
  - NVIDIA
  - Red Hat
  - Samsung
  - Valve
  - VMware
  - ...

# How does development work?

## Coordination, review, bugs

- Development Mailing list
  - https://lists.freedesktop.org/mailman/listinfo/mesa-dev
- IRC channels at Freenode, some drivers have their own.
  - #dri-devel #intel-3d #nouveau #radeon
- Issue tracker.
  - https://bugs.freedesktop.org/enter_bug.cgi?product=Mesa

## Important!

- All the patches are reviewed in the mailing list!
- No patch lands without a Reviewed-by!
- Avoid adding regressions.
  - Test the patch before submission.
  - Some companies have continuous integration instances.

# How does development work?

## Coordination, review, bugs

- Development Mailing list
  - https://lists.freedesktop.org/mailman/listinfo/mesa-dev
- IRC channels at Freenode, some drivers have their own.
  - #dri-devel #intel-3d #nouveau #radeon
- Issue tracker.
  - https://bugs.freedesktop.org/enter_bug.cgi?product=Mesa

## Important!

- All the patches are reviewed in the mailing list!
- No patch lands without a Reviewed-by!
- Avoid adding regressions.
  - Test the patch before submission.
  - Some companies have continuous integration instances.

# How does development work?

## Coordination, review, bugs

- Development Mailing list
  - https://lists.freedesktop.org/mailman/listinfo/mesa-dev
- IRC channels at Freenode, some drivers have their own.
  - #dri-devel #intel-3d #nouveau #radeon
- Issue tracker.
  - https://bugs.freedesktop.org/enter_bug.cgi?product=Mesa

## Important!

- All the patches are reviewed in the mailing list!
- No patch lands without a Reviewed-by!
- Avoid adding regressions.
  - Test the patch before submission.
  - Some companies have continuous integration instances.

# How does development work?

## Coordination, review, bugs

- Development Mailing list
  - https://lists.freedesktop.org/mailman/listinfo/mesa-dev
- IRC channels at Freenode, some drivers have their own.
  - #dri-devel #intel-3d #nouveau #radeon
- Issue tracker.
  - https://bugs.freedesktop.org/enter_bug.cgi?product=Mesa

## Important!

- All the patches are reviewed in the mailing list!
- No patch lands without a Reviewed-by!
- Avoid adding regressions.
  - Test the patch before submission.
  - Some companies have continuous integration instances.

# How does development work?

## Releases

- There is one major Mesa stable release per quarter.
  - Version numbering is now YEAR.release_number.
  - For example: 18.0.
- Minor releases fortnightly.
  - Fixes for bugs, security issues, etc.
  - Version numbering is now YEAR.release_number.minor_number.
  - For example: 18.0.1.
- In each case, there are release candidates for testing before releasing.
- Announcements
  - https://lists.freedesktop.org/mailman/listinfo/mesa-announce

# How does development work?

## Releases

- There is one major Mesa stable release per quarter.
    - Version numbering is now YEAR.release_number.
    - For example: 18.0.
- Minor releases fortnightly.
    - Fixes for bugs, security issues, etc.
    - Version numbering is now YEAR.release_number.minor_number.
    - For example: 18.0.1.
- In each case, there are release candidates for testing before releasing.
- Announcements
    - https://lists.freedesktop.org/mailman/listinfo/mesa-announce

# How does development work?

## Releases

- There is one major Mesa stable release per quarter.
  - Version numbering is now YEAR.release_number.
  - For example: 18.0.
- Minor releases fortnightly.
  - Fixes for bugs, security issues, etc.
  - Version numbering is now YEAR.release_number.minor_number.
  - For example: 18.0.1.
- In each case, there are release candidates for testing before releasing.
- Announcements
  - https://lists.freedesktop.org/mailman/listinfo/mesa-announce

# How does development work?

## Releases

- There is one major Mesa stable release per quarter.
  - Version numbering is now YEAR.release_number.
  - For example: 18.0.
- Minor releases fortnightly.
  - Fixes for bugs, security issues, etc.
  - Version numbering is now YEAR.release_number.minor_number.
  - For example: 18.0.1.
- In each case, there are release candidates for testing before releasing.
- Announcements
  - https://lists.freedesktop.org/mailman/listinfo/mesa-announce

# Interested in contributing to Mesa?

## How to install latest version of Mesa

- Install it from PPAs. Choose one of these:
  - https://launchpad.net/~oibaf/+archive/ubuntu/graphics-drivers
  - https://launchpad.net/~paulo-miguel-dias/+archive/ubuntu/mesa
- Install it from git repository
  - $ sudo apt install git
  - $ sudo apt build-dep mesa
  - $ git clone git://anongit.freedesktop.org/mesa/mesa
  - $ cd mesa && ./autogen.sh && make && make install
  - Set LIBGL_DRIVERS_PATH and LD_LIBRARY_PATH environment variables to run the application with it.
  - More info: https://blogs.igalia.com/itoral/2014/09/15/setting-up-a-development-environment-for-mesa/

# Interested in contributing to Mesa?

### How to install latest version of Mesa

- Install it from PPAs. Choose one of these:
  - https://launchpad.net/~oibaf/+archive/ubuntu/graphics-drivers
  - https://launchpad.net/~paulo-miguel-dias/+archive/ubuntu/mesa
- Install it from git repository
  - $ sudo apt install git
  - $ sudo apt build-dep mesa
  - $ git clone git://anongit.freedesktop.org/mesa/mesa
  - $ cd mesa && ./autogen.sh && make && make install
  - Set LIBGL_DRIVERS_PATH and LD_LIBRARY_PATH environment variables to run the application with it.
  - More info: https://blogs.igalia.com/itoral/2014/09/15/setting-up-a-development-environment-for-mesa/

# Interested in contributing to Mesa?

## As a non-developer

- Use the open-source drivers!
- Help testing!
  - Run 3D applications, games.
  - Testing suites like piglit, dEQP and Vulkan/OpenGL CTS.
- Report bugs to upstream issue tracker!
  - https://bugs.freedesktop.org/enter_bug.cgi?product=Mesa

# Interested in contributing to Mesa?

## How to report a bug

- Check first if it was already reported!
- Select the affected driver and think about a good title for the bug.
- Explain the steps to reproduce it.
- Include software version, Mesa version, kernel version or any other relevant info.
- In case of proprietary software, attach the output of apitrace.
    - http://apitrace.github.io/
- More info
    - https://01.org/linuxgraphics/documentation/how-report-bugs

# Interested in contributing to Mesa?

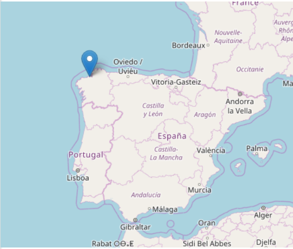
## As a developer

- There is always a need for developers.
- Look for missing features/extensions, or bugs affecting your HW.
- Help debugging existing issues!
- Send patches to the mesa-dev mailing list for review.
- After several successful submissions, you can ask for commit rights!
  - https://www.freedesktop.org/wiki/AccountRequests/
- More info
  - https://www.mesa3d.org/codingstyle.html
  - https://www.mesa3d.org/submittingpatches.html
  - https://www.mesa3d.org/devinfo.html
  - https://www.mesa3d.org/envvars.html
  - https://www.mesa3d.org/helpwanted.html

# Interested in contributing to Mesa?

## X.org Developer's Conference 2018

- Where: A Coruña, Spain
- When: September 26-28, 2018
- Attendees: developers that work on: Linux kernel graphics drivers, Mesa, DRM, X11, Wayland, frameworks, etc.
- Website: https://xdc2018.x.org
- Twitter: https://twitter.com/xdc2018



# More info

## Links

- Website: https://www.mesa3d.org/
- Repository: https://cgit.freedesktop.org/mesa/mesa/
- Mailing lists: https://www.mesa3d.org/lists.html
- Issue tracker: https://bugs.freedesktop.org/describecomponents.cgi?product=Mesa
- IRC (Freenode): #dri-devel #intel-3d #nouveau #radeon.
- Blog aggregation: https://planet.freedesktop.org
- Mesa Matrix: https://mesamatrix.net/

# More info

## Links

- Piglit
  - https://cgit.freedesktop.org/piglit
  - How to use it: https://blogs.igalia.com/siglesias/2014/11/11/piglit-an-open-source-test-suite-for-opengl-implementations/
- Vulkan/OpenGL CTS
  - https://github.com/KhronosGroup/VK-GL-CTS
- apitrace
  - http://apitrace.github.io/

# QA

Questions?

### Slides of the talk

Slides will be available at http://samuelig.es and at Ubucon website (?) in the coming days.

# Introduction to Mesa

## The open-source graphics API implementation library

Samuel Iglesias Gonsálvez
siglesias@igalia.com

Ubucon Europe 2018
Gijón, Spain