

0 A.D. game: Vulkan API

Vladislav Belov



A.D.
Empires Ascendant

FOSDEM'24

0 A.D.



0 A.D.



∅ A.D.



Vladislav Belov

⊕ A.D. game: Vulkan API

0 A.D.

Filters

Clear

Search

TYPE

- Language Pack 4
- Total Conversion 4
- Factions 12
- Gameplay 38
- Utility 5



0 A.D. Delenda Est

57K 399.11 MB 100%

Subscribe



Millennium A.D.

34K 437.32 MB 96%

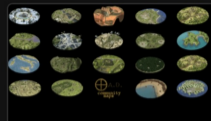
Subscribe



Terra Magna

31K 89.82 MB 100%

Subscribe



Community Maps

18K 10.98 MB 100%

Subscribe



Mayas 0 AD

10K 119.59 MB 100%

Subscribe



Theban Greeks

10K 1.29 MB 100%

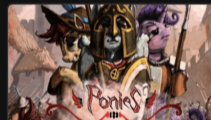
Subscribe



0 A.D. Community Mod

17K 5.36 MB 100%

Subscribe



Ponies Ascendant

11K 80.41 MB 100%

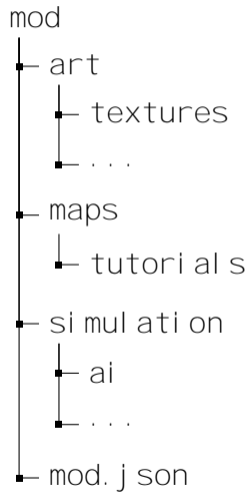
Subscribe



Vladislav Belov

A.D. game: Vulkan API

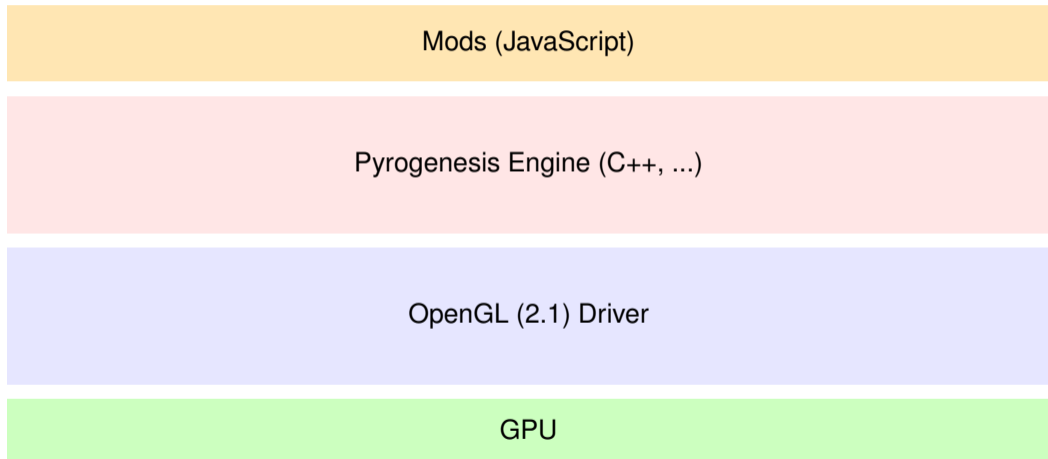
0 A.D.



Graphics



Structure Before



Code Before

```
gl DepthMask(0);
```

```
CPatchRData::RenderBlends(vsi bl ePatches, context, shadow);
```

```
CDecal RData::RenderDecals(vsi bl eDecals, context, shadow);
```

```
g_Renderer.BindTexture(1, 0);
```

```
g_Renderer.BindTexture(2, 0);
```

```
g_Renderer.BindTexture(3, 0);
```

```
gl DepthMask(1);
```

```
gl BlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

```
gl Disable(GL_BLEND);
```

OpenGL Problems

CPU performance and unpredictable cost of some function calls

No proper queries for supported features/GPUs

Lack of additional/debug validations

Single-threaded

OpenGL Versions reported by drivers

3.3

3.3.0

3.3.0 - Build 8.0.0.1000

3.3.9000 Compatibility Profile Context

3.3 (Compatibility Profile) Mesa 23.1.2

3.3 Mesa 21.3.5

3.3 ATI-4.8.101

3.3 NVIDIA-10.17.5 355.10.05.45f01

OpenGL GPUs reported by drivers

Radeon RX 550

Radeon (TM) RX 550

Radeon(TM) RX 550

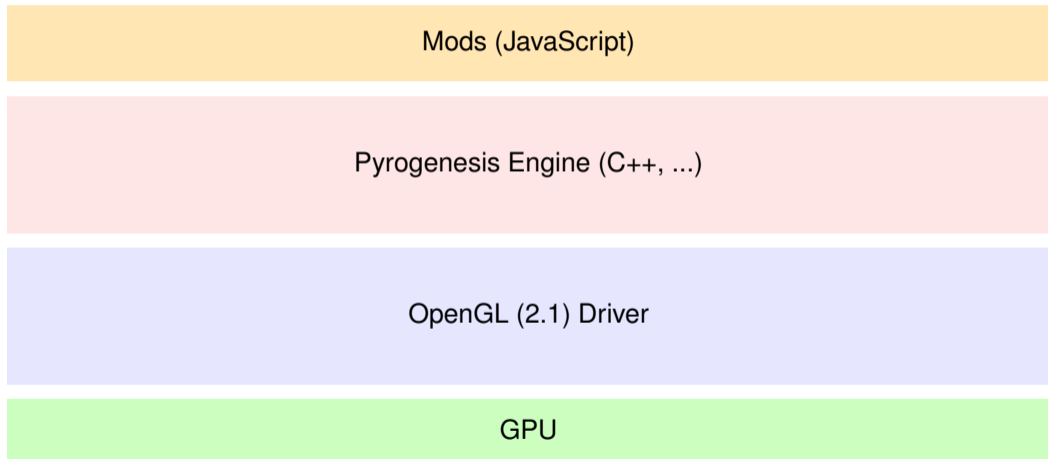
Radeon RX550/550 Series

AMD Radeon RX 550 / 550 Series

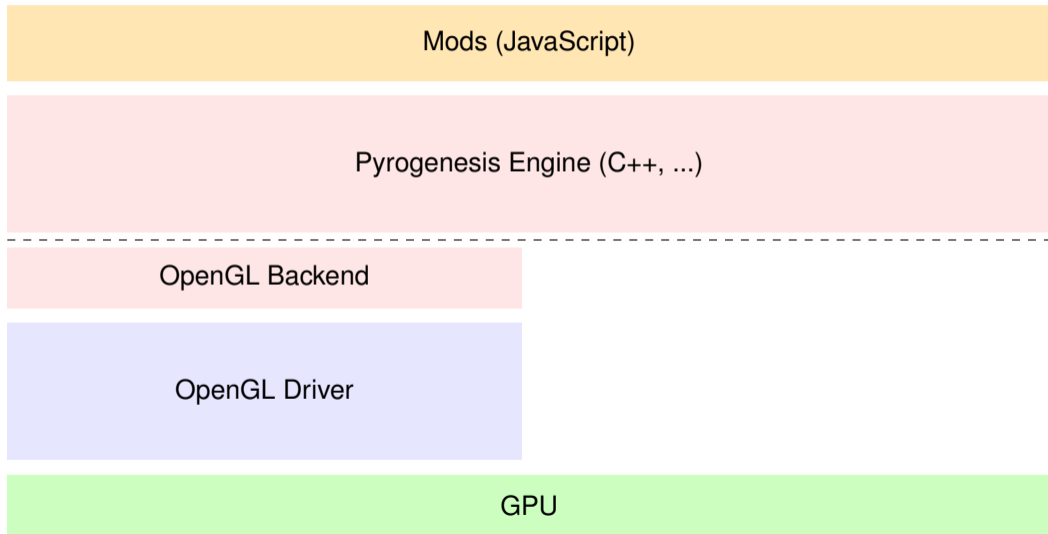
AMD Radeon RX 550 Series

AMD Radeon RX550 series

Structure Before



Structure



Interfaces

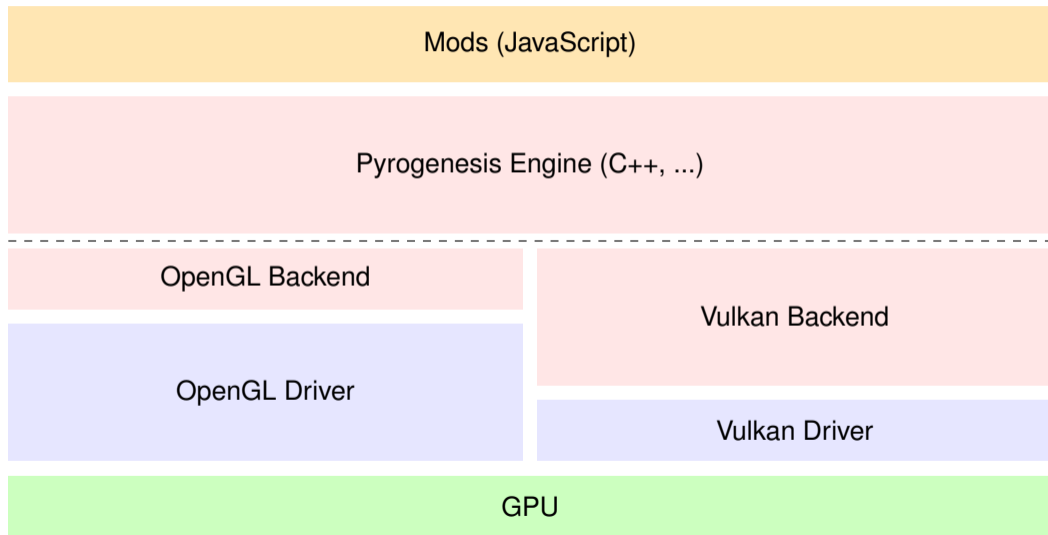
```
class IDevice
{
    // ...
    virtual std::unique_ptr<ITexture> CreateTexture(
        const char* name, const ITexture::Type type, const uint32_t usage,
        const Format format, const uint32_t width, const uint32_t height,
        const Sampler::Desc& defaultSamplerDesc, const uint32_t MIPLevelCount,
        const uint32_t sampleCount) = 0;
    // ...
};

class IDeviceCommandContext
{
    // ...
    virtual void SetTexture(const int32_t bindingSlot, ITexture* texture) = 0;
    // ...
};
```

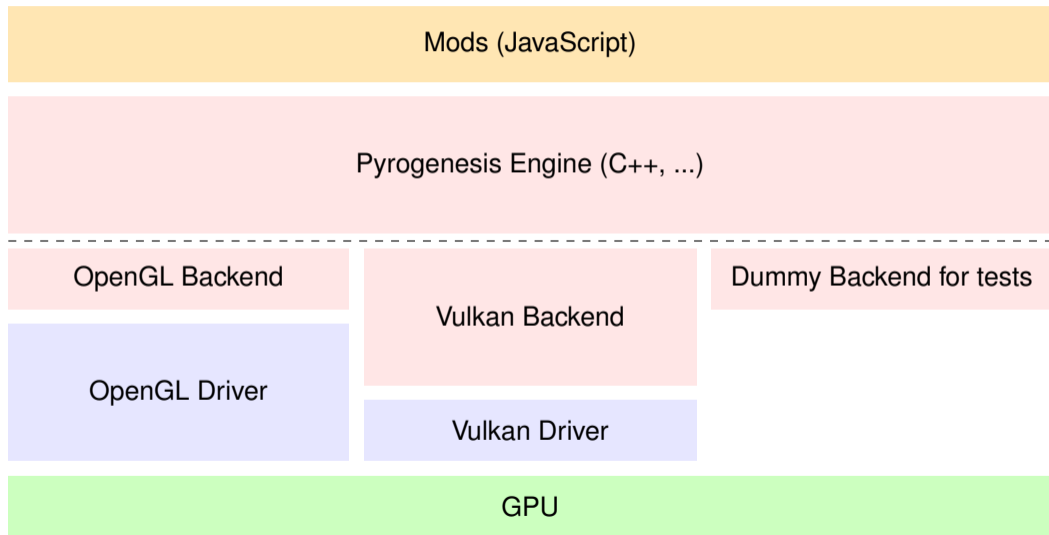
```
CPatchRData::RenderBlends(  
    deviceCommandContext, m->blendVertexInputLayout,  
    visiblePatches, context, shadow);
```

```
CDecalRData::RenderDecals(  
    deviceCommandContext, m->decalVertexInputLayout,  
    visibleDecals, context, shadow);
```

Structure



Structure



Code Comparison

OpenGL backend 4k lines of code

Vulkan backend 8k + VMA 17k lines of code

Results

From 2021 to 2023 it took 1.5-2 months of full time work in total to add the abstraction

About 2 weeks of full time work in total to add the Vulkan backend

Validation layers

Proper information about GPU and its Vendor

Possible multi-threading support

Performance

Was it worth it?

Definitely!

Performance (up to 3x times faster for some platforms)

Stability and more predictable behavior (less amount of silent errors)

Features (modern GPU functionality)

Does everyone need Vulkan at all?

Ideally - yes!

If your engine/application supports Vulkan - enable it

If you can relatively easy switch to Vulkan (or a third party library which supports Vulkan) - do it

If you use an own custom engine - spend time on it only if you need it

