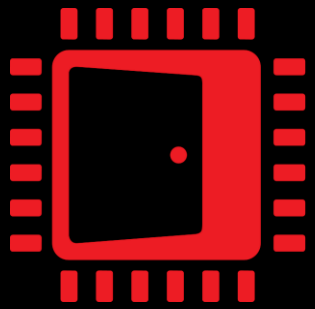
AMD EPYC

AMD RYZEN

AMD RADEON



AMD GPUOpen

THE FIDELITYFX™ SDK

JASON LACROIX – PMTS, AMD GAME ENGINEERING

AMD
together we advance_

OUTLINE

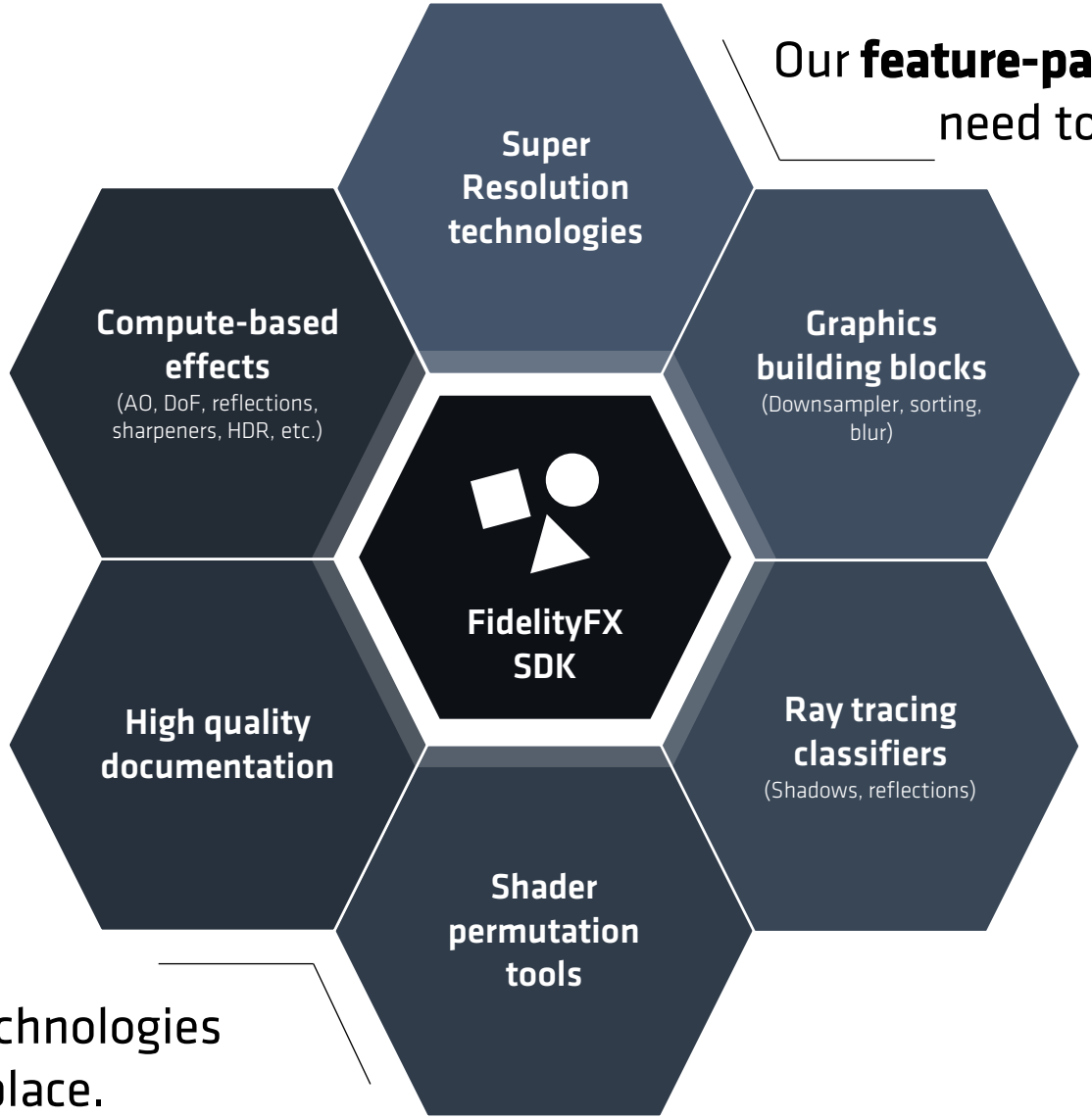
- FidelityFX SDK
- Cauldron 2.0
- SDK Samples
- Wrap up

AMD FidelityFX SDK

FIDELITYFX SDK

- Overview

Our **feature-packed** SDK offers the tools you need to create world-class renderers.



All the great FidelityFX™ technologies you know and love, in one place.

	MIT license	Open source	Consistent user experience	Clean architecture	Easy-to-use API	High quality documentation	DirectX®12 Support	Vulkan® Support	Xbox Support	Easy to integrate	Shader compilation tools
FidelityFX™	✓	✓	✗	✗	✗	✗	✓	–	–	–	✗

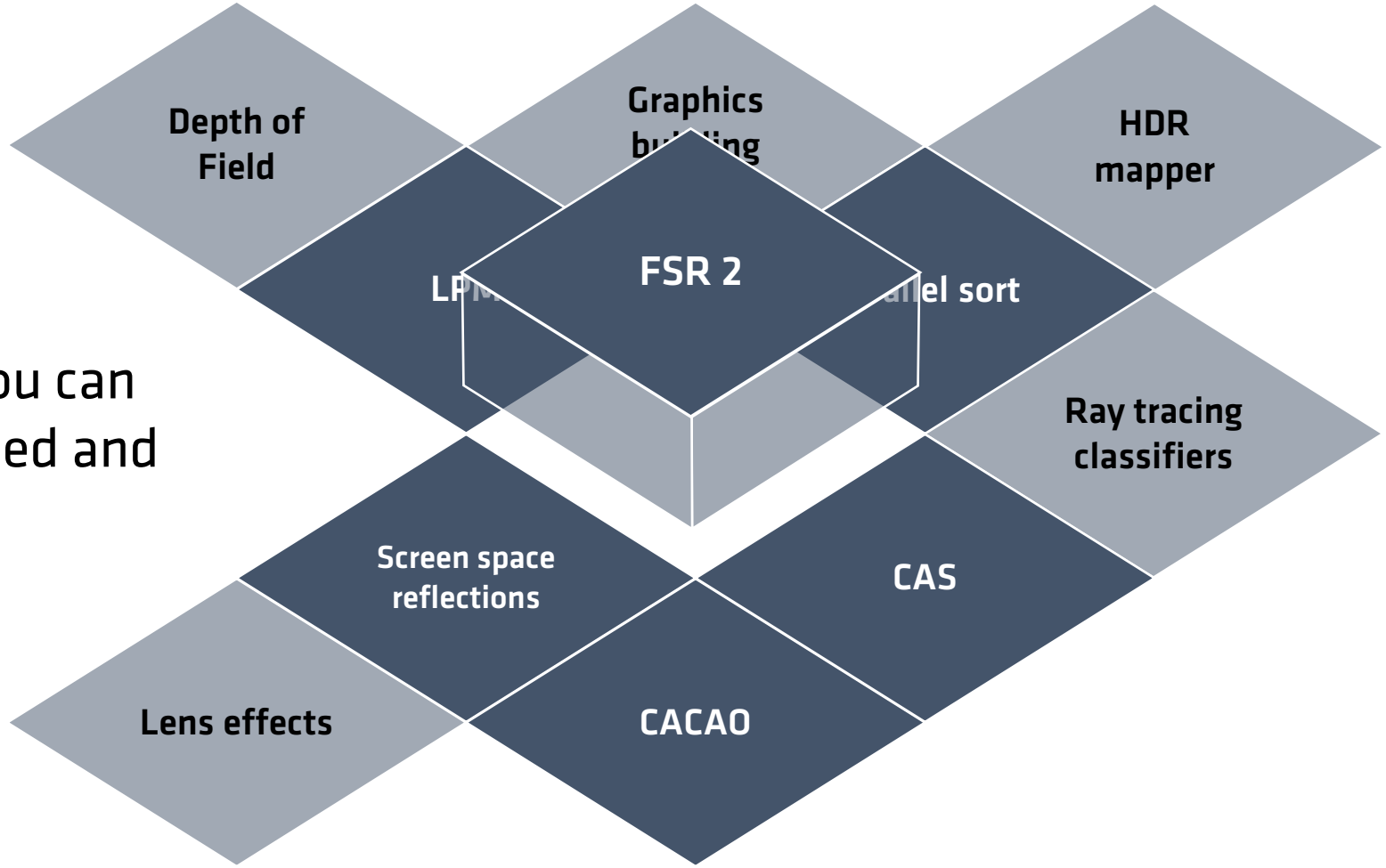
– Mixed

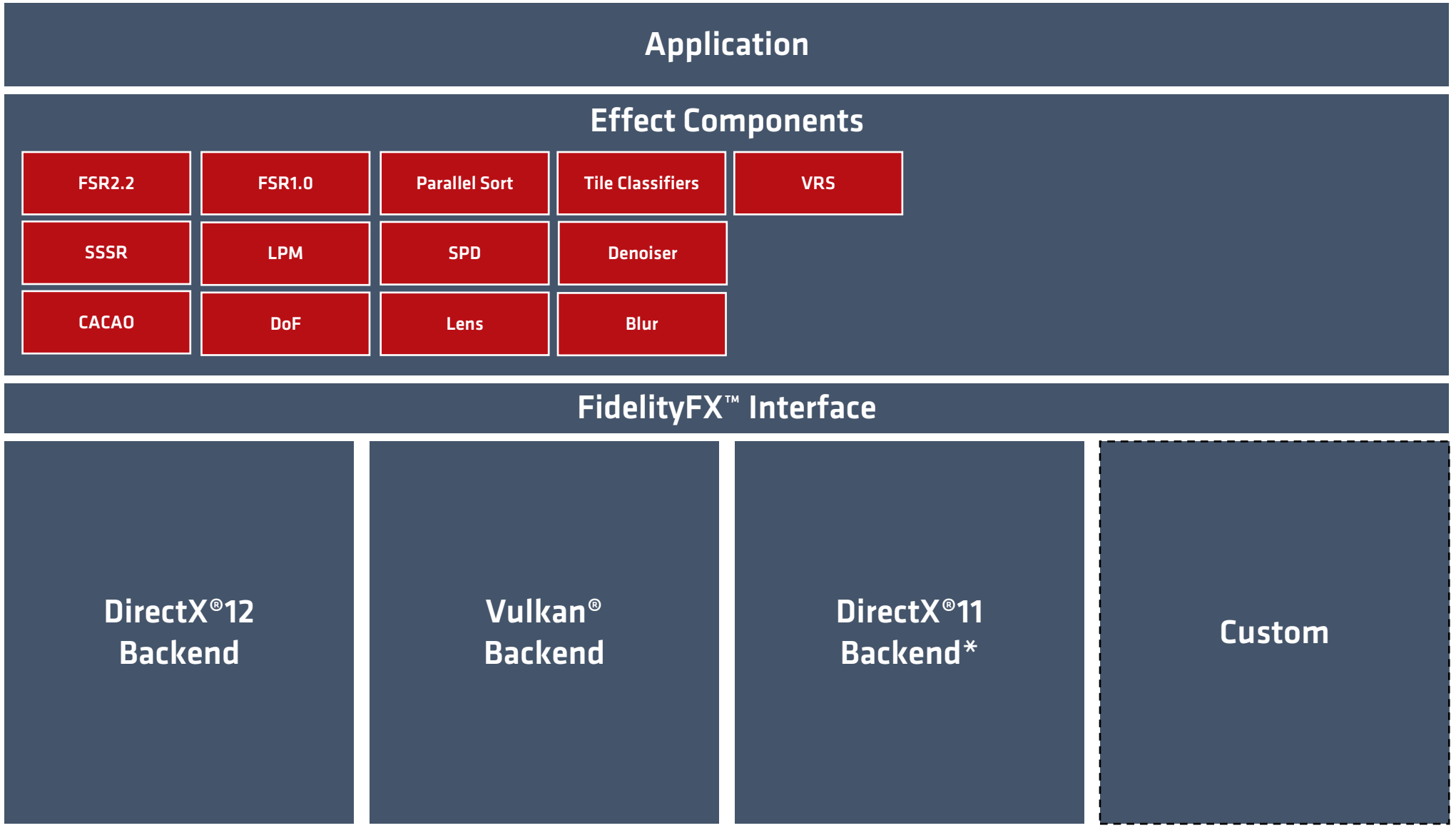
	MIT license	Open source	Consistent user experience	Clean architecture	Easy-to-use API	High quality documentation	DirectX®12 Support	Vulkan® Support	Xbox Support	Easy to integrate	Shader compilation tools
FidelityFX™	✓	✓	✗	✗	✗	✗	✓	–	–	–	✗
FidelityFX™ SDK	✓	✓	✓	✓	✓	✓	✓	✓	✓ *	✓	✓

– Mixed

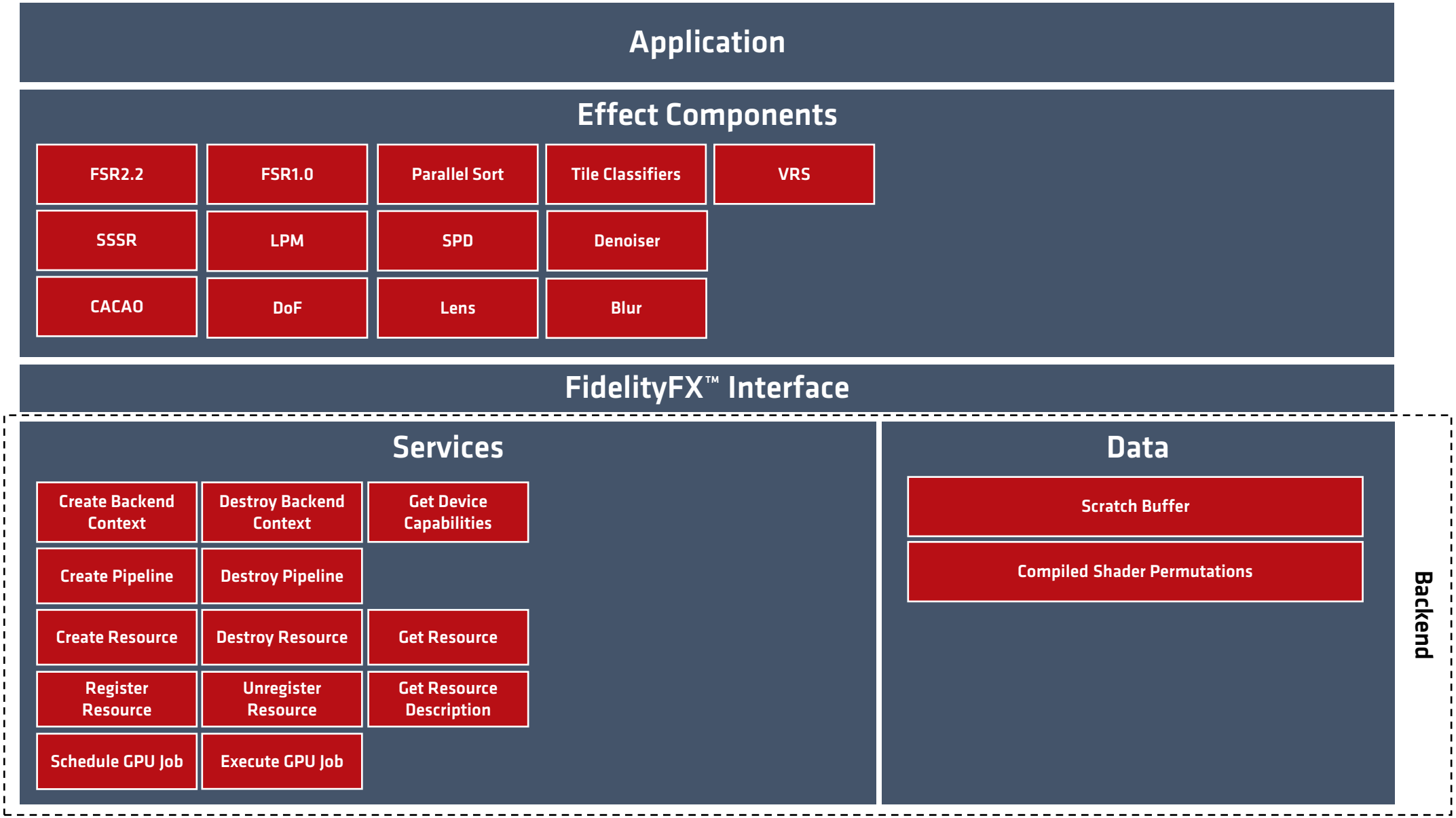
* Coming soon

A **modular** architecture means you can take just the components you need and effortlessly bolt them together.





* By request from NDA partners


Application
Effect Components
FSR2.2
FSR1.0
Parallel Sort
Tile Classifiers
VRS
SSSR
LPM
SPD
Denoiser
CACAO
DoF
Lens
Blur
FidelityFX™ Interface
Services
Create Backend Context
Destroy Backend Context
Get Device Capabilities
Create Pipeline
Destroy Pipeline
Create Resource
Destroy Resource
Get Resource
Register Resource
Unregister Resource
Get Resource Description
Schedule GPU Job
Execute GPU Job
Data
Scratch Buffer
Compiled Shader Permutations
Backend

SDK

Include/FidelityFX

GPU

FFX Core Headers
(Shader APIs and general GPU programming abstractions and utilities)

Effect Components
(one directory per effect component, each containing shader code for that effect)

Host

ffx.h

Backends

DX12

Vulkan

Effect Components
(one directory per effect component, each containing C-style API header files)

Source

Effects
(one directory per effect component, each containing source code for that effect)

Backends

DX12

Vulkan

Shared

Tools

FidelityFX™ Shader Pre-compiler

Samples

Effects

Blur	CACAO
CAS	DOF
FSR1	FSR2
HDR	Lens
Parallel Sort	SPD
SSSR	VRS

Cauldron 2

Render Modules

Third Party

Documentation

FIDELITYFX SDK

- Usage example

```
#include <FidelityFX/ffx.h>

// Create a backend for FidelityFX.
const size_t scratchBufferSize = ffxGetScratchMemorySizeDX12(FFX_BLUR_CTX_COUNT);
void* scratchBuffer = malloc(scratchBufferSize);
FFX_ASSERT(scratchBuffer);
FfxInterface* backendInterface = NULL;
FfxErrorCode errorCode = ffxGetInterfaceDX12(&backendInterface, dx12Device,
    scratchBuffer, scratchBufferSize, FFX_BLUR_CTX_COUNT);

// Create the blur effect context.
FfxBlurContext blurContext = {};
FfxBlurContextDescription contextDescription = {};
contextDescription.backendInterface = backendInterface;
contextDescription.kernelSizes = FFX_BLUR_KERNEL_SIZE_ALL; // allow all kernel sizes
contextDescription.floatPrecision = FFX_BLUR_FLOAT_PRECISION_16BIT;
errorCode = ffxBlurContextCreate(&blurContext, & contextDescription);
FFX_ASSERT(errorCode == FFX_OK);
```

```
// Describe the blur we want to perform.
FfxBlurDispatchDescription dispatchDescription = {};
dispatchDescription.commandList = ffxGetCommandListDX12(dx12CommandList);
dispatchDescription.kernelSize = FFX_BLUR_KERNEL_SIZE_13x13; // many choices!
dispatchDescription.inputAndOutputSize.width = AppGetWidth();
dispatchDescription.inputAndOutputSize.height = AppGetHeight();
dispatchDescription.input = ffxGetResourceDX12(
    inputResource,
    L"MyBlurInput",
    FFX_RESOURCE_STATE_PIXEL_COMPUTE_READ);
dispatchDescription.output = ffxGetResourceDX12(
    outputResource,
    L"MyBlurOutput",
    FFX_RESOURCE_STATE_UNORDERED_ACCESS);

// Perform the blur.
ffxBlurContextDispatch(&blurContext, &dispatchDescription);
```

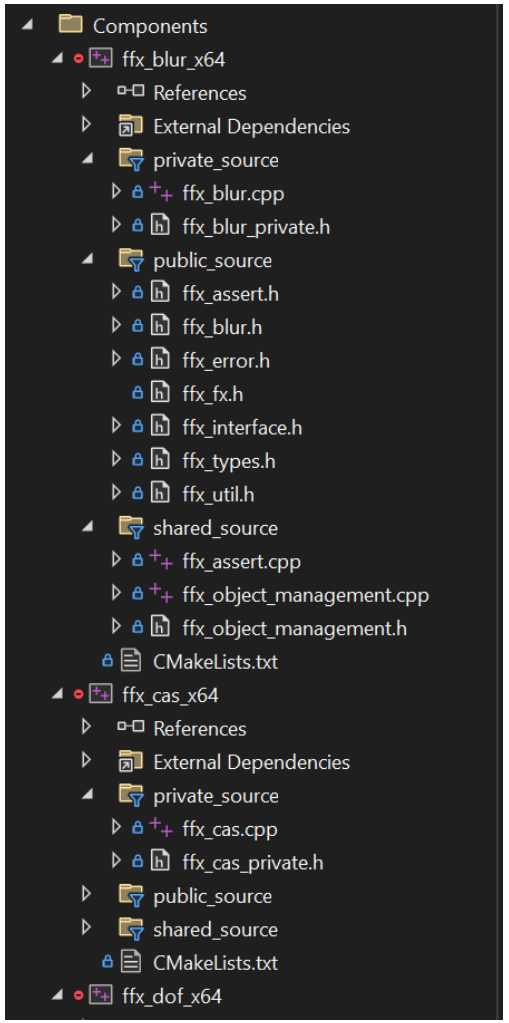
```
// Destroy the blur effect context when shutting everything down.
ffxBlurContextDestroy(&blurContext);

// Don’t forget to destroy the backing memory for the interface.
if (backendInterface.scratchBuffer) {
    free(backendInterface.scratchBuffer);
}
```

FIDELITYFX SDK

- Internals – Effect Components

ANATOMY OF AN EFFECT COMPONENT

- Public source
  - ffx common headers
  - `ffx_<effect>.h`
- Shared source
  - Asserts and object management
- Private source
  - Component implementation
  - Private include



ffx_<effect>.h

- Core file pulled in by your application
  - Directly or via `ffx_fx.h`
- Versioning information
- Pass definitions
- Context description structures
- Main accessors
  - `FfxErrorCode ffx<Effect>ContextCreate(Ffx<Effect>Context* pContext, const Ffx<Effect>ContextDescription* pContextDescription);`
  - `FfxErrorCode ffx<Effect>ContextDispatch(Ffx<Effect>Context* pContext, const Ffx<Effect>DispatchDescription* pDispatchDescription);`
  - `FfxErrorCode ffx<Effect>ContextDestroy(Ffx<Effect>Context* pContext);`

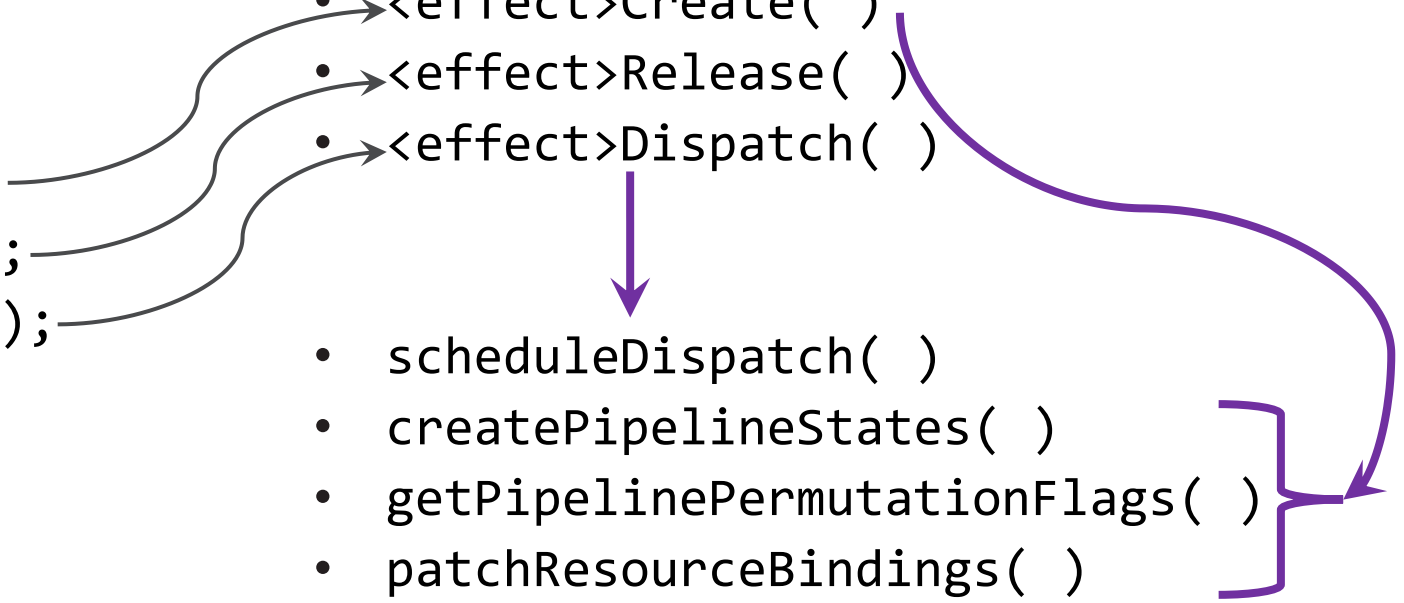
ffx_<effect>_private.h

- Shader permutation option definitions
- CPU-side constant buffer definitions
- Private effect context definitions

```
// FfxSpdContext_Private
// The private implementation of the SPD context.
typedef struct FfxSpdContext_Private
{
    FfxSpdContextDescription    contextDescription;
    FfxUInt32                   effectContextId;
    SpdConstants                constants;
    FfxDevice                   device;
    FfxDeviceCapabilities       deviceCapabilities;

    FfxPipelineState            pipelineDownsample;

    FfxResourceInternal srvResources[FFX_SPD_RESOURCE_IDENTIFIER_COUNT];
    FfxResourceInternal uavResources[FFX_SPD_RESOURCE_IDENTIFIER_COUNT];

} FfxSpdContext_Private;
```

ffx_<effect>.cpp

- **Implementation file**
- **Backs main API functions**
  - `ffx<Effect>ContextCreate( );`
  - `ffx<Effect>ContextDestroy( );`
  - `ffx<Effect>ContextDispatch( );`
- **Has all internal implementations**
  - `<effect>Create( )`
  - `<effect>Release( )`
  - `<effect>Dispatch( )`
    - `scheduleDispatch( )`
    - `createPipelineStates( )`
    - `getPipelinePermutationFlags( )`
    - `patchResourceBindings( )`



FIDELITYFX SDK

- Internals – Backends

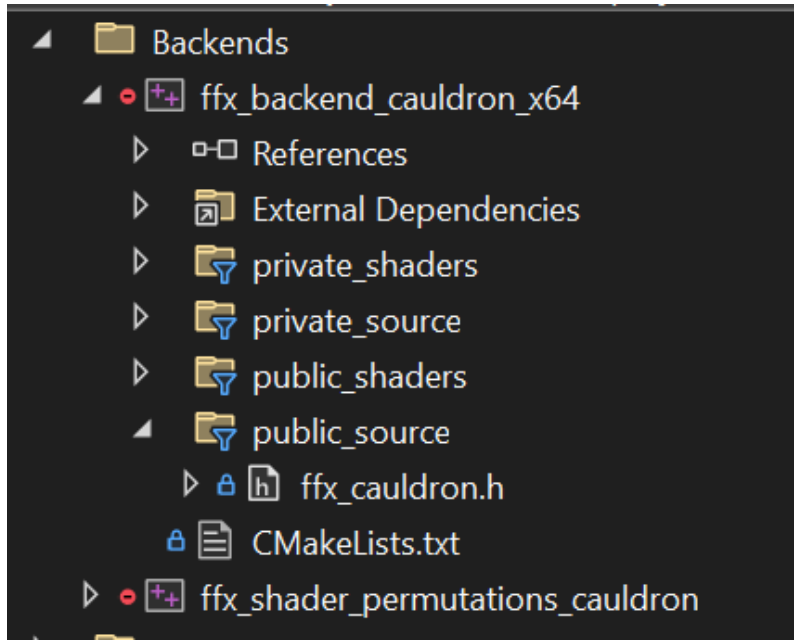
BACKENDS

- Core API/Platform support for resource creation and GPU workload execution
- Derived from FSR2's Interface model
- Backs API-specific versions of `FfxInterface` used by effect components

```
typedef struct FfxInterface {

    FfxCreateBackendContextFunc     fpCreateBackendContext;    ///< A callback function to create and initialize the backend context.
    FfxGetDeviceCapabilitiesFunc    fpGetDeviceCapabilities;   ///< A callback function to query device capabilities.
    FfxDestroyBackendContextFunc    fpDestroyBackendContext;   ///< A callback function to destroy the backend context.
    FfxCreateResourceFunc           fpCreateResource;          ///< A callback function to create a resource.
    FfxRegisterResourceFunc         fpRegisterResource;        ///< A callback function to register an external resource.
    FfxGetResourceFunc              fpGetResource;             ///< A callback function to convert an internal resource to external resource type
    FfxUnregisterResourcesFunc      fpUnregisterResources;     ///< A callback function to unregister external resource.
    FfxGetResourceDescriptionFunc   fpGetResourceDescription;  ///< A callback function to retrieve a resource description.
    FfxDestroyResourceFunc          fpDestroyResource;         ///< A callback function to destroy a resource.
    FfxCreatePipelineFunc           fpCreatePipeline;          ///< A callback function to create a render or compute pipeline.
    FfxDestroyPipelineFunc          fpDestroyPipeline;         ///< A callback function to destroy a render or compute pipeline.
    FfxScheduleGpuJobFunc           fpScheduleGpuJob;          ///< A callback function to schedule a render job.
    FfxExecuteGpuJobsFunc           fpExecuteGpuJobs;          ///< A callback function to execute all queued render jobs.

    void*                           scratchBuffer;             ///< A preallocated buffer for memory utilized internally by the backend.
    size_t                          scratchBufferSize;         ///< Size of the buffer pointed to by <c><i>scratchBuffer</i></c>.
    FfxDevice                       device;                    ///< A backend specific device

} FfxInterface;
```

ANATOMY OF A BACKEND

- Public source
  - `ffx_<backend>.h`
- Public shaders
  - All effect glsl/hlsl callback files
  - All effect resource files
  - All effect pass files (headers)
- Private source
  - All effect shader blob accessors
  - `ffx_<backend>.cpp`
- Private shaders
  - All effect glsl/hlsl pass entry point files
- `ffx_shader_permutations_<backend>`



FIDELITYFX SDK

- Internals – FidelityFX Shader Pre-compiler

FIDELITYFX SHADER PRE-COMPILER

- Open-source shader blob generation tool
- Builds all permutations of provided shader sources
- Currently supports
  - HLSL compile via DXC and FXC*
  - GLSL compile via glslang
- Copy to `sdk\tools\binary_store` with any needed dlls/libs/exes if modified

*FXC support for DX11 backend only

FIDELITYFX SHADER PRE-COMPILER - EXTENDING

- If new language support is needed:
  - Create a new compiler which inherits from `ICompiler`
  - Define your own structs which derive from `IShaderBinary`
  - Implement all pure virtual functions
    - `Compile`
    - `ExtractReflectionData`
    - `WriteBinaryHeaderReflectionData`
    - `WritePermutationHeaderReflectionStructMembers`
    - `WritePermutationHeaderReflectionData`
  - Handle language selection in command line parsing (`ffx_sc.cpp`)

CAULDRON 2.0

Cauldron

Application

Config
(Core Cauldron configuration files)

Icon
(Core application resources and icons)

Sample
(Shared sample source for simple sample setup)

(Core entry point and app manifest files)

Framework

Config
(Cauldron standalone configuration files)

Inc
(Cauldron public include files)

libs
(Cauldron third party libraries)

Src
(Cauldron private source files)
API abstraction code
Internal rendermodules
(UI, SwapChain, Tonemapping, Animation)

RenderModules

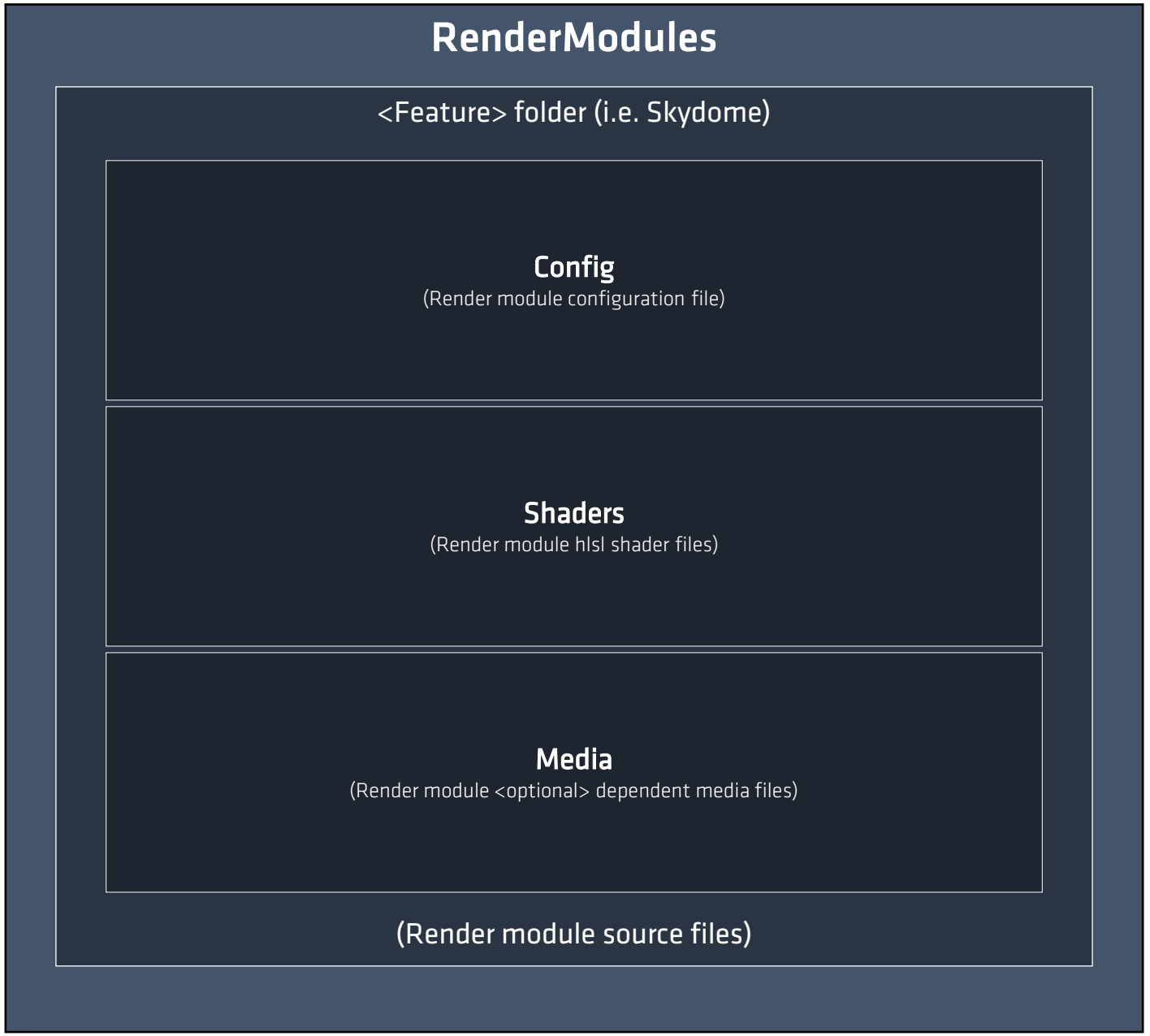
- Common
- GBuffer
- Lighting
- RasterShadows
- Skydome
- TAA
- Translucency

Effects

- Blur
- CACAO
- CAS
- DOF
- FSR
- LPM
- Hybrid Shadows
- Lens
- Parallel Sort
- SPD
- SSSR
- VRS

FFX_Cauldron
(Custom SDK backend implementation)

Media
(Deployed content that accompanies SDK samples)

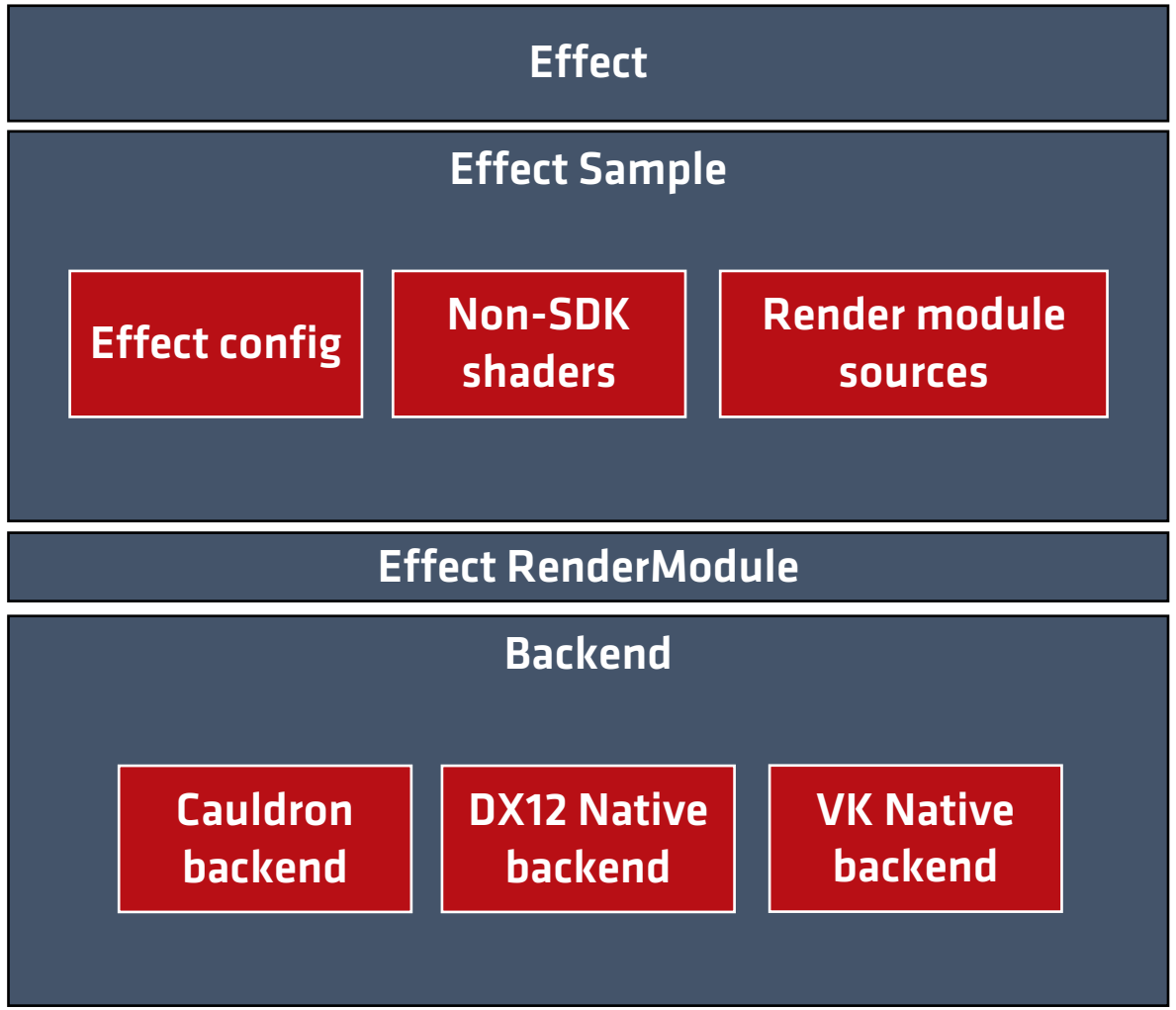

RenderModules
<Feature> folder (i.e. Skydome)
Config
(Render module configuration file)
Shaders
(Render module hlsl shader files)
Media
(Render module <optional> dependent media files)
(Render module source files)

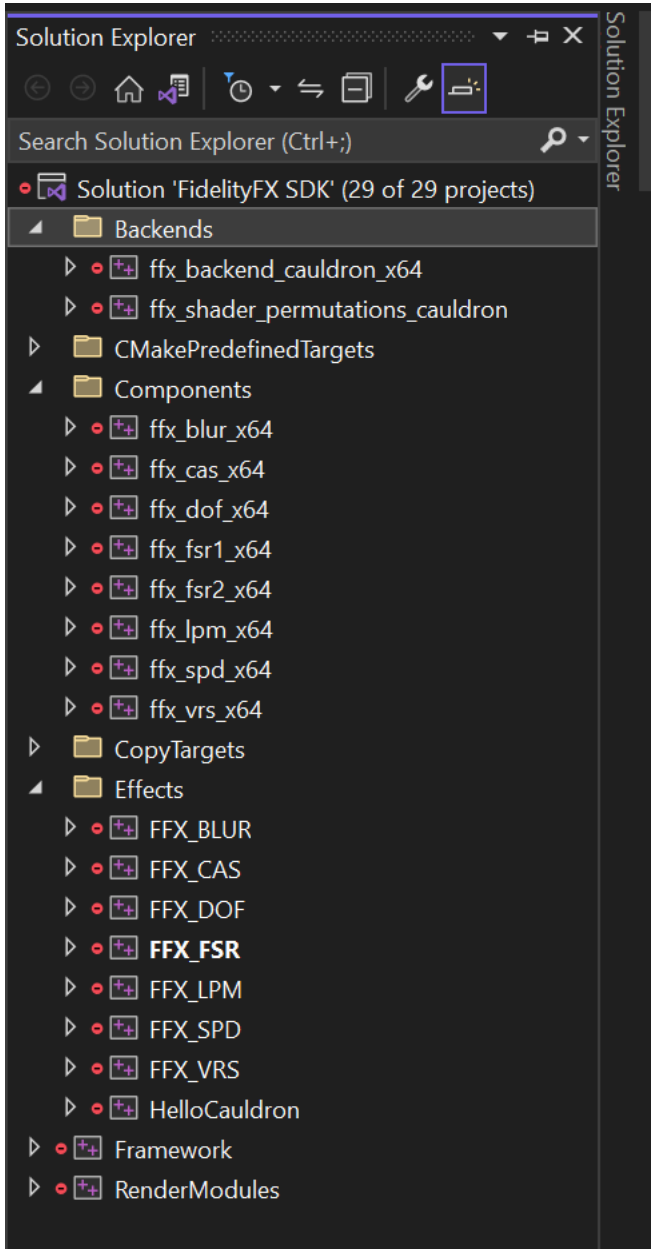

FidelityFX FSR

Future proofing at every iteration

Microsoft®
DirectX®

SDK SAMPLES


Effect
Effect Sample
Effect config
Non-SDK shaders
Render module sources
Effect RenderModule
Backend
Cauldron backend
DX12 Native backend
VK Native backend


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'FidelityFX SDK' (29 of 29 projects)
Backends
ffx_backend_cauldron_x64
ffx_shader_permutations_cauldron
CMakePredefinedTargets
Components
ffx_blur_x64
ffx_cas_x64
ffx_dof_x64
ffx_fsr1_x64
ffx_fsr2_x64
ffx_lpm_x64
ffx_spd_x64
ffx_vrs_x64
CopyTargets
Effects
FFX_BLUR
FFX_CAS
FFX_DOF
FFX_FSR
FFX_LPM
FFX_SPD
FFX_VRS
HelloCauldron
Framework
RenderModules

AMD FidelityFX Contrast Adaptive Sharpening

AMD FidelityFX HDR Mapper

AMD FidelityFX Denoiser

AMD FidelityFX Downsampler

AMD FidelityFX Hybrid Reflections

AMD FidelityFX Screen Space Reflections

AMD FidelityFX Super Resolution

AMD FidelityFX Variable Shading

AMD FidelityFX Hybrid Shadows

AMD FidelityFX Depth of Field

AMD FidelityFX Classifier

AMD FidelityFX Lens

AMD FidelityFX Ambient Occlusion

AMD FidelityFX Parallel Sort

AMD FidelityFX Super Resolution 2

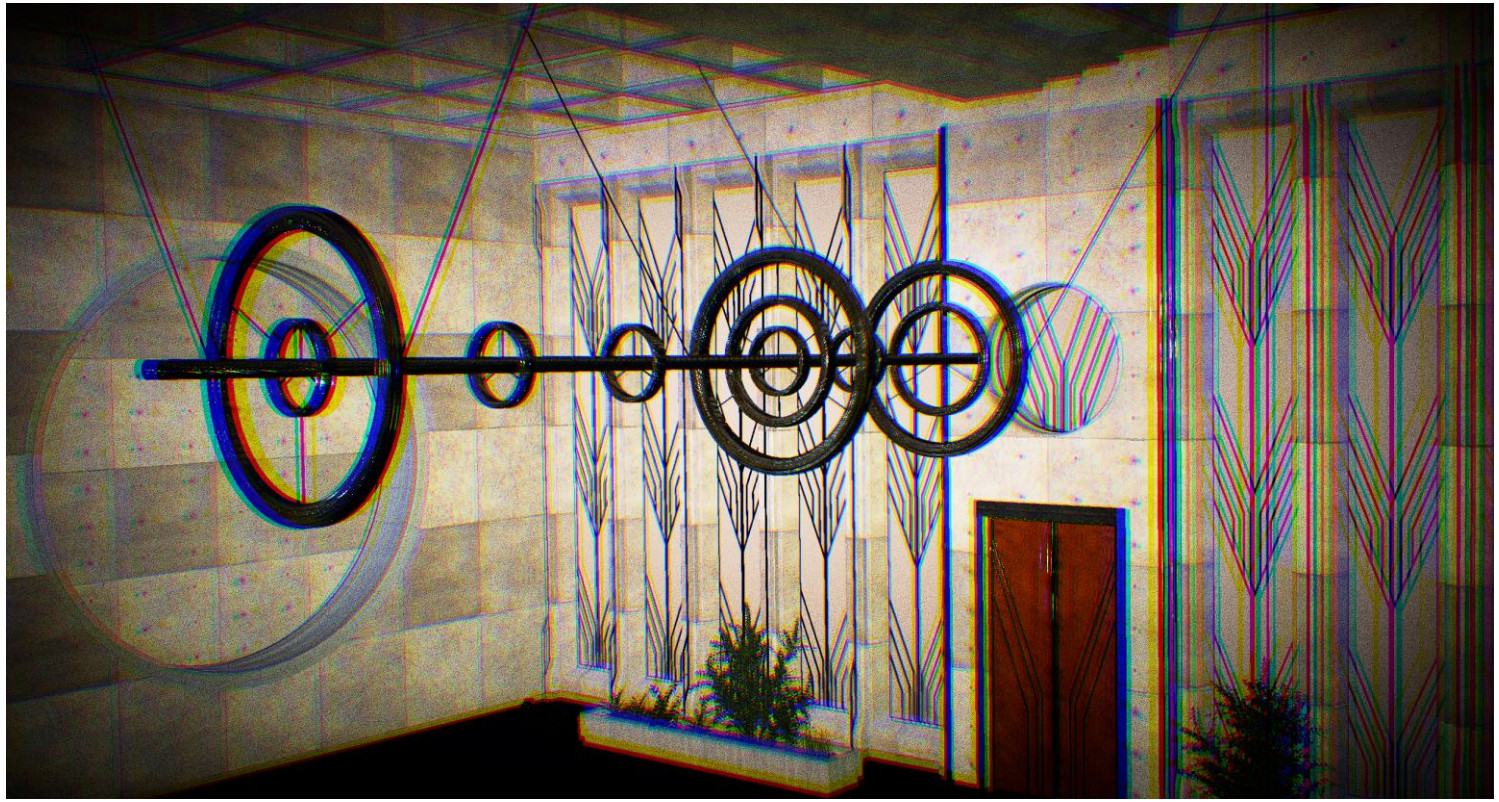
AMD FidelityFX Blur

DEPTH OF FIELD

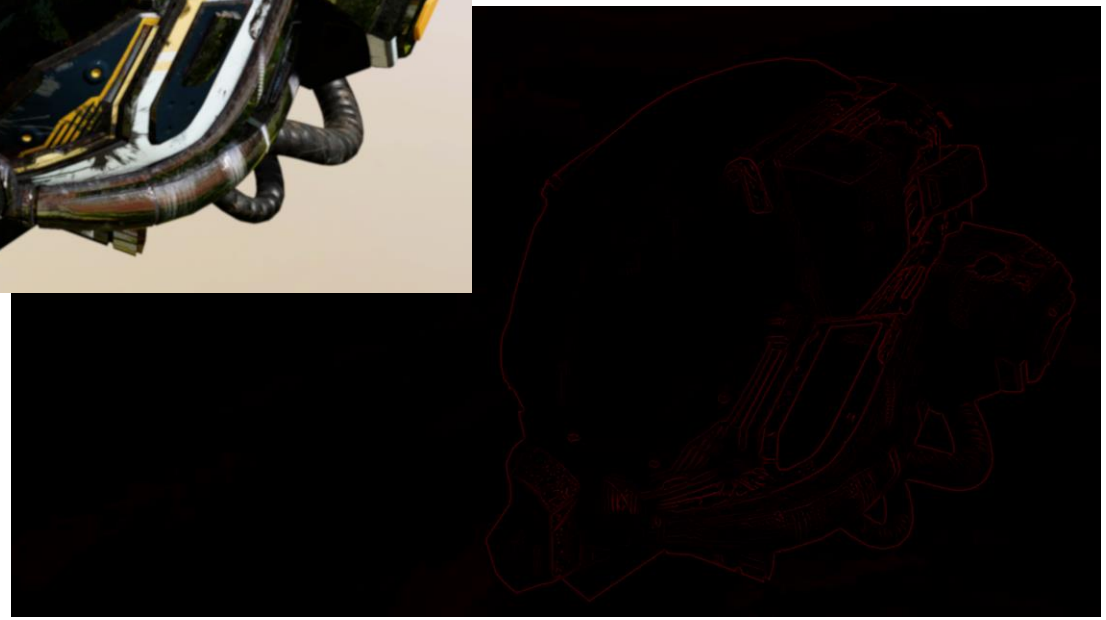
- Physically correct DoF
- Optimized for speed
- Tile-based classification for fast path selection
- Single pass near/far blur
- Single pass filter/upsample/composition

LENS

- Highly optimized
- Support for popular game lens effects
  - Film Grain
  - Chromatic Aberration
  - Vignette



BLUR



- Highly optimized blur kernels
  - 3x3 through 21x21
- Comparative tool for diff visualization
  - Compare traditional algorithms vs. FFX

WRAP UP

AMD

together we advance_

C99 APIs for all existing FidelityFX™ effects
Depth of Field (New)
Lens effects (New)
Blur (New)
FidelityFX™ shader pre-compiler

1.0
Q2 2023

1.1
Q3 2023

Xbox GDK backend
New FidelityFX effects
Cauldron improvements
Scene interaction support

1.2
Q4 2023

Feature improvements
New FidelityFX effects

Full source code is available on GitHub under a genuine open source license.

Use FidelityFX™ SDK however you like.


FidelityFX™ technologies are **trusted** by some of the greatest game developers on the planet.

In fact, we're **used in over 250 games**.

AMD

together we advance_ gaming

- Same great technology. Cleaned up and simplified.
- The future platform for all future FidelityFX technology.
- Three new effects and graphics building blocks.
- Optional easy-to-use C APIs for every effect.
- Modular backends that put you in control.
- Massively improved documentation.
- Great new shader compilation tool.

SPECIAL THANKS

- **FidelityFX SDK is the cumulative work of many people at AMD:**

Alex Pecoraro
Bernat Garcia
FangChen Yang
Ihor Szlachtycz
Jun Kamoshima
Li Qiu
Minesh Parekh
Pierre Yves Boers
Rys Sommefeldt
Stuart Adams
Ameya Gadkari
Colin Riley
François Guthmann
Jooho Jeong
Marek Machlinski
Nick Thibieroz
Piotr Koziol
Stephan Hodes
Tom Lewis
Aurélien Sérandour
Fabio Camaiora
Hao Zheng
Joseph Klinger
Meith Jhaveri
Oskar Homburg
Rosanna Ashworth-Jones
Steven Tovey
Yifan Fang

THANK YOU

- Please look forward to the public release of the **FidelityFX SDK** in Q2 2023.

DISCLAIMER

DISCLAIMERS

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18

Use of third-party marks / products is for informational purposes only and no endorsement of or by AMD is intended or implied. GD-83