XShaderCompiler

0.02 Alpha

Generated by Doxygen 1.8.11

Contents

Chapter 1

Main Page

Welcome to the XShaderCompiler, Version 0.02 Alpha

Here is a quick start example:

```
#include <Xsc/Xsc.h>
#include <fstream>

int main()
{
    // Open input and output streams
    auto inputStream = std::make_shared<std::ifstream>("Example.hlsl");
    std::ofstream outputStream("Example.vertex.glsl");

    // Initialize shader input descriptor structure
    Xsc::ShaderInput inputDesc;
    {
        inputDesc.sourceCode     = inputStream;
        inputDesc.shaderVersion  = Xsc::InputShaderVersion::HLSL5
      ;
        inputDesc.entryPoint     = "VS";
        inputDesc.shaderTarget   = Xsc::ShaderTarget::VertexShader
      ;
    }

    // Initialize shader output descriptor structure
    Xsc::ShaderOutput outputDesc;
    {
        outputDesc.sourceCode    = &outputStream;
        outputDesc.shaderVersion =
      Xsc::OutputShaderVersion::GLSL330;
    }

    // Compile HLSL code into GLSL
    Xsc::StdLog log;
    bool result = Xsc::CompileShader(inputDesc, outputDesc, &log);

    // Show compilation status
    if (result)
        std::cout << "Compilation successful" << std::endl;
    else
        std::cerr << "Compilation failed" << std::endl;

    return 0;
}
```

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Xsc Namespace Reference

Main XShaderCompiler namespace.

Namespaces

- ConsoleManip

  Namespace for console manipulation.

Classes

- struct Formatting

  Formatting descriptor structure for the output shader.
- class IncludeHandler

  Interface for handling new include streams.
- class IndentHandler

  Indentation handler base class.
- class Log

  Log base class.
- struct Options

  Structure for additional translation options.
- class Report

  Report exception class.
- struct ShaderInput

  Shader input descriptor structure.
- struct ShaderOutput

  Shader output descriptor structure.
- struct Statistics

  Structure for shader output statistics (e.g. texture/buffer binding points).
- class StdLog

  Standard output log (uses std::cout to submit a report).

Enumerations

- enum ShaderTarget {
ShaderTarget::Undefined, ShaderTarget::VertexShader, ShaderTarget::TessellationControlShader, Shader↩
Target::TessellationEvaluationShader,
ShaderTarget::GeometryShader, ShaderTarget::FragmentShader, ShaderTarget::ComputeShader }

  Shader target enumeration.
- enum InputShaderVersion { InputShaderVersion::HLSL3 = 3, InputShaderVersion::HLSL4 = 4, InputShader↩
Version::HLSL5 = 5 }

  Input shader version enumeration.
- enum OutputShaderVersion {
OutputShaderVersion::GLSL110 = 110, OutputShaderVersion::GLSL120 = 120, OutputShaderVersion::GL↩
SL130 = 130, OutputShaderVersion::GLSL140 = 140,
OutputShaderVersion::GLSL150 = 150, OutputShaderVersion::GLSL330 = 330, OutputShaderVersion::GL↩
SL400 = 400, OutputShaderVersion::GLSL410 = 410,
OutputShaderVersion::GLSL420 = 420, OutputShaderVersion::GLSL430 = 430, OutputShaderVersion::GL↩
SL440 = 440, OutputShaderVersion::GLSL450 = 450,
OutputShaderVersion::GLSL = ∼0 }

  Output shader version enumeration.

Functions

- XSC_EXPORT std::string TargetToString (const ShaderTarget target)

  Returns the specified shader target as string.
- XSC_EXPORT std::string ShaderVersionToString (const InputShaderVersion shaderVersion)

  Returns the specified shader input version as string.
- XSC_EXPORT std::string ShaderVersionToString (const OutputShaderVersion shaderVersion)

  Returns the specified shader output version as string.
- XSC_EXPORT bool CompileShader (const ShaderInput &inputDesc, const ShaderOutput &outputDesc, Log ∗log=nullptr)

  Cross compiles the shader code from the specified input stream into the specified output shader code.

5.1.1 Detailed Description

Main XShaderCompiler namespace.

5.1.2 Enumeration Type Documentation

5.1.2.1 enum Xsc::InputShaderVersion [strong]

Input shader version enumeration.

Enumerator

HLSL3	HLSL Shader Model 3.0 (DirectX 9).
HLSL4	HLSL Shader Model 4.0 (DirectX 10).
HLSL5	HLSL Shader Model 5.0 (DirectX 11).

5.1.2.2 enum Xsc::OutputShaderVersion `[strong]`

Output shader version enumeration.

Enumerator

GLSL110 GLSL 1.10 (OpenGL 2.0).

Note

Currently not supported!

GLSL120 GLSL 1.20 (OpenGL 2.1).

Note

Currently not supported!

GLSL130 GLSL 1.30 (OpenGL 3.0).

GLSL140 GLSL 1.40 (OpenGL 3.1).

GLSL150 GLSL 1.50 (OpenGL 3.2).

GLSL330 GLSL 3.30 (OpenGL 3.3).

GLSL400 GLSL 4.00 (OpenGL 4.0).

GLSL410 GLSL 4.10 (OpenGL 4.1).

GLSL420 GLSL 4.20 (OpenGL 4.2).

GLSL430 GLSL 4.30 (OpenGL 4.3).

GLSL440 GLSL 4.40 (OpenGL 4.4).

GLSL450 GLSL 4.50 (OpenGL 4.5).

GLSL Auto-detect minimal required GLSL version.

5.1.2.3 enum Xsc::ShaderTarget `[strong]`

Shader target enumeration.

Enumerator

Undefined Undefined shader target.

VertexShader Vertex shader.

TessellationControlShader Tessellation-control (also Hull-) shader.

TessellationEvaluationShader Tessellation-evaluation (also Domain-) shader.

GeometryShader Geometry shader.

FragmentShader Fragment (also Pixel-) shader.

ComputeShader Compute shader.

5.1.3 Function Documentation

5.1.3.1 XSC_EXPORT bool Xsc::CompileShader (const ShaderInput & *inputDesc*, const ShaderOutput & *outputDesc*, Log ∗ *log* = `nullptr`)

Cross compiles the shader code from the specified input stream into the specified output shader code.

Parameters

in	*inputDesc*	Input shader code descriptor.
in	*outputDesc*	Output shader code descriptor.
in	*log*	Optional pointer to an output log. Inherit from the "Log" class interface.

Returns

True if the code has been translated successfully.

Exceptions

std::invalid_argument	If either the input or output streams are null.

See also

ShaderInput
ShaderOutput
Log

5.2 Xsc::ConsoleManip Namespace Reference

Namespace for console manipulation.

Classes

- struct ColorFlags

  Output stream color flags enumeration.
- class ScopedColor

  Helper class for scoped color stack operations.

Functions

- void XSC_EXPORT Enable (bool enable)

  Enables or disables console manipulation. By default enabled.
- bool XSC_EXPORT IsEnabled ()

  Returns true if console manipulation is enabled.
- void XSC_EXPORT PushColor (std::ostream &stream, long front)

  Push the specified front color onto the stack.
- void XSC_EXPORT PushColor (std::ostream &stream, long front, long back)

  Push the specified front and back color onto the stack.
- void XSC_EXPORT PopColor (std::ostream &stream)

  Pops the previous front and back colors from the stack.

5.2.1 Detailed Description

Namespace for console manipulation.

Chapter 6

Class Documentation

6.1 Xsc::Statistics::Binding Struct Reference

Public Attributes

- std::string ident
  Identifier of the binding point.
- int location
  Zero based binding point or location. If this is -1, the location has not been set explicitly.

The documentation for this struct was generated from the following file:

- Xsc.h

6.2 Xsc::ConsoleManip::ColorFlags Struct Reference

Output stream color flags enumeration.

```
#include <ConsoleManip.h>
```

Public Types

- enum {
  Red = (1 << 0), Green = (1 << 1), Blue = (1 << 2), Intens = (1 << 3),
  Black = 0, Gray = (Red | Green | Blue), White = (Gray | Intens), Yellow = (Red | Green | Intens),
  Pink = (Red | Blue | Intens), Cyan = (Green | Blue | Intens) }

6.2.1 Detailed Description

Output stream color flags enumeration.

6.2.2 Member Enumeration Documentation

6.2.2.1 anonymous enum

Enumerator

Red Red color flag.

Green Green color flag.

Blue Blue color flag.

Intens Intensity color flag.

Black Black color flag.

Gray Gray color flag (Red | Green | Blue).

White White color flag (Gray | Intens).

Yellow Yellow color flag (Red | Green | Intens).

Pink Pink color flag (Red | Blue | Intens).

Cyan Cyan color flag (Green | Blue | Intens).

The documentation for this struct was generated from the following file:

- ConsoleManip.h

6.3 Xsc::Formatting Struct Reference

Formatting descriptor structure for the output shader.

```
#include <Xsc.h>
```

Public Attributes

- std::string indent = " "

  Indentation string for code generation. By default std::string(4, ' ').
- std::string prefix = "xsc_"

  Prefix string for name mangling. By default "xsc_".
- bool blanks = true

  True if blanks are allowed. By default true.
- bool lineMarks = false

  True if line marks are allowed. By default false.

6.3.1 Detailed Description

Formatting descriptor structure for the output shader.

6.3.2 Member Data Documentation

6.3.2.1 std::string Xsc::Formatting::prefix = "xsc_"

Prefix string for name mangling. By default "xsc_".

Remarks

This prefix is used because GLSL does not allow interface blocks as input for vertex shaders or output for fragment shaders. Thus some identifiers of local variables may overlap with input variables. This prefix is added to all local function variables.

The documentation for this struct was generated from the following file:

- Xsc.h

6.4 Xsc::IncludeHandler Class Reference

Interface for handling new include streams.

`#include <IncludeHandler.h>`

Public Member Functions

- virtual std::unique_ptr< std::istream > Include (const std::string &filename, bool useSearchPathsFirst)
  Returns an input stream for the specified filename.

Public Attributes

- std::vector< std::string > searchPaths
  List of search paths.

6.4.1 Detailed Description

Interface for handling new include streams.

Remarks

The default implementation will read the files from an std::ifstream.

6.4.2 Member Function Documentation

6.4.2.1 virtual std::unique_ptr<std::istream> Xsc::IncludeHandler::Include (const std::string & *filename,* bool *useSearchPathsFirst*) [virtual]

Returns an input stream for the specified filename.

Parameters

in	*includeName*	Specifies the include filename.
in	*useSearchPathsFirst*	Specifies whether to first use the search paths to find the file.

Returns

Unique pointer to the new input stream.

The documentation for this class was generated from the following file:

- IncludeHandler.h

6.5 Xsc::IndentHandler Class Reference

Indentation handler base class.

```
#include <IndentHandler.h>
```

Public Member Functions

- **IndentHandler** (const std::string &initialIndent=std::string(2, ' '))
- void SetIndent (const std::string &indent)

  Sets the next indentation string. By default two spaces.
- void IncIndent ()

  Increments the indentation.
- void DecIndent ()

  Decrements the indentation.
- const std::string & FullIndent () const

  Returns the current full indentation string.

6.5.1 Detailed Description

Indentation handler base class.

The documentation for this class was generated from the following file:

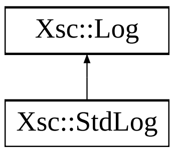
- IndentHandler.h

6.6 Xsc::Log Class Reference

Log base class.

```
#include <Log.h>
```

Inheritance diagram for Xsc::Log:



Public Member Functions

- virtual void SumitReport (const Report &report)=0

  Submits the specified report.
- void SetIndent (const std::string &indent)

  Sets the next indentation string. By default two spaces.
- void IncIndent ()

  Increments the indentation.
- void DecIndent ()

  Decrements the indentation.

Protected Member Functions

- const std::string & FullIndent () const

  Returns the current full indentation string.

6.6.1 Detailed Description

Log base class.

The documentation for this class was generated from the following file:

- Log.h

6.7 Xsc::Options Struct Reference

Structure for additional translation options.

```
#include <Xsc.h>
```

Public Attributes

- bool warnings = false

  True if warnings are allowed. By default false.
- bool optimize = false

  If true, little code optimizations are performed. By default false.
- bool preprocessOnly = false

  If true, only the preprocessed source code will be written out. By default false.
- bool validateOnly = false

  If true, the source code is only validated, but no output code will be generated. By default false.
- bool allowExtensions = false

  If true, the shader output may contain GLSL extensions, if the target shader version is too low. By default false.
- bool explicitBinding = false

  If true, explicit binding slots are enabled. By default false.
- bool preserveComments = false

  If true, commentaries are preserved for each statement. By default false.
- bool showAST = false

  If true, the AST (Abstract Syntax Tree) will be written to the log output. By default false.
- bool showTimes = false

  If true, the timings of the different compilation processes are written to the log output. By default false.

6.7.1 Detailed Description

Structure for additional translation options.

The documentation for this struct was generated from the following file:

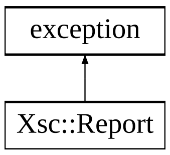
- Xsc.h

6.8 Xsc::Report Class Reference

Report exception class.

`#include <Report.h>`

Inheritance diagram for Xsc::Report:



Public Types

- enum Types { Types::Info, Types::Warning, Types::Error }

  Report types enumeration.

Public Member Functions

- **Report** (const Report &)=default
- Report & **operator=** (const Report &)=default
- **Report** (const Types type, const std::string &message, const std::string &context="")
- **Report** (const Types type, const std::string &message, const std::string &line, const std::string &marker, const std::string &context="")
- const char ∗ what () const override throw ()

  Overrides the 'std::exception::what' function.
- void TakeHints (std::vector< std::string > &&hints)

  Moves the specified hints into this report.
- Types Type () const

  Returns the type of this report.
- const std::string & Context () const

  Returns the context description string (e.g. a function name where the report occured). This may also be empty.
- const std::string & Message () const

  Returns the message string.
- const std::string & Line () const

  Returns the line string where the report occured. This line never has new-line characters at its end.
- const std::string & Marker () const

  Returns the line marker string to highlight the area where the report occured.
- const std::vector< std::string > & GetHints () const

  Returns the list of optional hints of the report.
- bool HasLine () const

  Returns true if this report has a line with line marker.

6.8.1 Detailed Description

Report exception class.

6.8.2 Member Enumeration Documentation

6.8.2.1 enum **Xsc::Report::Types** `[strong]`

Report types enumeration.

Enumerator

Info Standard information.
Warning Warning message.
Error Error message.

6.8.3 Member Function Documentation

6.8.3.1 bool Xsc::Report::HasLine ( ) const `[inline]`

Returns true if this report has a line with line marker.

See also

Line
Marker

The documentation for this class was generated from the following file:

- Report.h

6.9 Xsc::ConsoleManip::ScopedColor Class Reference

Helper class for scoped color stack operations.

`#include <ConsoleManip.h>`

Public Member Functions

- ScopedColor (std::ostream &stream, long front)
  Constructor with output stream and front color flags.
- ScopedColor (std::ostream &stream, long front, long back)
  Constructor with output stream, and front- and back color flags.
- ∼ScopedColor ()
  Destructor which will reset the previous color from the output stream.

6.9.1 Detailed Description

Helper class for scoped color stack operations.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Xsc::ConsoleManip::ScopedColor::ScopedColor (std::ostream & *stream*, long *front*) `[inline]`

Constructor with output stream and front color flags.

Parameters

`in,out`	*stream*	Specifies the output stream for which the scope is to be changed. This is only used for Unix systems.
`in`	*front*	Specifies the front color flags. This can be a bitwise OR combination of the entries of the ColorFlags enumeration.

See also

ColorFlags
PushColor(std::ostream&, long)

6.9.2.2 Xsc::ConsoleManip::ScopedColor::ScopedColor (std::ostream & *stream*, long *front*, long *back*) `[inline]`

Constructor with output stream, and front- and back color flags.

Parameters

`in,out`	*stream*	Specifies the output stream for which the scope is to be changed. This is only used for Unix systems.
`in`	*front*	Specifies the front color flags. This can be a bitwise OR combination of the entries of the ColorFlags enumeration.
`in`	*back*	Specifies the back color flags. This can be a bitwise OR combination of the entries of the ColorFlags enumeration.

See also

ColorFlags
PushColor(std::ostream&, long, long)

6.9.2.3 Xsc::ConsoleManip::ScopedColor::∼ScopedColor ( ) `[inline]`

Destructor which will reset the previous color from the output stream.

See also

PopColor

The documentation for this class was generated from the following file:

- ConsoleManip.h

6.10 Xsc::ShaderInput Struct Reference

Shader input descriptor structure.

```
#include <Xsc.h>
```

Public Attributes

- std::string filename

  Specifies the filename of the input shader code. This is an optional attribute, and only a hint to the compiler.
- std::shared_ptr< std::istream > sourceCode

  Specifies the input stream. This must be valid HLSL code.
- InputShaderVersion shaderVersion = InputShaderVersion::HLSL5

  Specifies the input shader version (e.g. InputShaderVersion::HLSL5 for "HLSL 5"). By default InputShaderVersion↩ ::HLSL5.
- std::string entryPoint

  Specifies the HLSL shader entry point.
- ShaderTarget shaderTarget = ShaderTarget::Undefined

  Specifies the target shader (Vertex, Fragment etc.). By default ShaderTarget::Undefined.
- IncludeHandler ∗ includeHandler = nullptr

  Optional pointer to the implementation of the "IncludeHandler" interface. By default null.

6.10.1 Detailed Description

Shader input descriptor structure.

6.10.2 Member Data Documentation

6.10.2.1 IncludeHandler∗ Xsc::ShaderInput::includeHandler = nullptr

Optional pointer to the implementation of the "IncludeHandler" interface. By default null.

Remarks

If this is null, the default include handler will be used, which will include files with the STL input file streams.

The documentation for this struct was generated from the following file:

- Xsc.h

6.11 Xsc::ShaderOutput Struct Reference

Shader output descriptor structure.

```
#include <Xsc.h>
```

Public Attributes

- std::string filename

  Specifies the filename of the output shader code. This is an optional attribute, and only a hint to the compiler.
- std::ostream $*$ sourceCode = nullptr

  Specifies the output stream. This will contain the output GLSL code. This must not be null when passed to the "CompileShader" function!
- OutputShaderVersion shaderVersion = OutputShaderVersion::GLSL

  Specifies the output shader version. By default OutputShaderVersion::GLSL (to auto-detect minimum required version).
- Formatting formatting

  Output code formatting descriptor.
- Options options

  Additional options to configure the code generation.
- Statistics $*$ statistics = nullptr

  Optional output statistics. By default null.

6.11.1 Detailed Description

Shader output descriptor structure.

The documentation for this struct was generated from the following file:

- Xsc.h

6.12 Xsc::Statistics Struct Reference

Structure for shader output statistics (e.g. texture/buffer binding points).

```
#include <Xsc.h>
```

Classes

- struct Binding

Public Attributes

- std::vector< std::string > macros

  All defined macros after pre-processing.
- std::vector< Binding > textures

  Texture bindings.
- std::vector< Binding > constantBuffers

  Constant buffer bindings.
- std::vector< Binding > fragmentTargets

  Fragment shader output targets.

6.12.1 Detailed Description

Structure for shader output statistics (e.g. texture/buffer binding points).

The documentation for this struct was generated from the following file:

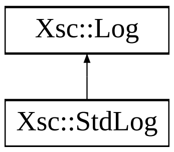
- Xsc.h

6.13 Xsc::StdLog Class Reference

Standard output log (uses std::cout to submit a report).

```
#include <Log.h>
```

Inheritance diagram for Xsc::StdLog:



Public Member Functions

- void SumitReport (const Report &report) override

  Implements the base class interface.
- void PrintAll (bool verbose=true, bool warnings=true)

  Prints all submitted reports to the standard output.

Additional Inherited Members

6.13.1 Detailed Description

Standard output log (uses std::cout to submit a report).

The documentation for this class was generated from the following file:

- Log.h

Index