

PixelLight Features



February 23, 2012
PixelLight 0.9.11-R1



The content of this PixelLight document is published under the Creative Commons
Attribution-NonCommercial-ShareAlike 3.0 Unported
Copyright © 2002-2012 by The PixelLight Team

Contents

1	Introduction	5
1.1	Application Programming Interface (API)	5
1.2	Documentation and Examples	5
1.3	PixelLight is not an...	6
1.4	General	6
1.5	Debugging and Tweaking Tools	7
2	Base	9
2.1	PLCore	9
2.1.1	Text Processing	9
2.1.2	System	10
2.1.3	Network	10
2.1.4	File System	10
2.1.5	Script	11
2.1.6	A huge set of Additional Tools	11
2.2	PLDatabase	12
2.3	PLGraphics	12
2.4	PLMath	12
2.5	PLGui	12
2.6	PLInput	13
2.7	PLRenderer	13
2.7.1	Texturing	14
2.7.2	Materials & Effects	15
2.8	PLMesh	15
2.8.1	Meshes	15
2.8.2	Animation System	16
2.9	PLScene	16
2.9.1	Scene System	16
2.9.2	Compositing System	17
2.10	PLSound	19
2.11	PLPhysics	19
2.12	PLEngine	19
3	Plugins	21
3.1	Frontends	21
3.2	File Formats	21

Contents

3.3	Particle Effects	22
3.4	Post Process Effects	23
3.5	Network	23
4	Tools	25
4.1	Autodesk 3ds Max Scene Exporter Plugin	25
5	Contact	27
	Abbreviations	29

1 Introduction

This is the PixelLight feature list which is split into the different project packages. This introduction will inform you about the general features. Please note that due the size of PixelLight, it's not possible to mention each and every single feature within this document.

1.1 API

- By design the framework is split up into an 'immediate' and a 'composed' part
- The 'immediate' functions make it possible to immediately display all kinds of resources (e.g. textures, meshes or sounds) on the screen. These low-level functions can be used for customization of the framework or for simplicity where no internal structures (like objects or scenes) are needed
- The 'composed' part actually contains the main framework features by defining structures for whole virtual scenes and objects reacting to each other. See 'Scene System' for detailed description

1.2 Documentation and Examples

- In general, the PixelLight Software Development Kit (SDK) comes with a huge set of documents and diagrams which will make your work with this technology much easier
- Quite comfortable SDK browser
- Separate documentation for each main component
- Detailed API documentation for programmers - automatically created with Doxygen¹
- A huge set of different example programs and scenes which will demonstrate different things
- More documentation, tutorials, Frequently Asked Questions (FAQ) and so on can be found within the official PixelLight Wiki²

¹<http://www.doxygen.org/>

²https://developer.pixellight.org/wiki/index.php/Main_Page

1.3 PixelLight is not an...

- Pure Click'n'Play toolkit - there are several useful tools, but you still need programmers to add own special functionality!
- Pure game engine. PixelLight is an universal framework which can also be used for 'serious' applications like product visualization, simulation or E-learning.
- One package - PixelLight consists of different components working together

1.4 General

- Completely programmed in C++ using modern C++11 language features like the null pointer literal *nullptr* or *override* enabling the compiler to detect and blame errors related to overwriting methods
- Whenever possible, well known design patterns are used
- Well structured and understandable code, due to strictly object oriented design
- Detailed documented code with explanations of parameters and return values, examples and notes
- Easily portable to other operation systems - currently Microsoft Windows, Linux, Mac OS X ³ and the mobile platforms Android⁴ and Maemo⁵ are supported
- 32 bit and 64 bit support
- Build system using CMake to make multi-platform development more comfortable⁶
- Clever Run-Time Type Information (RTTI) system which handles different classes with attributes, methods, signals, slots and properties. Classes are defined in modules where a module can be an executable or dynamic library. Therefore, the 'plugin-system' is naturally provided through the RTTI in an efficient manner.
- The framework comes with several development tools, plugins and libraries, as well as a lot of documentation and example programs
- The SDK itself is split up into several sub-projects to increase productivity. For instance there is a project with general classes like lists, hashtables etc. a mathematics library with vector, matrix etc.

³>=10.6

⁴>=2.3, Gingerbread, android-9 API level

⁵Experimental, used for example by *Nokia N900*

⁶*Microsoft Visual Studio 2010* project files are provided as well for comfortable development under *Microsoft Windows*

- Most PixelLight file formats are eXtensible Markup Language (XML) based⁷
- Many components are implemented with abstract interfaces and multiple default implementations. Own implementations can be added without any effort, making PixelLight extremely flexible and extensible.
- The headers are compact by using forward declarations where possible instead of hard includes. They are well documented and contain everything needed, and as such don't require to include many resources just to use one.

1.5 Debugging and Tweaking Tools

- PixelLight comes with a set of tools which shorten the development time. Using them makes it easier to find bugs and performance consumers in order to eliminate them!
- A console with a user friendly interface including auto-complete, history etc. It is also possible to register own new commands!
- Log system which is connected with the console and is able to print messages which could also be split up into debug mode dependent information
- Useful and customizable profiling tool which allows you to control different code parts during runtime in order to find out where your performance is burned! There is a lot of standard framework information like current Frames Per Second (FPS), triangle count, rendering time etc.
- For debugging purposes, PixelLight provides an extensive debug dialog were you are able to, for example, inspect and manipulate the different managers like the textures, meshes etc. There you can see whether your resources were loaded in the correct way and how often they are used by the certain resource managers. It is also possible to load new or replace/update existing resources as well as deleting some. For example, it is possible to open a texture, edit it and then reload it in order to see the changes at once in your project... it is not required to restart your program! This is an extremely useful feature of the technology for the whole development team, as it enables real-time editing and saves a lot of time.

⁷The primary chunk and mesh formats are binary based for efficiency, but they have XML counterparts as well for better data exchange

2 Base

2.1 PLCore

- Generic functor class¹
- Event system²
- RTTI with attributes, methods, signals, slots and properties
- Plugin system with support for delayed shared library loading to speed up the program start and to reduce memory consumption
- Loadable system. Everything that can be loaded and saved is not limited to just one 'hacked in' file format. Hence, it is possible to add, for instance, a loader plugin for your own mesh or image file formats in a quite universal way!
- Localization system
- Configuration system

2.1.1 Text Processing

- An advanced dynamic string class with American Standard Code for Information Interchange (ASCII), Unicode and UTF-8 support + extensive UTF-8 tool class
- An advanced tokenizer which allows you to parse texts without any effort
- XML classes (Document Object Model (DOM)) making it comfortable to load, edit and save XML files
- Regular expression class³
- Extensive command line parser

¹Delegate, a form of type-safe function pointer, 'callback'

²The principle is also known as signals & slots

³Internally PCRE (<http://www.pcre.org/>) is used

2.1.2 System

- Multi threading functionality (thread, semaphore, mutex)
- Dynamic/shared libraries
- Pipes
- Class for comfortable processes interaction
- Easy access to system information like used Operating System (OS), user name, available memory, etc.
- Access to environment variables
- Basic OS console functions
- Class for working with the registry (if the OS has one)

2.1.3 Network

- Basic universal standard network functionality like sockets
- Additional basic network functionality like client/server classes

2.1.4 File System

- File class for access to standard OS files, archives like zip⁴ and even http with password support and other special access settings
- Quite universal design. For instance, it is possible to open a file within an archive from an http server.
- Comfortable Uniform Resource Locator (URL) class taking the pain when dealing with filenames, especially when you want to work platform independent
- Access to the OS standard streams using the file class
- File search functionality with filters (wildcard, regular expression)

⁴Internally zlib (<http://www.zlib.org/>) is used

2.1.5 Script

- Generic script language independent script interface⁵
- Supported script features are global variables, global functions⁶, RTTI objects⁷ and namespaces
- Scripting is heavily using PLCore features like the RTTI, therefore adding script bindings or using RTTI objects within scripts is fairly straightforward and don't require the writing of thousands of proxy/wrapper classes exposing C++ functionality to script languages
- Certain parts of PixelLight are exposed to script languages through the loose plugin *PLScriptBindings*
- Scripting is completely optional, not mandatory

2.1.6 A huge set of Additional Tools

- Log with flexible implementation so we can output into a file (unformatted text, XML, Hypertext Markup Language (HTML)) or into the console
- Unified abstract checksum interface (the SDK comes with a MD5, CRC32 and SHA-1 plugin). Supports checksum from string, memory or files.
- General list interface with concrete implementations for linked list, array, bitset and so on
- Many basic tool classes like stack, queue, hash map, heap, quick sort, singleton, iterator etc.
- PixelLight comes with an efficient resource manager template system which is used by several managers, e.g. textures, meshes, sounds, paths, shaders etc.
- PixelLight provides an advanced timing class which offers a lot of timing relevant functions. Time difference since the last frame with clamp functionality, past time since start, slow motion, pause, freeze, FPS limiter, stopwatch
- Basic memory manager so you can mix release and debug builds as good as possible

⁵The PixelLight SDK comes with Null and Lua (<http://www.lua.org/>) backends, within the repository are also experimental JavaScript (using V8, a ECMA-262 compliant JavaScript engine, see <http://code.google.com/p/v8/>), Python (<http://www.python.org/>) and AngelScript (<http://www.angelcode.com/angelscript/>) backends.

⁶C++ calls script and script calls C++

⁷Properties, attributes, methods, signals, slots

2.2 PLDatabase

- Abstract unified database interfaces⁸

2.3 PLGraphics

- Extensive image class which is able to load and save the image formats dds, png⁹, tga, jpg¹⁰, ppm and bmp by default. Further formats can be added without any effort.
- Comfortable rgb and rgba color classes

2.4 PLMath

- Various comfortable and feature rich vector classes for 2D, 3D and 4D vectors
- Easily usable 3x3, 3x4 and 4x4 matrices with also offers important functions, like converting a direction vector into a rotation matrix
- Quaternions
- Planes
- Bounding box (AABB and OBB)
- Advanced Euler angles conversion tool class (from/to rotation matrix, from/to quaternion -> and all for multiple axis orders)
- Helper functions like transforming a 2D coordinate to an 3D and backward to, for example, find out were in the 3D world the 2D mouse cursor is in

2.5 PLGui

- PLGui is an universal library that offers many functions for creating editors, etc. The framework itself also uses this Graphical User Interface (GUI) for its own 'ingame' dialogs.
- The PLGui library provides simple classes for creating standard User Interface (UI) applications (using windows, dialogs etc.) and is designed to easily integrate PixelLight rendering into those programs. But you can also use other UI libraries and

⁸The PixelLight SDK comes with Null and SQLite (<http://sqlitebrowser.sourceforge.net/>) backends, within the repository are also experimental MySQL (<http://www.mysql.com/>) and PostgreSQL (<http://www.postgresql.org/>) backends.

⁹Using libpng <http://www.libpng.org/>

¹⁰Using libjpeg <http://www.ijg.org/>

integrate PixelLight rendering into them if you wish (e.g. Microsoft Foundation Class Library (MFC) or Qt)

- By using the PLGui for creating editors, they will also be compilable under every platform the framework is ported to (e.g. Linux)
- Tons of controls like bitmap, button, check box, combo box, edit box, group box, group box group, label, list box, list view, menu, menu bar, pull down button, progress bar, slider, spinner, splitter, status bar, tab bar, tool bar, tree view)
- Various window types like normal window, content window, frame, logo window, progress window, scroll window, tooltip, tool window
- Different dialogs like class view, color dialog, config view, file/directory dialog, file system view, input box, message box, system view

2.6 PLInput

- Aimed at modern input devices (6DOF tracking devices)
- The input component gives you a feature rich access to standard devices like keyboard, mouse and joystick/joypad
- Wii Remote support
- SpaceMouse (SpaceNavigator, SpacePilot and so on) support
- Device output controls such as rumble and force-feedback effects as well as control over device LEDs
- Internally HID, Bluetooth and special OS functionality is used

2.7 PLRenderer

- Dynamic API design¹¹
- Multiple output windows
- Render To Texture (RTT) (also rendering to floating-point for High Dynamic Range (HDR) buffer is possible)
- Multiple Render Targets (MRT) for rendering into different textures at the same time

¹¹The PixelLight SDK comes with Null and Open Graphics Library (OpenGL) (using FreeType (<http://www.freetype.org/>) for font support) backends, within the repository there's also a OpenGL ES 2.0 backend and an experimental Direct3D 9 backend

- Primitives are drawn through Vertex Buffer Object (VBO) and Index Buffer Object (IBO) for maximum performance
- Vertex streaming for combining the data of different VBOs
- 1D, 2D, 2D array¹², rectangle¹³, 3D and cube textures
- Mipmaps and texture compression support¹⁴. These can be created automatically on the fly or used from given dds data, for instance, for maximum control and best loading times.
- Abstract GPU program interface with support for vertex, geometry and fragment shaders
- The OpenGL renderer backend has a build in GPU program interface implementation for OpenGL Shading Language (GLSL)
- The OpenGL renderer backend has an optional GPU program interface implementation for Cg (http://developer.nvidia.com/object/cg_toolkit.html) via plugin
- Uniform buffer (UBO) support
- Fonts and draw helpers so it's easy to draw lines, images and so on
- Interface for fixed functions to support legacy graphics cards without, or just limited shader support
- A lot of standard functions like stencil, blend, fog, point sprites, anisotropic texture filtering and much more not worth to be mentioned in here because it's just standard must have stuff
- GPU program generator class for Über-shaders

2.7.1 Texturing

- Different texture creator plugins (for instance blank texture, normalization cube map, angle cube map...)
- The image class of PLGraphics is used to load and save textures which makes things quite comfortable
- Alpha blended textures which can be loaded through formats like tga automatically. It is also possible to define a color key by providing a RGB and tolerance value to describe transparent areas

¹²Kind of 3D texture, but without filtering between the depth layers

¹³Predecessor of the modern and universal Non-Power-Of-Two (NPOT) textures Graphics Processing Unit (GPU) feature

¹⁴DXT1 (= BC1), DXT3 (= BC2), DXT5 (= BC3), LATC1 (= 3DC+/ATI1N/BC4) and LATC2 (= 3DC/ATI2N/BC5)

- Maximum texture size is only limited by hardware, today nearly every card has at least textures with a size of 2048x2048, a GeForce4 for instance has maximum texture size of 4096x4096... and ATI Radeon HD 5870 up to 16384x16384...
- Textures are automatically resized by the framework if their size is too large for the given hardware or if its dimensions are invalid¹⁵. Moreover, there is a texture quality option in the configuration where the user can change the texture quality in order to gain more performance
- Procedural texturing
- Different texture animation techniques are provided, e.g. changing textures, texture matrix and color animations for a maximum freedom of creativity

2.7.2 Materials & Effects

- PixelLight comes with a powerful material & effect system which enables you to create amazing effects like Normal-Mapping etc.
- Each material consists of different flexible properties like color, shininess etc.
- In addition to the main effect properties, it is possible to setup a lot of options like blending, culling, polygon offset etc. for each material
- You can use as many texture layers as supported by hardware. Today there are at least 2 texture layers available, but a GeForce4 for instance has 4 and the latest hardware even has up to 16! Each texture layer can be animated which opens a huge animation freeness.
- The effect system supports different techniques (fallbacks), passes were you can assigned shaders to each pass etc.

2.8 PLMesh

2.8.1 Meshes

- Own flexible binary chunk based mesh format. The mesh library comes with different mesh import plugins for 3ds, lwo, ase, obj, smd, x (any many more) - through the nature of PixelLight it's no problem to implement more importer and exporter by self.
- Meshes can consist of different geometries and therefore it's possible to have a mesh with different materials per geometry.

¹⁵When the texture is for instance NPOT but the modern and universal NPOT textures GPU feature is not supported

- Mesh animations can be mixed together, therefore different animation channels for vertex and skeleton animations are provided per mesh
- Different mesh creator plugins (for instance sphere, cube, cylinder, disk, torus...)

2.8.2 Animation System

- All animation types are handled equally, meaning that the same animation interface is used to control vertex, skeleton, texture etc. animation. So every animation type can be accessed in the same way and with the same features!
- Animations can cause events at certain frames. Those events will be sent to their owner entity which can react on it
- Vertex (morph targets, can for example be used for facial animations) and skeleton animation system with multiple weights per vertex
- Blending of different animations

2.9 PLScene

2.9.1 Scene System

The dynamic and universal hierarchical scene system consists of scene nodes and scene containers (composite design pattern). A scene node is the most basic scene element, and a scene container manages such scene nodes. A scene container can also manage child scene containers for a hierarchical scene. Within a scene container, a scene hierarchy like a KD-tree can be used to manage/access the scene nodes in a more effective way. Scene queries operate on this scene description. For instance, there is a query which returns all scene nodes intersecting a line or plane set. To make the system more efficient, scene nodes can have different modifiers. For instance, you can use a physics scene node modifier to give your scene node a physical behaviour. Physics are not fixed build in the scene graph itself - it comes through such plugins to make the framework more universal.

- Powerfull 'scene node modifier' concept allowing to add as many modifiers to scene nodes you want taking for instance control over the position of the node, controlling morph targets of meshes and so on - the possibilities are nearly unlimited!
- Because the RTTI is used nearly everywhere in PixelLight, the scene system can be extended without any effort and components can be reused in other projects
- KD-tree and list scene hierarchy plugins provided
- Line/plane set intersection and culling queries provided as well as multiple other queries

- Scenes can be rendered using no culling at all, frustum culling or coherent hierarchical occlusion culling for more complex scenes
- A comfortable post process manager class and a lot of prepared post processing effects are provided. Adding bloom, grey scale or even visualizing the scene using ASCII characters is no problem at all. You can combine existing effects to create a new one.
- Various useful scene nodes like camera, light, sky, terrain, ingame GUI, mirror and so on are built in
- The sky scene node comes with multiple sky layers and creates impressive animated and atmospheric backgrounds like hills, where slowly moving clouds appear behind them
- The terrain scene node is using GeoMipMapping to create easily usable, impressive and large landscapes
- The scene format is a simple XML text format and therefore it's nearly version change save
- Lights can have coronas, lens flares or can even blind the screen looking into them! Lights can also project images over the scene, like a projector, which produces impressive effects! Because nearly all content in the framework is managed in the same way, you can also project a video texture over the scene where you will see a movie!

2.9.2 Compositing System

Scenes are rendered using a realtime compositing system. For example a first layer may clear the screen color to black, another may write down depth information and ambient color, a next may add lighting, another one fog and so on. The system is using the PixelLight RTTI, as such, it's expandable and can be heavily configured¹⁶.

- There are fixed functions and shaders based compositing layers to support a broad range of graphics cards
- Fixed function: For legacy hardware without shader support and just fixed build in graphics features
- Forward: A classic forward renderer using shaders. Each object is drawn per light again.
- Deferred: A modern deferred renderer approach performing for example lighting in image space

¹⁶Adding new layers, changing layer order, changing layer attributes, everything directly while a program is running

- Scene rendering is usually using Über-shaders to enable many shader features, while using just the currently required features
- Several types of light sources: directional, omnidirectional, spot, omnidirectional projective and projective spot
- Shadow mapping
- Post process system with 'build in' effects like Depth Of Field (DOF) and many effects as plugins
- High Dynamic Range Rendering (HDRR)
- Reinhard tone mapping, light adaptation, HDR bloom
- Glow
- Depth fog and volumetric fog
- God rays¹⁷
- Screen-Space Ambient Occlusion (SSAO) with implementations for Horizon Based Ambient Occlusion (HBAO) and High Definition Ambient Occlusion (HDAO)
- Fast Approximate Anti-Aliasing (FXAA)
- Gamma correction
- Layers with debug information like wireframe, scene node icons, scene node names and so on
- Many material features like diffuse and opacity mapping, normal mapping, detail normal mapping, parallax mapping, two sided lighting, specular and gloss mapping, Ambient Occlusion (AO) mapping, light mapping, emissive mapping, glow mapping, fresnel reflection, spherical environment mapping, cubic environment mapping, reflectivity mapping
- Normal map compression using swizzled DXT5 (xGxR), LATC2¹⁸ and alternate XY swizzle LATC2 is supported

¹⁷This effect is also known as crepuscular rays, sunbeams, sunbursts, star flare, sun shafts, or light shafts

¹⁸Formally *3DC/ATI2N* on ATI GPUs only

2.10 PLSound

- Abstract and universal sound API¹⁹
- 2D and 3D sound
- Streaming
- Global volume and pitch control (for instance to slow down the sound playback)

2.11 PLPhysics

- Abstract and universal physics API²⁰
- Ragdoll scene node (also called 'online animation', 'articulated character')
- Physics tool scene nodes
- Physics tool scene node modifiers: Normally you add physics to a scene node by just adding a physics scene node modifier to it

2.12 PLEngine

- High-level application classes
- Comfortable picking features
- Screenshot tool class

¹⁹The PixelLight SDK comes with Null and OpenAL (<http://www.openal.org/>) backends, within the repository are also experimental FMOD (<http://www.fmod.org/>) and FMOD Ex (<http://www.fmod.org/>) backends. *FMOD Sound System, copyright © Firelight Technologies Pty, Ltd., 1994-2011.*

²⁰The PixelLight SDK comes with Null and Newton Game Dynamics (<http://www.newtondynamics.com/>) backends, within the repository are also experimental Open Dynamics Engine (<http://www.ode.org/>) and PhysX (<http://developer.nvidia.com/object/physx.html>), backends.

3 Plugins

The PixelLight SDK comes with multiple optional plugins.

3.1 Frontends

Applications, including their lifecycle, are controlled by frontends. The SDK provides the choice between

- A null frontend without any GUI involved, useful for example when only rendering into background buffers
- A lightweight native OS frontend
- A frontend for PixelLight's own GUI system (uniform for OS & ingame) perfectly integrated into the complete system¹
- A frontend for Qt²
- Experimental frontends for Mozilla and ActiveX so you can run your PixelLight applications also for example within a browser window

PLFrontendQt

- Frontend and adapter project to bring PixelLight and Qt together, use PixelLight to e.g. render into one or multiple Qt widgets
- Static adapter class for mapping Qt strings to PixelLight strings and vice versa

libRocket_PL

- Integrates the free open source HTML/Cascading Style Sheets (CSS) game interface middleware *libRocket* (<http://librocket.com/>) into PixelLight

3.2 File Formats

PLAssimp 34 mesh and 34 scene loaders using Open Asset Import Library (ASSIMP) (<http://assimp.sourceforge.net/>)³

¹Stuff like scripting doesn't need wrappers, it just works out of the box thank's to the RTTI

²Qt is a cross-platform application and UI framework, <http://qt.nokia.com/>

³3ds, obj, Blender, Collada etc. for a full list, please have a look at assimp.sourceforge.net/main_features_formats.html

PLImageLoaderEXR

- EXR using OpenEXR (<http://www.openexr.com/>) and HDR image plugins

3.3 Particle Effects

SPARK_PL

- Integrates the free open source particle engine *SPARK* (<http://spark.developpez.com>) as a plugin into PixelLight
- Particle systems are usual scene nodes

PLParticleGroups

- Particle groups are usual scene nodes
- They emit and manage particles. Each particle can be customized in an easy way in order to create unique effects
- Each particle group can only have one material for all its particles for performance reasons. But because you are able to set the texture coordinates for each particle individually, you are able to put many different particle images into one texture and then cut them out to create fast and amazing particle effects
- The texture coordinate setting of the particles can be done automatically. You only have to define the rows and columns of the single sub-textures in the main material and then you are able to set the used sub texture by its index. With this technique it is also possible to create particles with animated textures!
- Because the material of the particle group itself can also be animated there are even more particle animation possibilities!
- Advanced particle effects like distorted particles (beams, lasers etc), rotation of the individual particles, particles with individual orientation etc. are possible
- Point sprite support for maximum performance
- Because the RTTI is used, no extra particle editor is required - just tweak the variables
- Different particle effects you can use without any effort. You are able to tweak them over several parameters.

3.4 Post Process Effects

- A collection of some useful⁴ and some quite pointless⁵ post process effects

3.5 Network

- PLJabber project implementing the Jabber protocol
- PLIRC project implementing the IRC protocol

⁴Edge detection, sharpen, bloom, blur, old film and so on

⁵Pointless crazy bars and so on

4 Tools

4.1 Autodesk 3ds Max Scene Exporter Plugin

- Plugin for Autodesk 3ds Max 2008, Autodesk 3ds Max 2009, Autodesk 3ds Max 2010, Autodesk 3ds Max 2011 and Autodesk 3ds Max 2012
- Exports meshes, materials and whole scenes without any detour into the PixelLight framework own format for an optimal workflow
- Mesh morpher modifier support
- Position, rotation and scale keyframe animation of nodes supported
- Skin and physics modifier support for skeleton animation with skinning
- Point cache support ('PC2' file format version 1)
- We provide an FX shader that can be used for an realtime material shader preview inside the Autodesk 3ds Max viewports

5 Contact

contact@pixellight.org

<http://www.pixellight.org>

Abbreviations

FAQ Frequently Asked Questions
API Application Programming Interface
SDK Software Development Kit, also known as *devkit*
DOM Document Object Model
XML eXtensible Markup Language
OS Operating System
GPU Graphics Processing Unit
GUI Graphical User Interface
UI User Interface
RTTI Run-Time Type Information, or Run-Time Type Identification
MFC Microsoft Foundation Class Library
HTML Hypertext Markup Language
URL Uniform Resource Locator
CSS Cascading Style Sheets
ASSIMP Open Asset Import Library
ASCII American Standard Code for Information Interchange
FPS Frames Per Second
OpenGL Open Graphics Library
HDR High Dynamic Range
HDRR High Dynamic Range Rendering
DOF Depth Of Field
SSAO Screen-Space Ambient Occlusion
HBAO Horizon Based Ambient Occlusion
HDAO High Definition Ambient Occlusion
FXAA Fast Approximate Anti-Aliasing
AO Ambient Occlusion
GLSL OpenGL Shading Language, also known as GLSLang
VBO Vertex Buffer Object
IBO Index Buffer Object
UBO Uniform buffer, also known as constant buffer
RTT Render To Texture
MRT Multiple Render Targets
NPOT Non-Power-Of-Two