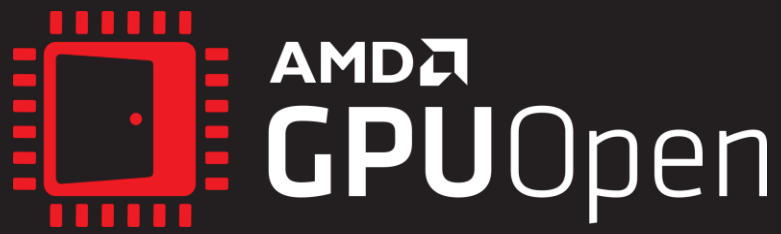
FFX CACAO - API

JAY FRASER



```
#ifndef FFX_CACAO_ENABLE_PROFILING
#define FFX_CACAO_ENABLE_PROFILING 0
#endif

#ifndef FFX_CACAO_ENABLE_D3D12
#define FFX_CACAO_ENABLE_D3D12 1
#endif
```

FFX_CACAO

- First in the ffx_cacao.h header disable/enable desired features by defining the FFX_CACAO_ENABLE_* macros as 0 or 1 respectively.

FFX_CACAO INITIALISATION

To initialise FFX CACAO, the input and output buffers must be passed to FFX CACAO. The buffers required are as follows:

- A depth buffer with inverse depths from the scene as calculated by the projection matrix.
- An optional normal buffer for the scene.
- A single channel output buffer.

All input and output buffers must be the same size.

```
#include “ffx_cacao.h”

size_t contextSize = ffxCacaoD3D12GetContextSize();
FfxCacaoD3D12Context *context = (FfxCacaoD3D12Context*)malloc(contextSize);

ID3D12Device *device = ...;
FfxCacaoStatus status = ffxCacaoD3D12InitContext(context, device);
if (status != FFX_CACAO_STATUS_OK) {
        // handle errors...
}

FfxCacaoD3D12ScreenSizeInfo ssi = {};
ssi.width = ...;
ssi.height = ...;
ssi.depthBufferResource = ...; // ID3D12Resource* for the depth buffer
ssi.depthBufferSrvDesc = ...;
ssi.normalBufferResource = ...; // Optional ID3D12Resource* for the normal buffer
ssi.normalBufferSrvDesc = ...;
ssi.outputResource = ...; // ID3D12Resource* for the output buffer
ssi.outputUavDesc = ...;

FfxCacaoBool downsampleSsao = ...;
status = ffxCacaoD3D12InitScreenSizeDependentResources(context, &ssi, downsampleSsao);
if (status != FFX_CACAO_STATUS_OK) {
        // handle errors...
}
```

FFX_CACAO INITIALISATION

- Allocate a CACAO context by getting its size with ffxCacaoD3D12GetContextSize
- Initialise the context with ffxCacaoD3D12InitContext and passing an ID3D12Device*.
- Fill in an FfxCacaoD3D12ScreenSizeInfo struct
- Initialise screen size dependent resources with ffxCacaoD3D12InitScreenSizeDependentResources

```
// Delete current screen size dependent resources
status = ffxCacaoD3D12DestroyScreenSizeDependentResources(context);
if (status != FFX_CACAO_STATUS_OK) {
        // handle errors...
}

// Recreate screen size dependent resources
FfxCacaoD3D12ScreenSizeInfo ssi = {};
ssi.width = ...;
ssi.height = ...;
ssi.depthBufferResource = ...;
ssi.depthBufferSrvDesc = ...;
ssi.normalBufferResource = ...;
ssi.normalBufferSrvDesc = ...;
ssi.outputResource = ...;
ssi.outputUavDesc = ...;

FfxCacaoBool downsampleSsao = ...;
status = ffxCacaoD3D12InitScreenSizeDependentResources(context, &ssi, downsampleSsao);
if (status != FFX_CACAO_STATUS_OK) {
        // handle errors...
}
```

FFX_CACAO SCREEN SIZE CHANGE

- For screen/window size changes, first free screen size dependent resources with ffxCacaoD3D12DestroyScreenSizeDependentResources then recreate resources with ffxCacaoD3D12InitScreenSizeDependentResources

```
FfxCacaoSettings settings = {};
settings.radius = ...; // world space radius of occlusion sphere
settings.shadowMultiplier = ...; // effect strength multiplier
settings.shadowPower = ...; // effect strength power modifier
settings.shadowClamp = ...; // effect strength max limit (applied after multiplier)
settings.horizonAngleThreshold = ...; // minimum angle between surface normal and occlude
settings.fadeOutFrom = ...; // world space distance to start fading out the effect
settings.fadeOutTo = ...; // world space distance to finish fading out the effect
settings.qualityLevel = ...; // effect quality, affects number of taps etc
settings.adaptiveQualityLimit = ...; //
settings.blurPassCount = ...; // number of edge sensitive blurs to apply
settings.sharpness = ...; // how much to bleed over edges
settings.temporalSupersamplingAngleOffset = ...; // a sampling angle offset which may be
updated per frame to help with temporal supersampling
settings.temporalSupersamplingRadiusOffset = ...; // a sampling radius offset which may be
updated per frame to help with temporal supersampling
settings.detailShadowStrength = ...; // adds detail using neighbouring pixel depths
settings.generateNormals = ...; // set this option to true to generate a normal buffer from
the depth buffer, or false if a normal buffer is provided
settings.bilateralSigmaSquared = ...; // Gaussian blur parameter for bilateral upsampler.
Smaller values correspond to less blur
settings.bilateralSimilarityDistanceSigma = ...; // similarity parameter for bilateral
upsampler. Smaller values make the blur bleed less over edges

status = ffxCacaoD3D12UpdateSettings(context, &settings);
if (status != FFX_CACAO_STATUS_OK) {
        // handle errors
}
```

FFX_CACAO UPDATE SETTINGS

- Settings can be set/updated by filling in an FfxCacaoSettings struct and calling ffxCacaoD3D12UpdateSettings
- Settings changed by using this function may safely be changed per frame

FFX_CACAO DRAW

The draw call for FFX CACAO requires two matrices as input:

- The matrix “proj”, which is the projection matrix used for rendering the scene.
- The matrix “normalsToView”, which is used to transform the normals in the normal buffer to viewspace

```
// Projection matrix in row major order
FfxCacaoMatrix4x4 proj = {
        { { p._00, p._10, p._20, p._30 },
          { p._01, p._11, p._21, p._31 },
          { p._02, p._12, p._22, p._32 },
          { p._03, p._13, p._23, p._33 } }
};

FfxCacaoMatrix4x4 normalsToView = ...; // matrix to transform normals in the normal buffer
to viewspace

status = ffxCacaoD3D12Draw(context, commandList, &proj, &normalsToView);
if (status != FFX_CACAO_STATUS_OK) {
        // handle errors...
}
```

FFX_CACAO DRAW

- Allocate a CACAO context by getting its size with ffxCacaoD3D12GetContextSize
- Initialise the context with ffxCacaoD3D12InitContext and passing an ID3D12Device*.
- Fill in an FfxCacaoD3D12ScreenSizeInfo struct
- Initialise screen size dependent resources with ffxCacaoD3D12InitScreenSizeDependentResources

```
FfxCacaoDetailedTiming timings;
status = ffxCacaoD3D12GetDetailedTimings(context, &timings);
if (status != FFX_CACAO_STATUS_OK) {
        // handle error...
}

for (int i = 1; i < timings.numTimestamps; ++i) {
        printf("Stage '%s' took %llu ticks.\n",
               timings.timestamps[i].label,
               timigs.timestamps[i].ticks);
}

printf("Total ticks taken: %llu.\n", timigs.timestamps[0].ticks);
```

FFX_CACAO GETTING TIMINGS

- Detailed timings for CACAO can be retrieved with the ffxCacaoD3D12GetDetailedTimings function. This returns an array of timings for different stages of CACAO from the previous frame. The timing in entry 0 is guaranteed to be the total time in GPU ticks that CACAO took in the previous frame.

```
status = ffxCacaoD3D12DestroyScreenSizeDependentResources(context);
if (status != FFX_CACAO_STATUS_OK) {
        // handle errors...
}

FfxCacaoStatus status = ffxCacaoD3D12DestroyContext(context);
if (status != FFX_CACAO_STATUS_OK) {
        // handle errors...
}
```

FFX_CACAO FINALISATION

- To correctly finalise CACAO, you must first destroy screen size dependent resources, and then destroy the CACAO context.