Post Processing

Description

Post processing is organized as a modifiable post process stack. Individual stages of the stack can be turned on and off. The order in which the stages are applied can be changed.

Associated classes and structures

PostFXChief	Class that holds active post process stack and does post process rendering
PostFX	Base class for various post process effects
PostFXData	Structure that holds common post fx settings
PFX_1DLUTColorCorrection	FX that does 1D texture look up color correction
PFX_3DLUTColorCorrection	FX that does 3D texture look up color correction
PFX_AnaglyphComposite	FX that composits 2 Stereo images to be viewed with anaglyph glasses.
PFX_BlackWhiteColorCorrection	FX that desaturates the image with black and white image at its extreme settings.
PFX_BrightnessContrast	FX that adjusts brightness & contrast
PFX_BrightPass	FX that lets bright pixel through with certain threshold
PFX_CameraMotionBlur	FX that performs camera space motion blur
PFX_Combine	FX that combines 2 images using configurable blend mode and color write mask
PFX_Copy	FX that copies one image to another
PFX_DirectionalBlur	FX that blurs image in horizontal or vertical direction
PFX_DirectionalStreaks	FX that draws directional light streaks
PFX_DOFExtractNear	FX that extracts near pixel for near Depth of Field effect bluring
PFX_ExplosionBlur	FX that directionally blurs the screen. Direction is typically calculated from the centre of the closest explosion
PFX_ExtractBloom	FX that extracts pixels for bloom effect
PFX_ExtractGlow	FX that extracts pixels for glow effect
PFX_Fill	FX that fills the image with solid color with configurable color write mask.
PFX_FilmGrain	FX that emulates film noise
PFX_FXAA	FX that performs Fast Approximate Anti Aliasing
PFX_GammaCorrect	FX that performs gamma correction
PFX_GodRays	FX that draws god rays
PFX_Interpolate	FX that interpolates one screen with another
PFX_MinExpand	FX that attempts to expand minimal values to adjacent pixels
PFX_MLAA_DiscontMap	FX that performs discontinuity mapping for rendering morphological anti aliasing effect.

PFX_NightVision	FX that renders night vision effect
PFX_ObjectMotionBlur	FX that performs motion blur according to object's velocities
PFX_RadialBlur	FX that performs radial blur
PFX_ScopeEffect	FX that applies scope texture to the screen
PFX_SeedSunThroughStencil	FX that seeds sun through geometry stencil for God Rays effect
PFX_StencilToMask	FX that converts stencil to alpha mask
PFX_StereoReproject	FX that uses reprojection technique in order to produce 2 stereo images out of 1 image and its depth.
PFX_SunGlare	FX that renders sun glare (camera lens) effect
PFX_Transform	FX that does Multiply-Add transform with the image

Associated Source Files

PostFXChief.h	PostFXChief class header
PostFXChief.cpp	PostFXChief class implementation
PostFX.h	PostFX class header
PostFX.cpp	PostFX class implementation
PFX_1DLUTColorCorrection.h	PFX_1DLUTColorCorrection class header
PFX_1DLUTColorCorrection.cpp	GrassGen class implementation
PFX_3DLUTColorCorrection.h	GrassMap class header
PFX_3DLUTColorCorrection.cpp	GrassMap class implementation
PFX_AnaglyphComposite.h	PFX_AnaglyphComposite class header
PFX_AnaglyphComposite.cpp	PFX_AnaglyphComposite class implementation
PFX_BlackWhiteColorCorrection.h	PFX_BlackWhiteColorCorrection class header
PFX_BlackWhiteColorCorrection.cpp	PFX_BlackWhiteColorCorrection class implementation
PFX_BrightnessContrast.h	PFX_BrightnessContrast class header
PFX_BrightnessContrast.cpp	PFX_BrightnessContrast class implementation
PFX_BrightPass.h	PFX_BrightPass class header
PFX_BrightPass.cpp	PFX_BrightPass class implementation
PFX_CameraMotionBlur.h	PFX_CameraMotionBlur class header
PFX_CameraMotionBlur.cpp	PFX_CameraMotionBlur class implementation
PFX_Combine.h	PFX_Combine class header
PFX_Combine.cpp	PFX_Combine class implementation
PFX_Copy.h	PFX_Copy class header
PFX_Copy.cpp	PFX_Copy class implementation

PFX_DirectionalBlur.h	PFX_DirectionalBlur class header
PFX_DirectionalBlur.cpp	PFX_DirectionalBlur class implementation
PFX_DirectionalStreaks.h	PFX_DirectionalStreaks class header
PFX_DirectionalStreaks.cpp	PFX_DirectionalStreaks class implementation
PFX_DOFExtractNear.h	PFX_DOFExtractNear class header
PFX_DOFExtractNear.cpp	PFX_DOFExtractNear class implementation
PFX_ExplosionBlur.h	PFX_ExplosionBlur class header
PFX_ExplosionBlur.cpp	PFX_ExplosionBlur class implementation
PFX_ExtractBloom.h	PFX_ExtractBloom class header
PFX_ExtractBloom.cpp	PFX_ExtractBloom class implementation
PFX_ExtractGlow.h	PFX_ExtractGlow class header
PFX_ExtractGlow.cpp	PFX_ExtractGlow class implementation
PFX_Fill.h	PFX_Fill class header
PFX_Fill.cpp	PFX_Fill class implementation
PFX_FilmGrain.h	PFX_FilmGrain class header
PFX_FilmGrain.cpp	PFX_FilmGrain class implementation
PFX_FXAA.h	PFX_FXAA class header
PFX_FXAA.cpp	PFX_FXAA class implementation
PFX_GammaCorrect.h	PFX_GammaCorrect class header
PFX_GammaCorrect.cpp	PFX_GammaCorrect class implementation
PFX_GodRays.h	PFX_GodRays class header
PFX_GodRays.cpp	PFX_GodRays class implementation
PFX_Interpolate.h	PFX_Interpolate class header
PFX_Interpolate.cpp	PFX_Interpolate class implementation
PFX_MinExpand.h	PFX_MinExpand class header
PFX_MinExpand.cpp	PFX_MinExpand class implementation
PFX_MLAA_DiscontMap.h	PFX_MLAA_DiscontMap class header
PFX_MLAA_DiscontMap.cpp	PFX_MLAA_DiscontMap class implementation
PFX_NightVision.h	PFX_NightVision class header
PFX_NightVision.cpp	PFX_NightVision class implementation
PFX_ObjectMotionBlur.h	PFX_ObjectMotionBlur class header
PFX_ObjectMotionBlur.cpp	PFX_ObjectMotionBlur class implementation
PFX_RadialBlur.h	PFX_RadialBlur class header
PFX_RadialBlur.cpp	PFX_RadialBlur class implementation
PFX_ScopeEffect.h	PFX_ScopeEffect class header
PFX_ScopeEffect.cpp	PFX_ScopeEffect class implementation
PFX_SeedSunThroughStencil.h	PFX_SeedSunThroughStencil class header

PFX_SeedSunThroughStencil.cpp	PFX_SeedSunThroughStencil class implementation
PFX_StencilToMask.h	PFX_StencilToMask class header
PFX_StencilToMask.cpp	PFX_StencilToMask class implementation
PFX_StereoReproject.h	PFX_StereoReproject class header
PFX_StereoReproject.cpp	PFX_StereoReproject class implementation
PFX_SunGlare.h	PFX_SunGlare class header
PFX_SunGlare.cpp	PFX_SunGlare class implementation
PFX_Transform.h	PFX_Transform class header
PFX_Transform.cpp	PFX_Transform class implementation

class PostFXChief

Summary

Class that holds active post process stack and does post process rendering

Important methods

---

void Init()

Summary:

Initializes post processing system.

---

void Close()

Summary:

Frees all resources allocated by post processing stage.

---

void AddFX(PostFX& fx, RTType dest, RTType src)

Summary:

Adds post fx *fx* to post processing chain. Post processing chain is formed starting from empty chain each frame. Thus the desired effect has to be added at every frame.

Parameters:

fx	- post fx to add
dest	- destination render target
src	- source render target

RTType may be one of the following

RTT_PINGPONG_LAST	- ping ponged render target that was used as previous output
RTT_PINGPONG_NEXT	- ping ponged render target that should be used as current output for ping pong operation
RTT_PINGPONG_LAST_AS_TEMP	- last ping ponged render target when it is desired to be used as temporary – not for ping pong operation
RTT_PINGPONG_NEXT_AS_TEMP	- ping ponged render target that is to be used as next output when it is actually used as temporary – not for ping pong operation
RTT_FULL0	- Full size render target 0. It is simultaneously either *RTT_PINGPONG_LAST or RTT_PINGPONG_NEXT*
RTT_FULL1	- Full size render target 1. It is simultaneously either *RTT_PINGPONG_LAST or RTT_PINGPONG_NEXT*
RTT_TEMP0	- Additional full size temporary render target
RTT_HALVED0	- Half size render target 0 (half the screen width and height)
RTT_HALVED1	- Half size render target 1
RTT_ONEFOURTH0	- One forth the screen sized render target 0.
RTT_ONEFOURTH1	- One forth the screen sized render target 1.
RTT_ONEFOURTH2	- One forth the screen sized render target 2.
RTT_DIFFUSE	- Diffuse render target out of deferred rendering pipeline output. To be used as input only.
RTT_DEPTH	- Depth render target out of deferred rendering pipeline output. To be used as input only.
RTT_AUX	- Auxiliary render target out of deferred rendering pipeline output.
RTT_MLAA_LINES_H	- Helper render target for MLAA effect
RTT_MLAA_LINES_V	- Helper render target for MLAA effect
RTT_NORMALS	- Normal render target out of deferred rendering pipeline output. To be used as input only.
RTT_FLASHBANG_MULTIFRAME	- Render targets that holds frame captured for flash bang effect during flash explosion
RTT_DISTORTION	- Render target that is used for rendering distortion during transparent object rendering stage

---

void AddClear(DWORD color, RTType dest)

Summary:

Adds clear operation to the post processing stack.

Parameters:

color	- color to clear render target with
dest	- render target to clear

---

void AddSwapBuffers()

Summary:

Adds swap buffers operation to the post processing stack. The following pointers to the following render target types get swapped:

RTT_PINGPONG_LAST and *RTT_PINGPONG_NEXT,*
RTT_PINGPONG_LAST_AS_TEMP and *RTT_PINGPONG_NEXT_AS_TEMP*

---

void AddGrabScreen(r3dScreenBuffer* target, RTType source)

Summary:

Adds screen grabbing operation to the post processing stage. This operation can be used to grab screenshots before color correction occurs in the post processing stage. Such screenshot can be then appended reference color strips, and then modified in external graphic tool along with the reference color strips. Afterwards, 3D color correction texture can be build using it.

Parameters:

target	- render target to grab screenshots to
source	- render target type to grab screenhot from

---

void Execute(bool toBackBuffer, bool resetTargets)

Summary:

Executes post processing commands that where previously appended to post processing stack.

Parameters:

toBackBuffer	- **true** if final operation is to be performed to the back buffer, **false** otherwise
resetTargets	- **true** if render target pointers need to be reset to their initial state, **false** otherwise. When render target pointers are reset, *RTT_PINGPONG_LAST* becomes equal to *RTT_FULL0 ,* and *RTT_PINGPONG_NEXT* becomes equal to *RTT_FULL1*

---

int GetDefaultVSId() const

Summary:

Retrieves default vertex shader id to use with post processing.

Int GetRestoreWVSId() const

Summary:

Retrieves id of the vertex shader, outputs of which allow restoring world position using depth render target in pixel shader.

---

r3dFilter GetZeroTexStageFilter() const

Summary:

Retrieves default filter for texture at zero stage. This stage is idiomatically assigned either point, or bi-linear filtering depending on relationship between sizes of the input and output render targets. If a postprocess needs to override this filter, it is expected to restore the filter to its default state after it has done executing. Failure to do so will result in assertion failure.

---

void BindBufferTexture(RTType type, int stage)

Summary:

Binds one of the render targets enumerated in *RTType* to stage *stage.*

Parameters:

type	- type of the render target to bind
stage	- stage to bind render target to

---

void SetDefaultTexAddressMode(int stage)

Summary:

Sets default address mode for for texture stage *stage.* If address mode has been changed by a post process, it is expected to be restored to its default state after post process finishes execution. Incorrect texture address mode will result in assertion failure.

Parameters:

stage	- stage to set default address mode for.

---

template< int Stage >
void SetDefaultFiltering()

Summary:

Sets defaults texture filtering for stage *Stage*. If texture filtering is changed by the post process, it is expected to be restored to its original state after post process execution. If this is not done assertion failure will occur.

Parameters:

Stage	- texture stage to set default filtering for. Passed as template parameter.

---

r3dScreenBuffer* GetBuffer(RTType type) const

Summary:

Retrieves render target corresponding to type *type.*

Return value:

Render target for type *type*

---

class PostFX

Summary

Base class for post process effect implementation.

Important methods

void Init()

Summary:

Initializes post fx. Calls virtual function *InitImpl*, which has to be implementation by child classes.

---

void Close()

Summary:

Releases resources allocated by post fx. Calls virtual function *CloseImpl*, which has to be implemented by child classes.

---

const PostFXData& Prepare(r3dScreenBuffer* dest, r3dScreenBuffer* src)

Summary:

Prepares post fx to be executed. Sets up default states of *PostFXData* structure. Calls virtual *PrepareImpl*, which should apply additional changes.

Parameters:

dest	- render target, to which current post process is about to render
src	- source render target. Note: source render target may be actually unused. Post fx may set up additional source targets by itself

Return value:

PostFXData structure which is filled by this function and by *PrepareImpl.*

---

void FinishImpl()

Summary:

Finishes post fx execution. Calls *FinishImpl* which should be implemented by child classes.

---

void PushDefaultSettings()

Summary:

In case post fx is used multiple times in the same frame with different settings, it should support settings stack. PushDefaultSettings is called in order to push default settings into this stack, in case the user himself specified no settings. In case no user settings where pushed, *PushDefaultSettings* calls *PushDefaultSettingsImpl*, which should be implemented by child classes.

---

const char* GetName() const

Summary:

Returns the name of this post fx

Return value:

Name of the post fx.

---

struct PostFXData

Summary

Describes render state changes to be applied before post fx is rendered.

Structure Fields

PixelShaderID	*int*	Id of the pixel hader to set
VertexShaderID	*int*	Id of the vertex hader to set. Can be -1 indicating the need to set default vertex shader
VSType	*EVSType*	The type of the vertex shader to set. Can be one of the following: *VST_DEFAULT* -set default vertex shader. *VertexShaderID* field is ignored in this case. *VST_RESTORE_W* - set vertex shader that provides outputs for easier world position reconstruction using the depth texture. *VertexShaderID* field is ignored in this case. *VST_CUSTOM* – set vertex custom vertex shader. *VertexShaderID* is expected to be correct shader id in this case.
TexTransform	*float[4]*	Texture transformation to apply. These are 4 components for Multiply-Add operation to be performed on original texture .xy components.