PowerVR

OpenVG Application Development

Recommendations

Filename	:	PowerVR.OpenVG Application Development Recommendations.mht
Version	:	1.2f (PowerVR SDK 2.03.23.1196)
Issue Date	:	18 Aug 2008
Author	:	PowerVR

Contents

List of Figures

1. Introduction

This document provides an initial introduction to developing high quality and high performance OpenVG applications for PowerVR platforms.

The document is split up into four sections:

- **Offline Recommendations**
  Offers recommendations for when you are designing your application.
- **Pre-Processing Recommendations**
  Contains recommendations on what you can do before run-time to improve performance.
- **Run-time Recommendations**
  What to avoid in order to maintain a steady frame-rate.
- **Reference Material & Contact Details**
  Provides various online resources and contact details.

1.1. Golden Rules

The following are a set of generic guidelines which should be followed to get the best performance out of your application:

- Avoid very large paths.
- Cache the triangulation of your paths by drawing them on or before the first frame.
- Avoid modifying paints, masks or paths regularly.
- Use batching to reduce API calls.
- Do not use vgReadPixels() in a real application.
- Do not change anti-aliasing mode mid-frame.
- Release all the Path, Paint and Image data if your application is done with it.

When designing your application it is important to take into consideration that the OpenVG implementation on PowerVR & SGX hardware uses the chip's 3D capabilities for displaying paths. This means that for paths to be displayed, any Bezier curves are first tessellated into multiple line segments. The piecewise linear path is then triangulated into a set of (non-overlapping) triangles. Depending on various properties of the path this process can be costly. This is why most of the recommendations in this document are based on getting the best performance out of your application with regard to the tessellation of paths.

2. Offline Recommendations

This section offers recommendations for when you are designing the paths for your application. In particular the recommendations here are for use at the start of your project when you are designing the vector graphic artwork for your application with the view to maximise performance.

2.1. Good Path Design

As the target for your application is likely to be a mobile device, the resources available to you will be limited. The complexity and size of your paths have a direct bearing on the amount of memory your application requires. There are many parts in the drivers which deal with a single path at a time and, as large paths require large commitments of platform resources, they may reduce performance. For example, the triangulation of large or complex paths can have an adverse affect on the memory requirements of your application. Therefore, it is recommended, where possible, to split very large paths into simpler sub-paths.

It is important to note that, assuming you do not run out of memory and the paths and the applied transform go unchanged, the size of a path or the complexity of the shapes will only affect performance the first time they are drawn. After that point the resultant triangulation is cached in memory so avoids re-tessellation and triangulation.

3. Pre-Processing Recommendations

This section details things you can do at the initialisation stage of your application to improve performance when it is running. Though these steps will add time onto your initial setup phase, they will help to achieve smoother performance at run-time, where a drop in performance would be more noticeable to the user.

3.1. Pre-loading

It is recommended that, where possible, you pre-load items at the start of your application. This way you can be sure of being allocated the minimum necessary resources with the added benefit of avoiding noticeable pauses in your application at a later stage. Table 1 lists items that are recommended for pre-loading as they perform pre-calculations and cache intermediate data. Once the pre-calculation is performed, the items listed will perform optimally at run-time as long as they are not modified.

Recommended for Pre-loading
Paints
Paths (see Section 3.2 for how to do this and why)
Colour Ramps
Masks
Images

Table 1 – Recommended for Pre-loading

3.2. Cache Paths

As already mentioned, to display a path the drivers need to draw them as a set of triangles. If a suitable triangulation of the path is not available when the application attempts to draw the path, the drivers will generate one and cache it for future use. As triangulation is a potentially costly procedure, it is best to make the drivers do this at the start of your application instead of mid-animation. To cache the triangulation of a shape you only need to draw it on the first frame. If your shape is not required in the first frame then you can cache it by drawing the shape before the clear, that way it will get triangulated and cached but will not be displayed. A cautionary note is also in order here: Please use this technique wisely, all devices do have limited memory, so fair judgment will be required.

When a path is scaled up, there may be a point where more line segments are required to represent the curved shape accurately, and this forces a re-tessellation and re-triangulation. PowerVR drivers always cache the most detailed version of the shape and scales that down to the smaller sizes without performing re-triangulation. Therefore, if you plan to scale up/down a path which contains curved segments, then it is best, from a caching perspective, to render the path at its maximum required scale. This render may be just a superficial one and may not even be used for your first few frames, but it will definitely deliver a better runtime experience.

More information on what can cause re-triangulation is available in Section 4.2. Once again, this only affects paths with curved segments as scaling does not affect paths composed of straight lines.

The pseudo code below demonstrates how to cache paths in a situation where they are getting up-scaled across several frames but they are not at their largest scale on the first frame.

```
Frame 0

Scale Paths to largest size needed
Draw Paths
Clear Screen
Draw Paths at required size for the frame

Frame n

Clear Screen
Scale Paths to required size
Draw Paths
```

In the case of Figure 1 where a Tiger's head is getting up-scaled across 6 frames at Frame 0 the largest head would be drawn before the clear and then the smallest one would be drawn afterwards. On successive frames nothing special needs to be done; the paths are just resized as normal.

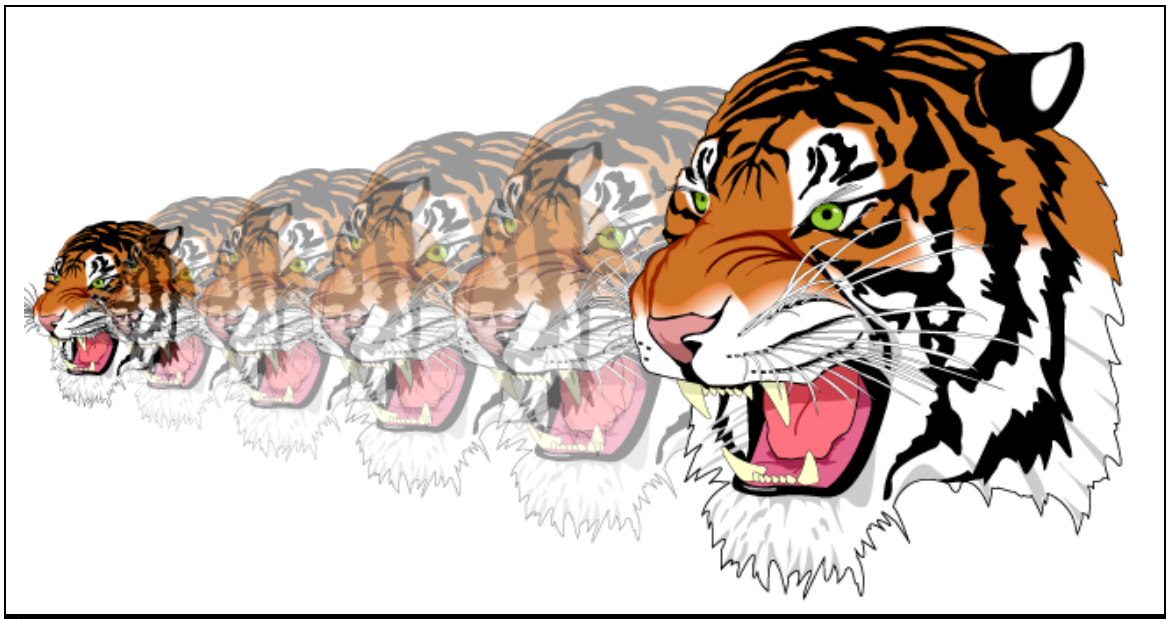

Figure 1 Tiger up-scaling

4. Run-time Recommendations

The following recommendations are designed to get the best performance out of your application whilst also offering recommendations on how to keep this performance consistent.

4.1. Batching

Each API call introduces an amount of overhead. To reduce the CPU load, the number of API calls should be minimised. This can be achieved through the concept of batching – grouping operations together, as much as possible, into larger batches. Batching can be done on multiple levels, for example based on image (e.g. render all objects with the same image in one go) – consider using an image atlas (one large image containing multiple sub images) to achieve this; paint (e.g. render all objects with the same colour); etc. Batching also reduces internal state changes.

4.2. Paths

As mentioned previously when a path is drawn it first needs to be triangulated which is an expensive process therefore it is best to try and limit the amount of re-triangulation required. Whenever an application modifies a path's segment coordinates or properties, it triggers re-tessellation and re-triangulation. As an example of avoiding re-triangulation if you are animating a character with a repeating animation (e.g. a walking animation) it would be better to put each frame into its own path (memory permitting) and allow the drivers to cache each frame instead of modifying the segment coordinates of the path every frame. If you cause re-triangulation on a regular basis the time and resources taken to triangulate will be smaller on convex shapes so where possible try and split concave shapes up into convex shapes. Another design guideline may be to break up the geometry into static and dynamic shapes which need to be redone often.

4.2.1. Batch Appends

If you are appending segment data to a path, it is recommended to do it all at once instead of doing each segment one at a time.

4.3. Paints and Masking

Where possible, try not to change paints, colour ramps or masks often as this causes cached intermediate data to be regenerated, which is again expensive. It is important to note that the radial gradient is tailored for the path so modifying path data will also result in the radial gradient being recalculated.

4.4. Images

When using images, you will get better performance if the image uses an identical format to the frame buffer, as no software based conversion between formats is required. The same is also true if you are copying data from one image to another. Setting the format of your frame buffer to 565 will offer optimal performance.

If an image is created in a format that is not directly supported by the hardware (Note: MBX & SGX pixel data is supplied and rendered in ARGB format), then image pixel data must be processed into a hardware compatible format when the image is first drawn. Changing such images causes the image data to be re-processed. Users should minimize the modifications to such an image during a scene as much as possible.

For best performance make the dimensions of your images powers of two and if the image is opaque then use of a texture format with no alpha channel (XRGB, RGB, L) will help the driver optimise drawing.

The following image-associated items are expensive procedures so they should be used sparingly during run-time;

- Image filters
- Tile-filled mode
- Stencil images

4.5. Blending

When performing blending it is best to use pre-multiplied colours as blending is always done with pre-multiplied colours regardless of the supplied format. Passing in images that are in pre-multiplied format should reduce (though this is also dependent on other things) conversion overhead of converting to pre-multiplied format. Avoid the VG_BLEND_DARKEN and VG_BLEND_LIGHTEN blend modes as they are expensive.

4.6. Anti-Aliasing

It is recommended to not change the anti-aliasing mode mid-scene as this causes the hardware to perform additional renders.

4.7. Do not use vgReadPixels() in a real application

The use of vgReadPixels() is very expensive as it causes a rendered frame to be flushed and, as such, should only be used for non-performance related apps like testing and frame capture etc. However, if you do need to use it then it is best to use the same data format as the frame buffer to prevent any software conversion.

4.8. Power Consumption

While a dedicated hardware core such as PowerVR uses considerably less power than running a software engine, the storage capacity of the batteries in embedded devices is still a limited resource. On a high performance hardware 3D renderer it is possible to create more frames per second than are really required (e.g. above LCD refresh rate or above a rate required for the application). To avoid wasting performance under these conditions, it is possible to limit the hardware core peak frame rate and hence reduce power consumption. In OpenVG this can be done using the eglSwapInterval() function. For example setting a swap interval of two on a 60Hz display ensures the frame-rate cannot go above 30Hz.

5. Reference Material & Contact Details

PowerVR Public SDKs can be found on the PowerVR Developer Website:

http://www.pvrdev.com/Pub//Download/default.htm

Additional OpenVG Programming information can be found on the Khronos Website:

http://www.khronos.org/

Further Performance Recommendations can be found in the Khronos Developer University Library:

http://www.khronos.org/devu/library/

Developer Community Forums are available:

http://www.khronos.org/message_boards/

Additional information and Technical Support is available from PowerVR Technical Support who can be reached on the following email address:

devrel@powervr.com