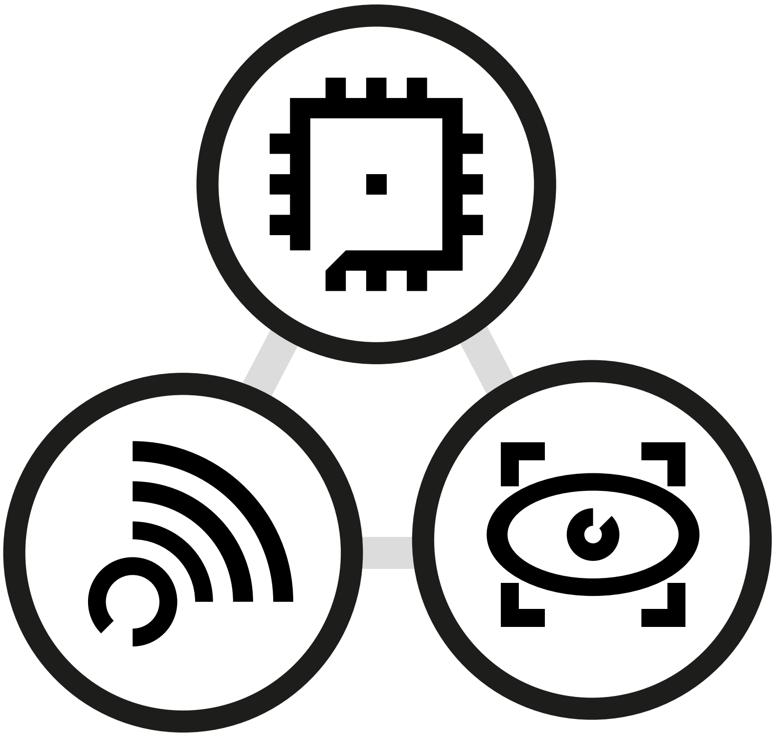
PFX Language Format

Revision: 1.0
11/02/2020
Public



Published: 11/02/2020-10:17

Contents

1. Introduction to PFX Version 2

What is PFX and what is it used for?

The purpose of this document is to act as a specification for the PowerVR Effects (PFX) format (PFX specification version 2.0.2).

What is PFX?

The PFX format is a small, simple, easy to use effects format consisting of several blocks that describe how a given graphics effect is put together (see Section *Blocks*).

As a minimum, a correctly formatted PFX consists of:

- One `EFFECT` block.
- One `VERTEXSHADER` block.
- One `FRAGMENTSHADER` block.

It is also possible for PFXs to contain the following:

- One `TARGET` block.
- Zero or more `TEXTURE` blocks.

By default, PFXs are stored in PFX files. It is possible for multiple PFXs to exist within a single PFX file, each described by a separate effect block. In this instance multiple PFXs may share blocks.

Finally, it is possible for a PFX to reference a `TARGET` block as an input as if it were a `TEXTURE` block, enabling the simple creation of complex post-processing effects. For this to function correctly the `TARGET` block render should be completed prior to being read as an input. If the `TARGET` block render has not been completed prior to being read as an input, the behaviour will vary based on the render target implementation of the platform.

2. Blocks

A PFX file is composed of several different types of blocks

This section outlines the structure of the various blocks which make up a PFX.

Each block contains a number of keywords which define the properties of that block. These keywords can be set to different valid values. This section will explain all of the available keywords and their potential associated values.

Blocks can also contain sub-blocks which contains extra information. These sub-blocks will be explained in detail when examining the relevant block.

An example of part of a PFX file is shown below:

```
// Comments can be used in PFX files as a handy way to describe the purpose of each of the blocks.
// This is a standard PFX header block.
// Click the text of any block to find out more.
[HEADER]
 VERSION  00.00.00.00
 DESCRIPTION TestPFXFile: Demonstrating what a PFX file looks like.
 COPYRIGHT Imagination Technologies Ltd.
[/HEADER]

// This is an effect block. It describes the entire effect with references to other types of
 blocks.
// Multiple effects can be embedded in a single PFX file.
[EFFECT]
 NAME exampleEffect

 ATTRIBUTE       inVertex POSITION
 ATTRIBUTE inNormal NORMAL

 UNIFORM MVPMatrix WORLDVIEWPROJECTION
 UNIFORM LightColor LIGHTCOLOR
 UNIFORM sLightMap TEXTURE0
 TEXTURE 0 cat
 TARGET COLOR<0> exampleTarget

 VERTEXSHADER SphereVertexShader
 FRAGMENTSHADER SphereFragmentShader
[/EFFECT]

// This is a texture block which describes a particular texture that is going to used in the
 effect.
[TEXTURE]
       // Keyword               // Value
 NAME   cat
 PATH   cat_texture.pvr
 MINIFICATION       LINEAR
 MAGNIFICATION      LINEAR
 MIPMAP       NEAREST
 WRAP_S       CLAMP
 WRAP_T       CLAMP
[/TEXTURE]

// This is a target block which specifies a surface to which an EFFECT block can render.
[TARGET]
       NAME              exampleTarget
       MINIFICATION      LINEAR
       MAGNIFICATION     LINEAR
 MIPMAP     NONE
 RESOLUTION       128 128
 WRAP_S     REPEAT
 WRAP_T     REPEAT
       SURFACETYPE       RGBA8888
[/TARGET]

// These are two simple shader blocks with GLSL code for a simple textured sphere.
[VERTEXSHADER]
 NAME   SphereVertexShader
 // LOAD GLSL AS CODE
 [GLSL_CODE]
  attribute highp   vec3  inVertex;
  attribute mediump vec3  inNormal;
  uniform highp   mat4  MVPMatrix;
  varying mediump vec3   ReflectDir;

  void main()
  {
   // Transform position
   gl_Position = MVPMatrix * vec4(inVertex, 1.0);

   // Pass the inverse of the normal (to map into the cubamap)
   ReflectDir = -inNormal;
  }
 [/GLSL_CODE]
[/VERTEXSHADER]

[FRAGMENTSHADER]
 NAME   SphereFragmentShader

 // LOAD GLSL AS CODE
 [GLSL_CODE]
  uniform samplerCube sLightMap;
  uniform lowp  vec3  LightColor;
  varying mediump vec3 ReflectDir;

  void main()
  {
   // Final colour is the modulaion of the base texture with the diffuse colour.
   gl_FragColor = (textureCube(sLightMap, ReflectDir)) * vec4(LightColor, 1.0);
  }
 [/GLSL_CODE]
[/FRAGMENTSHADER]
```

HEADER

The header block contains metadata for description and labelling

The header block contains metadata used for description and labelling purposes.

Table 1: Keywords

Keyword	Description
VERSION	The version of this PFX. The format follows this convention: MAJOR.MINOR.BUILD.REVISION
DESCRIPTION	A plain-text description of what this PFX file contains and the desired effect.
COPYRIGHT	Copyright descriptor of author(s).

Example code

```
[HEADER]
    VERSION         00.00.00.00
    DESCRIPTION     HeaderExample: The basic layout of a header block.
    COPYRIGHT       Imagination Technologies Ltd.
[/HEADER]
```

EFFECT

The EFFECT block is used to describe an individual effect by referencing TEXTURE, TARGET, VERTEXSHADER, and FRAGMENTSHADER blocks

The EFFECT block is the primary block used to describe a PFX. It references other blocks which can contain textures, targets, and shaders, as well as containing a number of application-specific *semantics*. These *semantics* can be used by an application to identify the meaning of a given attribute. The meaning of the available keywords and valid sub-blocks are described below.

The available semantics for attributes and uniforms are described in *PVRShaman PFX Semantics*.

Table 2: Keywords

Keyword	Description
NAME	A text identifier for this effect.
ATTRIBUTE	Specifies GLSL attribute variable. Format: ATTRIBUTE varName SEMANTIC • varName references a variable as specified in the shader blocks. • SEMANTIC references an application-specific semantic.
UNIFORM	Specifies GLSL uniform variable. Format: UNIFORM varName SEMANTIC • varName references a variable as specified in the shader blocks. • SEMANTIC references an application-specific semantic.
TEXTURE	Specifies a texture name which will be bound to the given unit. Format: TEXTURE UNIT TextureName • UNIT specifies an integer-based texture unit to bind to. • TextureName references a TEXTURE block of a given name.
TARGET	Specifies a target which this effect will write to, instead of the default frame buffer. Two types of targets exist, colour targets and depth targets. Only colour target support is required to be compliant with this specification. Format: TARGET BUFFERTYPE<UNIT> TargetName • BUFFERTYPE can be of type COLOR or, optionally, DEPTH. • UNIT is an integer based value defining the buffer unit. Only 0 is required to be compliant with this specification. • TargetName references a TARGET block of a given name.
VERTEXSHADER	References a VERTEXSHADER block of a given name.
FRAGMENTSHADER	References a FRAGMENTSHADER block of a given name.

Table 3: Sub-blocks

Block	Description
ANNOTATION	A block containing plain-text which will be copied as-is into a text buffer, readable by the application.

Example code

```
[EFFECT]
 NAME ExampleEffect

 ATTRIBUTE       inVertex POSITION
 ATTRIBUTE inNormal NORMAL
 ATTRIBUTE inTexCoord     UV

 UNIFORM MVPMatrix WORLDVIEWPROJECTION
 UNIFORM LightPosition LIGHTPOSWORLD
 UNIFORM LightColor LIGHTCOLOR

       TEXTURE 0 exampleTexture
       TARGET COLOR<0> exampleTarget

 VERTEXSHADER exampleVertexShader
 FRAGMENTSHADER exampleFragmentShader
[/EFFECT]
```

PVRShaman PFX Semantics

A list of the PFX semantics understood by PVRShaman

Semantics are keywords used to tell an application the meaning of the attributes or uniforms which can going to be referenced in the VERTEXSHADER and FRAGMENTSHADER.

PVRShaman is a shader composer tool created by Imagination Technologies. It used to view scene data from POD file format and edit shaders in the PFX format. This makes it easy to see the effect changing a shader has on a scene.

The tables below outline the semantic keywords in PFX files that PVRShaman understands.

Attributes

The table below lists all the attributes used by PVRShaman.

Keyword	Format description
POSITION	`vec4` Position.
NORMAL	`vec3` Normal.
TANGENT	`vec3` Tangent.
BINORMAL	`vec3` Binormal.
UV[n]	`vec2` n-th set of UVs. Example: UV0
VERTEXCOLOR	`vec4` Vertex colour attribute.
BONEINDEX	`vec4` Bone Index.
BONEWEIGHT	`vec4` Bone Weight.

Uniforms

The table below lists the uniforms used by PVRShaman.

Keyword	Format description
WORLD	`mat4` World matrix.
WORLDI	`mat4` World Inverse matrix.
WORLDIT	`mat3` World Inverse Transpose matrix.
VIEW	`mat4` View matrix.
VIEWI	`mat4` View Inverse matrix.
VIEWIT	`mat3` View Inverse Transpose matrix.
PROJECTION	`mat4` Projection matrix.
PROJECTIONI	`mat4` Projection Inverse matrix.
PROJECTIONIT	`mat3` Projection Inverse Transpose matrix.

Keyword	Format description
WORLDVIEW	`mat4` World-View matrix.
WORLDVIEWI	`mat4` World-View Inverse matrix.
WORLDVIEWIT	`mat3` World-View Inverse Transpose matrix.
WORLDVIEWPROJECTION	`mat4` World-View-Projection matrix.
WORLDVIEWPROJECTIONI	`mat4` World-View-Projection Inverse matrix.
WORLDVIEWPROJECTIONIT	`mat3` World-View-Projection Inverse Transpose matrix.
UNPACKMATRIX	`mat4` Matrix used to scale and offset vertex positions if the data has been exported with an unpack matrix.
VIEWPROJECTION	`mat4` View-Projection matrix.
VIEWPROJECTIONI	`mat4` View-Projection Inverse matrix.
VIEWPROJECTIONIT	`mat3` View-Projection Inverse Transpose matrix.
OBJECT	`mat4` Object matrix, without any parent node transformations.
OBJECTI	`mat4` Object Inverse matrix, without any parent node transformations.
OBJECTIT	`mat3` Object Inverse Transpose matrix, without any parent node transformations.
MATERIALOPACITY	`float` Opacity of material.
MATERIALSHININESS	`float` Shininess of material.
MATERIALCOLORAMBIENT	`vec3` Ambient colour of material.
MATERIALCOLORDIFFUSE	`vec3` Diffuse colour of material
MATERIALCOLORSPECULAR	`vec3` Specular colour of material.
BONECOUNT	`int` Number of bones.
BONEMATRIXARRAY	`mat4[]` Array of bone transformation matrices.
BONEMATRIXARRAYIT	`mat4[]` Array of bone inverse transpose transformation matrices.
LIGHTCOLOR[n]	`vec3` Colour of light n (RGB). Example LIGHTCOLOR5.
LIGHTPOSMODEL[n]	`vec3` Position of light n in model space. Example LIGHTPOSMODEL1.
LIGHTPOSWORLD[n]	`vec3` Position of light n in world space. Example LIGHTPOSWORLD1.
LIGHTPOSEYE[n]	`vec3` Position of light n in view space. Example LIGHTPOSEYE1.
LIGHTDIRMODEL[n]	`vec3` Direction of light n in model space. Example LIGHTDIRMODEL1.
LIGHTDIRWORLD[n]	`vec3` Direction of light n in world space. Example LIGHTDIRWORLD1.

Keyword	Format description
LIGHTDIREYE[n]	`vec3` Direction of light n in view space. Example LIGHTDIREYE1.
LIGHTATTENUATION[n]	`vec3` Attenuation for spot lights (constant, linear, quadratic).
LIGHTFALLOFF[n]	`vec2` Falloff for spot lights (angle, exponent).
EYEPOSMODEL	`vec3` Eye position in model space.
EYEPOSWORLD	`vec3` Eye position in world space.
TEXTURE[n]	`sampler2D` Sampler for texture n. Example TEXTURE2.
ANIMATION	`float` Contains the objects distance through its animation. Range 0 to 1.
GEOMETRYCOUNTER	`int` Resets to 0 at the beginning of each render frame and increases by one for each submission of geometry.
VIEWPORTPIXELSIZE	`vec2` Size of the viewport in pixels.
VIEWPORTCLIPPING	`vec4` Near distance, far distance, width angle (radians), height angle (radians).
TIME	`float` The current time in seconds.
TIMECOS	`float` Cosine of the current time in seconds.
TIMESIN	`float` Sine of the current time in seconds.
TIMETAN	`float` Sine of the current time in seconds.
TIME2PI	`float` Tangent of the current time in seconds.
TIME2PICOS	`float` Cosine of the current time in seconds wrapped to 2п.
TIME2PISIN	`float` Sine of the current time in seconds wrapped to 2п.
TIME2PITAN	`float` Tangent of the current time in seconds wrapped to 2п
LASTTIME	`float` The last frame’s time.
ELAPSEDTIME	`float` The time between adjacent frames.
BOUNDINGCENTER	`vec3` Bounding box centre.
BOUNDINGSPHERERADIUS	`float` Bounding sphere radius.
BOUNDINGBOXSIZE	`vec3` Bounding box size.
BOUNDINGBOXMIN	`vec3` Bounding minimum for x, y, z.
BOUNDINGBOXMAX	`vec3` Bounding maximum for x, y, z.
RANDOM	`float` A random value (Range 0 to 1).
MOUSEPOSITION	`vec3` The mouse position on screen (x, y, time).
LEFTMOUSEDOWN	`vec4` The left mouse down state, and its position at that time (x, y, isdown, timedown).
RIGHTMOUSEDOWN	`vec4` The right mouse down state, and its position at that time (x, y, isdown, timedown).

TEXTURE

A TEXTURE block describes a surface which is based on texture data

A TEXTURE block describes a surface that can either be the contents of a texture file or the contents of a framebuffer/render texture.

The valid keywords and values for this block are outlined in the table below.

Table 4: Keywords

Keyword	Description
NAME	A text identifier for this texture.
PATH	The filename of the texture. If spaces are included, enclose the path in quotation marks, for example, "base map.pvr".
MINIFICATION	Minification texture filter flags. Valid values: • `NEAREST` • `LINEAR`
MAGNIFICATION	Magnification texture filter flags. Valid values: • `NEAREST` • `LINEAR`
MIPMAP	MIP-map texture filter flags. Valid values: • `NEAREST` • `LINEAR` • `NONE`
VIEW	Specifying the VIEW keyword modifies the functionality of the defined texture to be a render texture of the current scene. Valid values: • `PFX_CURRENTVIEW` • POD camera name in optional quotation mark, e.g., "Camera01".
CAMERA	An alias for VIEW.
RESOLUTION	Describes the resolution of the texture. This will be ignored if PATH is specified, where the resolution will be that of the loaded texture.
WRAP_*x*	Where *x* is a valid axis (S, T, or R). Specifies the texture wrapping in the defined axis. Valid values: • `REPEAT` • `CLAMP`
SURFACETYPE	Describes the surface/pixel type of the texture. This will be ignored if PATH is specified, where the surface type will be that of the loaded texture. Valid values: • `RGBA8888` • `RGBA4444` • `RGB888` • `RGB565`
FILTER	**[Deprecated]** Allows the specification of texture filter flags in short-hand. Valid values are hyphen separated filter flags in the order of minification, magnification, and MIP-map, e.g., `LINEAR-LINEAR-LINEAR` enables trilinear texture filtering, or `LINEAR-LINEAR-NONE` enables bilinear.

Keyword	Description
WRAP	**[Deprecated]** Allows the specification of texture wrapping flags in short-hand. Valid values are hyphen separated wrapping flags in the order of axis S, T, and R. For example `REPEAT-CLAMP-CLAMP`.

Table 5: Values (texture block)

Value	Associated Keyword	Description
PFX_CURRENTVIEW	VIEW CAMERA	Indicates that the render texture should be derived from the current view of the scene and not from a specific POD file camera.
LINEAR	MINIFICATION MAGNIFICATION MIP-MAP	Linearly interpolates between sampled texels.
NEAREST	MINIFICATION MAGNIFICATION MIP-MAP	Chooses nearest texel based on Manhattan distance.
NONE	MIP-MAP	Disable MIP-mapping.
CLAMP	WRAP_S WRAP_T WRAP_R	Clamps to texture border.
REPEAT	WRAP_S WRAP_T WRAP_R	Repeats at texture border.
RGBA8888	SURFACETYPE	32bit RGBA texture format.
RGBA4444	SURFACETYPE	16bit RGBA texture format.
RGB888	SURFACETYPE	24bit RGB texture format.
RGB565	SURFACETYPE	16bit RGB texture format.
INTENSITY8	SURFACETYPE	8bit intensity texture format.

Example code

```
[TEXTURE]
       // Keyword               // Value
 NAME   cat
 PATH   cat_texture.pvr
 MINIFICATION       LINEAR
 MAGNIFICATION      LINEAR
 MIPMAP       NEAREST
 WRAP_S       CLAMP
 WRAP_T       CLAMP
[/TEXTURE]
```

TEXTURES [Deprecated]

The TEXTURES block can be used to describe all of the textures which are associated with a particular effect

The TEXTURES block has been replaced with multiple TEXTURE blocks, which allows individual textures to be specified more verbosely.

This block is therefore deprecated and will be removed in a future version of the specification. The use of this block is now discouraged.

Each line of the TEXTURES block takes the form identified below:

```
FILE TextureName FileName.pvr FILTERFLAGS WRAPFLAGS
```

The table below outlines what each of the values in TEXTURES block means.

Table 6: Values (textures block)

Value	Description
TextureName	Specifies a text identifier.
FileName.pvr	Specifies the filename of the texture.
FILTERFLAGS	Specifies a hyphen separated list of texture filter flags (see *TEXTURE*).
WRAPFLAGS	Specifies a hyphen separated list of texture wrapping flags (see *TEXTURE*).

TARGET

A TARGET block describes a surface to which an EFFECT block can render

A TARGET block describes a surface to which an EFFECT block can render. A PFX may read from a TARGET block as if it were a TEXTURE block as long as the render to that block has been completed prior to the read.

The valid keywords and values are outlined below.

Table 7: Keywords (target block)

Keyword	Description
NAME	A text identifier for this target.
MINIFICATION	Minification texture filter flags. Valid values: • `NEAREST` • `LINEAR`
MAGNIFICATION	Magnification texture filter flags. Valid values: • `NEAREST` • `LINEAR`
MIPMAP	MIP-map texture filter flags. Valid values: • `NEAREST` • `LINEAR` • `NONE`
RESOLUTION	Describes the resolution of the texture. This will be ignored if PATH is specified, where the resultant resolution will be that of the loaded texture.
WRAP_*x*	Where *x* is a valid axis (S, T, or R). Specifies the texture wrapping in the defined axis. Valid values: • `REPEAT` • `CLAMP`
SURFACETYPE	Describes the surface/pixel type of the texture. This will be ignored if PATH is specified, where the resultant surface type will be that of the loaded texture. Valid values: • `RGBA8888` • `RGBA4444` • `RGB888` • `RGB565`

Table 8: Values (target block)

Value	Associated Keyword	Description
LINEAR	MINIFICATION MAGNIFICATION MIP-MAP	Linear interpolates between sampled texels.
NEAREST	MINIFICATION MAGNIFICATION MIP-MAP	Chooses nearest texel based on Manhattan distance.
NONE	MIP-MAP	Disable MIP-mapping.
CLAMP	WRAP_S WRAP_T WRAP_R	Clamps to texture border.
REPEAT	WRAP_S WRAP_T WRAP_R	Repeats at texture border.
RGBA88c88	SURFACETYPE	32bit RGBA texture format.
RGBA4444	SURFACETYPE	16bit RGBA texture format.
RGB888	SURFACETYPE	24bit RGB texture format.
RGB565	SURFACETYPE	16bit RGB texture format.
INTENSITY8	SURFACETYPE	8bit intensity texture format.

Example code

```
[TARGET]
    NAME             exampleTarget
    MINIFICATION     LINEAR
    MAGNIFICATION    LINEAR
    MIPMAP     NONE
    RESOLUTION       128 128
    WRAP_S     REPEAT
    WRAP_T     REPEAT
    SURFACETYPE      RGBA8888
[/TARGET]
```

VERTEXSHADER and FRAGMENTSHADER

Shader blocks are used to hold the GLSL code for the shaders used in an effect

VERTEXSHADER and FRAGMENTSHADER blocks contain the GLSL code for an effect. The code can either be embedded within the PFX file itself or be located elsewhere and referenced with a filename using the FILE keyword.

Table 9: Keywords

Keyword	Description
NAME	A unique identifier for this shader.
FILE	A file name of a text file containing valid GLSL code.

Table 10: Sub-blocks

Block	Description
GLSL_CODE	A block containing plain-text GLSL code.

Example code

```
[FRAGMENTSHADER]
 NAME SphereFragmentShader

 // LOAD GLSL AS CODE
 [GLSL_CODE]
  uniform samplerCube sLightMap;
  uniform lowp  vec3  LightColor;
  varying mediump vec3 ReflectDir;

  void main()
  {
   // Final colour is the modulaion of the base texture with the diffuse colour.
   gl_FragColor = (textureCube(sLightMap, ReflectDir)) * vec4(LightColor, 1.0);
  }
 [/GLSL_CODE]
[/FRAGMENTSHADER]
```

3. Contact Details

For further support, visit our forum:

http://forum.imgtec.com

Or file a ticket in our support system:

https://pvrsupport.imgtec.com

For general enquiries, please visit our website:

http://imgtec.com/corporate/contactus.asp