# PVRVecEx

# User Manual



| | | |
|---|---|---|
| Filename | : | PVRVecEx.User Manual.mht |
| Version | : | 1.1f (PowerVR SDK 2.03.23.1153) |
| Issue Date | : | 10 Jun 2008 |
| Author | : | PowerVR |

# Contents

# 1. Installation and Usage

PVRVecEx is a plug-in for Adobe Illustrator that allows exporting any vector graphics artwork in a OpenVG friendly format.

This plug-in is intended for versions CS, CS2 and CS3 of Adobe Illustrator.

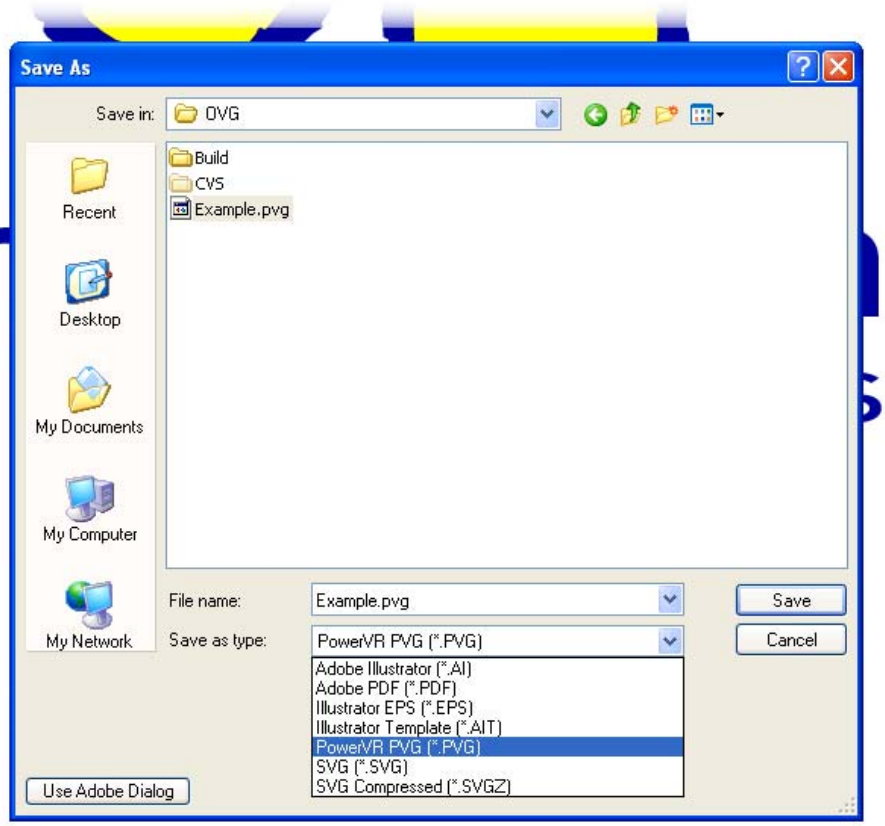
The same plug-in will work in versions CS2 and CS3

Copy the file supplied for your version in this folder:

*\Program Files\Adobe\Adobe Illustrator <version>\Plug-ins*

Now there is a new supported format when saving or loading a file. This format is called PVG and it is described below.

**Save as…**



# 2. PVG Format description

PVG is a binary format that stores sequentially the artwork data using OpenVG conventions. This makes a very compact format (half the size than a SVG file) and very easy to load in an OpenVG application.

All the code needed to load this format and an example on how to load and display a PVG file is supplied with the PowerVR OpenVG SDK.

## 2.1. Format Definition

Except the file header, any block (Paints, Path, Ramps, Dashes or Patterns) can be repeated any number of times, but they are grouped together following the other below.

**HEADER (Unique):**

- DWORD dwMagicToken;
  *dwMagicToken has to be POVG*
- DWORD dwSizeOfHeader;
  *Total size of this header.*
- DWORD dwVersion;
  *dwVersion containing Mayor(short) and Minor(short), 0xMMMMmmmm*
- DWORD dwBuild;
  *dwBuild containing Branch(short) and Build(short), 0xBBBBbbbb*
  *This build number is needed when reporting any bug related to a PVG file*
- DWORD dwFileFlags;
  *Currently unused*
- DWORD dwFileSize
  *Total file size counting this header*
- DWORD dwNumPaths;
- DWORD dwNumPaints
- DWORD dwNumRamps;
- DWORD dwNumDashes;
- DWORD dwNumPatterns
  *Number of objects to be found in this file*
- FLOAT fLeft, fTop, fRight, fBottom;
  *Drawing extensions*

**RAMPS (As many as dwNumRamps):**

**RAMP HEADER:**

- DWORD dwNumRampValues
  *Total number of Colour Ramp entries.*

**RAMP DATA:**

*Ramp values will be written after dwNumRampValues as (STOP, R, G, B). the total size of the RAMPS data block will be 4*dwNumRampValues*sizeof(float)*

**PAINTS (As many as dwNumPaints. Paints do not have data block):**

**PAINT HEADER:**

- DWORD dwPaint

*Unique identifier of this paint to be used by the Path definition*

- DWORD dwPaintStyle

*VG_PAINT_TYPE_COLOR, VG_PAINT_TYPE_LINEAR_GRADIENT, VG_PAINT_TYPE_RADIAL_GRADIENT or VG_PAINT_TYPE_PATTERN*

- FLOAT fPaintParameters[5]

*VG_PAINT_TYPE_COLOR:*

*R, G, B, Alpha, Fith parameter ignored.*

*VG_PAINT_TYPE_LINEAR_GRADIENT:*

*Origin.x, Origin.y, Target.x, Target.y, Alpha*

*VG_PAINT_TYPE_RADIAL_GRADIENT:*

*Origin.x, Origin.y, Focal.x, Focal.y, Radius*

*VG_PAINT_TYPE_PATTERN:*

*Currently not implemented.*

- DWORD dwRamp

*Identifies the Colour Ramp used for gradient. Ignored if not gradient.*

- DWORD dwPattern;

*Identifies the Pattern used for paint. Ignored if not pattern.*

**PATHS (As many as dwNumPaths):**

**PATH HEADER:**

- DWORD dwPathType;

*VG_FILL_PATH and/or VG_STROKE_PATH*

- DWORD dwClipping;

*Clipping path = 1 otherwise = 0*

- DWORD dwAlphaBlend;

*0 if no alpha, otherwise 1-255*

*opacity: fully translucent (1) to fully opaque (255)*

- DWORD StrokePaint;

*StrokePaint identifier in file (Paint list)*

- DWORD FillPaint;

*FillPaint identifier in file (Paint list)*

- DWORD StrokeDash;

*0 if none, otherwise Dash definition in file (Dash list)*

- DWORD dwFillRule;

*VG_EVEN_ODD or VG_NON_ZERO*

- DWORD dwStrokeJoin;

*VG_JOIN_MITER, VG_JOIN_ROUND or VG_JOIN_BEVEL*

- DWORD dwStrokeCap;

*VG_CAP_BUTT, VG_CAP_ROUND or VG_CAP_SQUARE*

- FLOAT fStrokeWidth;

*Same width value passed to OpenVG*

- FLOAT fStrokeMiterLimit;

*Same miter limit value passed to OpenVG*

- DWORD dwNumSegments;

*Total number of segments in this path*

- DWORD dwPathDataSize;

*Total Path Data size in Bytes.*

**PATH DATA:**
*Path Values will be write after this header in OpenVG format. First a list with all the commands* (there are dwNumSegments commands) f*ollowed by the list of all control points (size is variable depending on the commands).*

**DASHES (As many as dwNumDashes):**

**DASH HEADER:**

- DWORD dwNumDashComponents;

  *Number of dash elements (form 1 to 12)*

- DWORD dwLength;

  *Length of the dash segment.*

- FLOAT fOffset;

  *Displacement for the starting point.*

**DASH DATA:**

*Dash values to be written after this header as floats.*

**PATTERNS (As many as dwNumPatterns):**

**PATTERN HEADER:**

- DWORD dwWidth;

  *Total width of the rectangular pattern;*

- DWORD dwHeight;

  *Total height of the rectangular pattern*

**PATTERN DATA:**

*Pattern texture values to be written after this header in 16-bit 565RGB format*

# 3. PVG Functions

## 3.1. PVG.h

This file is the main interface of PVG format with the OpenVG application. It is distributed with the PowerVR OpenVG SDKs.

```
typedef struct _PVG_FILE_HEADER
{
	unsigned int	dwMagicToken;
	unsigned int	dwSizeOfHeader;
	unsigned int	dwVersion;
	unsigned int	dwBuild;
	unsigned int	dwFileFlags;
	unsigned int	dwNumPaths;
	unsigned int	dwNumPaints;
	unsigned int	dwNumRamps;
	unsigned int	dwNumDashes;
	unsigned int	dwNumPatterns;
	float		fLeft, fTop, fRight, fBottom;
} PVG_FILE_HEADER;


typedef struct _PVG_PATH_HEADER
{
	unsigned int	dwPathType;
	unsigned int	dwClipping;
	unsigned int	dwAlphaBlend;
	unsigned int	StrokePaint;
	unsigned int	FillPaint;
	unsigned int	StrokeDash;
	unsigned int	dwFillRule;
	unsigned int	dwStrokeJoin;
	unsigned int	dwStrokeCap;
	float		fStrokeWidth;
	float		fStrokeMiterLimit;
	unsigned int	dwNumSegments;
	unsigned int	dwPathDataSize;
} PVG_PATH_HEADER;


typedef struct _PVG_PAINT_HEADER
{
	unsigned int	dwPaint;
	unsigned int	dwPaintStyle;
	float		fPaintParameters[5];
	unsigned int	dwRamp;
	unsigned int	dwPattern;
} PVG_PAINT_HEADER;


typedef struct _PVG_COLOR_RAMP_HEADER
{
	unsigned int	dwNumRampValues;
} PVG_COLOR_RAMP_HEADER;


typedef struct _PVG_DASH_HEADER
{
	unsigned int	dwNumDashes;
	float		fPhase;
	float		fDashValues[6];
} PVG_DASH_HEADER;


typedef struct _PVG_PATTERN_HEADER
{
	unsigned int	dwWidth;
	unsigned int	dwHeight;
} PVG_PATTERN_HEADER;
```

## 3.2. PVRTLoadPVG Functions

PVRTLoadPVG is the source code to load and display any PVG file. This source, as pvg.h, can be found in the Tools section of the PowerVR OpenVG SDKs.

### 3.2.1. FromFile

```
static COvgObject* FromFile(const char* pszFilepath);
```

This function loads a PVG file and initialises all the pathswithing OpenVG. A COvgObject class containing the loaded data and the procedures for diplaying it will be created.

### 3.2.2. FromMemoryBuffer

```
static COvgObject* FromMemoryBuffer(unsigned char* pui8Buffer, int
i32Size);
```

Same as FromFile function but this time from a memory block. The whole file with full header is required in this function. As the previous function a COvgObject will be created.

### 3.2.3. COvgObject ::Draw

```
bool Draw(int i32StartPath = 0, int i32EndPath = -1);
```

This function draws the full art or a range of paths within the PVG file.

### 3.2.4. COvgObject ::SetAlpha

```
void SetAlpha(unsigned char ui8Alpha);
```

Set global alpha for the art.

### 3.2.5. COvgObject ::SetupCenterOnOrigin

```
void SetupCenterOnOrigin();
```

Cetre the full collection of paths to be easily displayed.

### 3.2.6. COvgObject ::SetupTranslateToOrigin

```
void SetupTranslateToOrigin();
```

Translate the full collection of path to the origin to be easily displayed.

### 3.2.7. COvgObject ::SetupScaleToSize

```
void SetupScaleToSize(float fTargetWidth, float fTargetHeight, bool
bKeepAspect = true);
```

This function and the two previous ones are used to prepare a PVG file to be displayed properly (centred and scaled) within the screen limits.