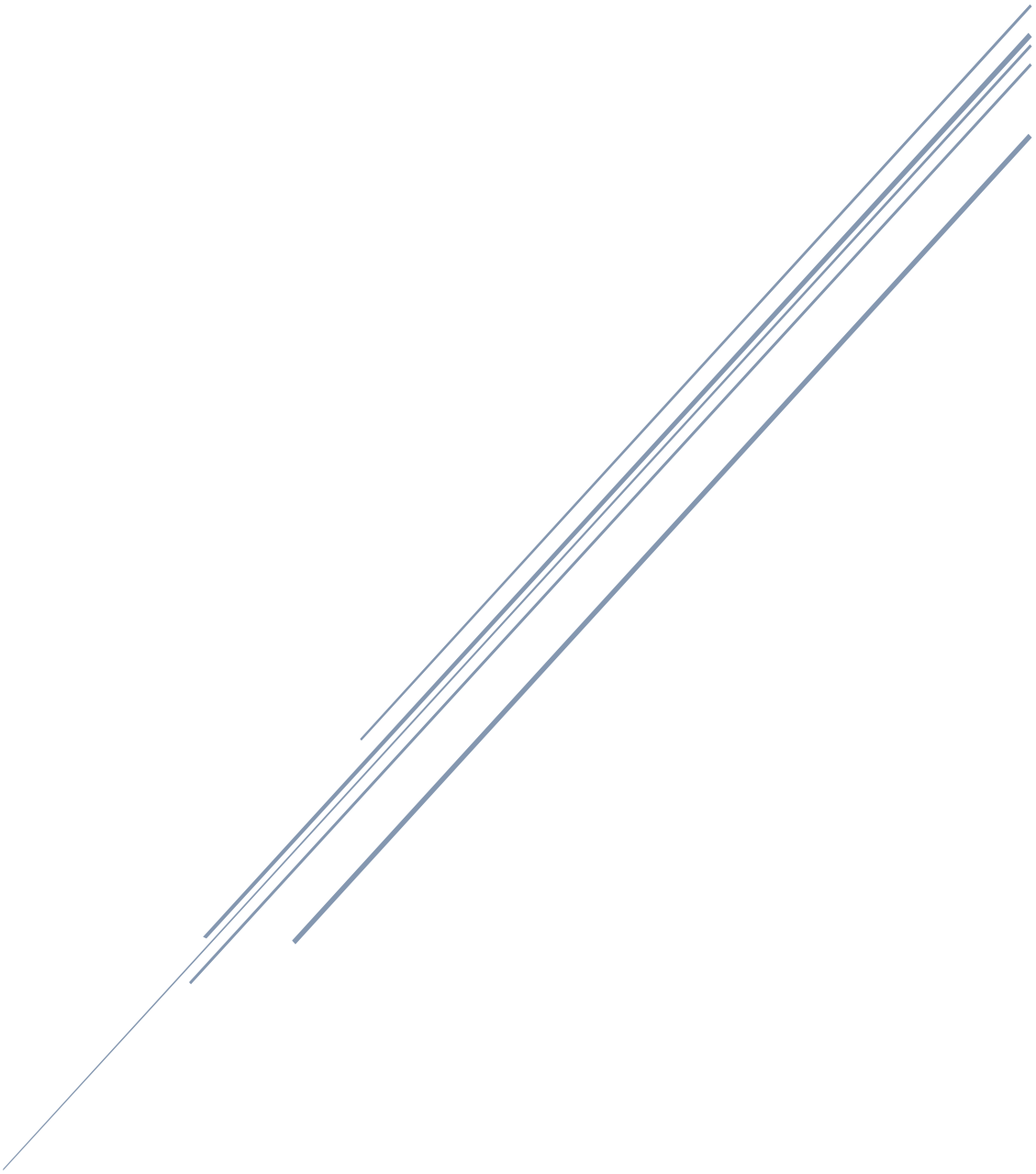
Orbiter 2D Graphics



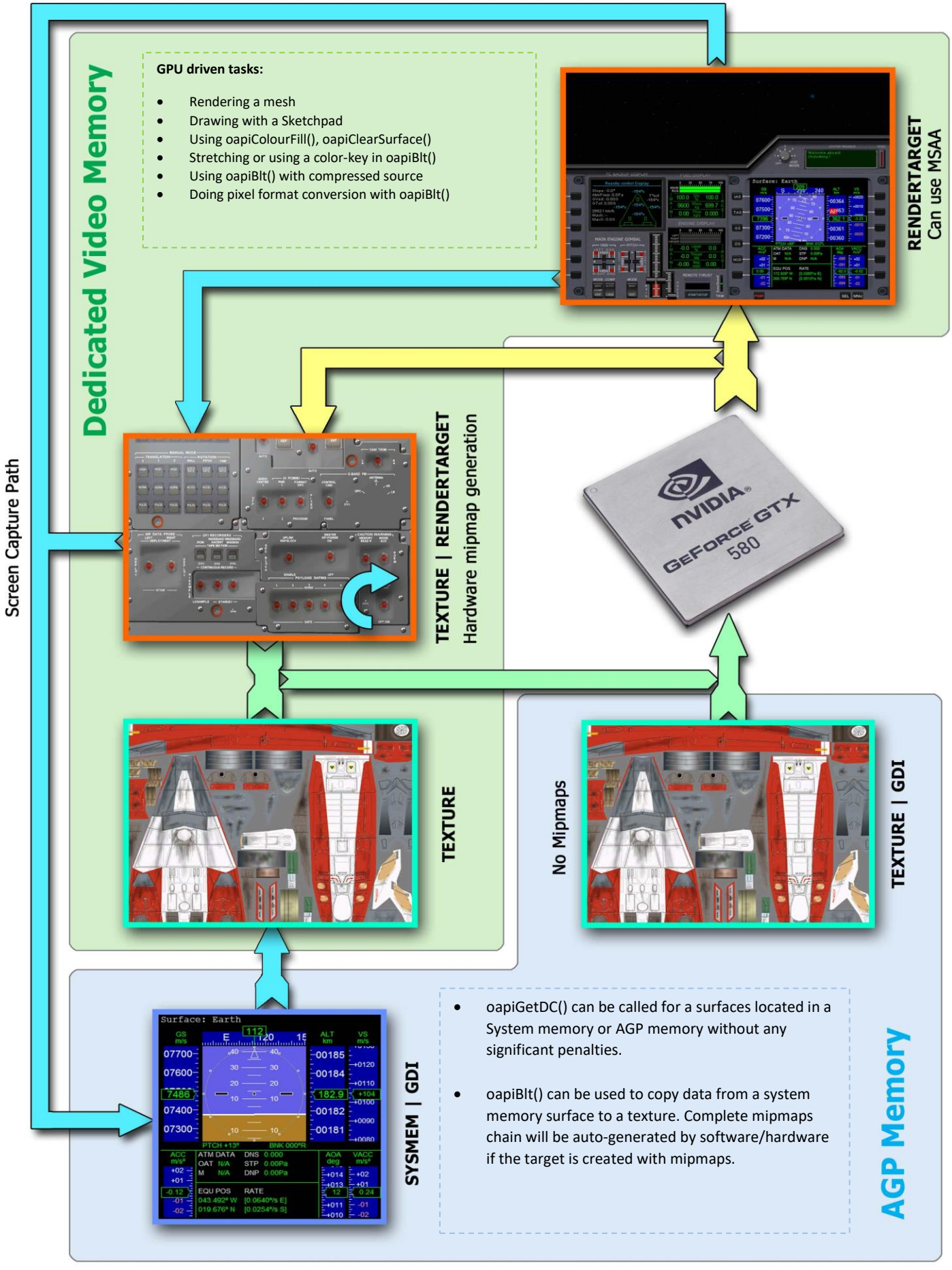


Figure 1- 2D Graphics Pipeline Schematics

The Schematics

When looking at the graphics flow chart from a previous page you can focus your attention to the GPU and how it is positioned relative to the five different surfaces in the schematics. The GPU is the primary execution unit that creates almost all of the graphics we see in the Orbiter. Rendering *Meshes*, drawing with the *Sketchpad* and executing commands like *oapiBlt, oapiClearSurface* and *oapiColourFill* functions are all done with the GPU. The main flow of graphics in the schematics is from the bottom to the top.

The GPU can only draw/render graphics in a surface created as **RENDERTARGET**. The yellow pathway in the schematics represents that path (*i.e. Output from the GPU*). Also, any surface that is being read by the GPU must be a **TEXTURE**. This is shown as a green pathway in the schematics (*i.e. Input to the GPU*). When *oapiBlt* is used to copy graphics along green and yellow pathways, *stretching* and *color-keying* are supported. However, for such operations a use of *Sketchpad* is recommended especially if there is a higher quantity of copy operations per target.

In addition to the GPU there also exists a resource management unit that can transfer data between various graphics resources, however the transfer is limited to a simple plain copy operation only. Surface formats must be identical in source and destination. Color operations such as *alpha-blending*, *color-keying* or *stretching* are **not** supported. These options are presented via the blue pathway in the schematics and the copy operations can be done by using *oapiBlt.* The "*screen capture pathway*" has a special requirements those are discussed later on.

Orbiter API calls *oapiCreateSurfaceEx* and *oapiLoadSurfaceEx* can be used for creating a surfaces or loading graphics directly into a specific type of a surface.

Surface Types and Descriptions

RENDERTARGET | TEXTURE

The **RENDERTARGET | TEXTURE** is the most versatile surface there is and likely the most common user allocated surface in the Orbiter. It can be used for both reading and writing by the GPU. If combined with **MIPMAPS** flag a full chain of mipmaps are created. Also the mipmaps are automatically regenerated by hardware when being used as a texture after the main/top surface level has been modified by any operation. So, one can assign this kind of surface directly into a mesh and draw onto it.

The small curved arrow in a lower-right corner of the surface indicates that the surface can be used for in-surface blitting (*i.e. the same surface can be a source and a target at the same time and the rects can overlap*), but that is limited to a simple copy rect operation with *oapiBlt*. That is not a native DX feature and it is actually done with two copy operations through a temporary surface. The old *oapiCreateSurface* and *oapiCreateTextureSurface* will create a **RENDERTARGET | TEXTURE** and the later will also add **MIPMAPS** flag and the format is XRGB in both cases.

RENDERTARGET

The plain vanilla **RENDERTARGET** is a single layer surface (e.g. no mipmap support). The only specialty in this surface is MSAA (Multi-Sample-Anti-Alias) which can be enabled by adding **ANTIALIAS** flag to a surface creation. The level of anti-aliasing will depend on Launchpad configurations. 3D rendering ability can be enabled for any **RENDERTARGET** by adding **RENDER3D** flag. This will also work for **RENDERTARGET | TEXTURE.**

TEXTURE

The old good **TEXTURE** is most common surface and it's mainly used a by meshes and are automatically loaded by the Orbiter. If **MIPMAPS** flag is specified during loading or surface creation, a full chain of mipmaps are created even if the file doesn't have any. Also **NOMIPMAPS** flag will tell the texture loader to reject any mipmaps the file might have. If neither flag is defined then the outcome is defined by the file being loaded.

TEXTURE | GDI

This is a dynamic surface located in AGP memory and can be updated by the CPU. Also can be used for drawing by GDI in other words *oapiGetSurfaceDC* works with this surface. There are no mipmap support in this surface type. If the surface data is being accessed by CPU its write only access. Always uncompressed format XRGB.

SYSMEM | GDI

This is a graphics surface located in a system memory or in an AGP memory depending on implementation. *oapiGetSurfaceDC* also works with this surface and the surface data can be accessed by CPU for reading and writing. No mipmap support. If the entire surface (*not a sub-rect*) is copied to a **TEXTURE|MIPMAPS** surface created with *oapiCreateSurfaceEx* then a full chain of mipmaps are created in destination. Although, the level of hardware support and therefore speed may wary depending on implementation.

Texture Decompression

There are ways to request a texture decompression and in some cases it will happen automatically. As an example **RENDERTARGET** can't be compressed, therefore, if a texture is being loaded directly into such a surface then a decompression is automatically applied. By default the format is XRGB but it can be user controlled by applying **ALPHA** flag or any of the **OAPISURFACE_PF_*** pixel format flags. Decompression of a regular texture can be enabled by passing **DECOMPRES** flag or defining a pixel format **OAPISURFACE_PF_***.

Addition of '**D**' charter after the texture name in a *mesh file* will trigger decompression into XRGB format during loading.

If *oapiBlt* is used in a compressed target surface, which can easily happen when a texture is loaded by a mesh, automatic decompression is triggered. DXT1 will decompress in XRGB format, DXT3 and DXT5 types will decompress in ARGB format.

Screen Capture

In a screen capture pathway source and target rectangles are not supported. So, the whole surface is being transferred at once. Both surfaces must be the same size and the pixel format must be the same as well. Also, since the pipeline is asynchronous a typical delay of two frames can exists between making an API call to copy the data and when the copy is actually completed. Real-time screen recording applications should poll the system memory surface (once per frame) if the copy is completed or not. *oapiGetSurfaceDC* will return 'NULL' if the copy is pending.

Shared Resources

Any resource loaded with *oapiLoadSurfaceEx* can be made shared or global by adding **SHARED** flag, after that any call to *oapiLoadSurfaceEx* with the same file name and **SHARED** flag will return a pointer to previously allocated resource. If the flag isn't provided then a local copy is made. Shared resources can't be released/deleted during runtime they are automatically deleted during exit.

oapiClearSurface() and oapiColourFill()

These functions run with the GPU when used in any **RENDERTARGET** surface. If used in a **TEXTURE|GDI** or **SYSMEM|GDI** surface the functions will reroute the call to GDI by a use of *GetDC*, these functions will fail if used in a plain **TEXTURE** surface.

SketchPad interface

SketchPad is a GPU driven 2D drawing interface which can be created with *oapiGetSketchpad* for any surface combined with **RENDERTARGET** flag. The **OAPISURFACE_SKETCHPAD** flag is replaced with **RENDERTARGET** flag if used. See Orbiter API documentation or *DrawAPI.h* for more information about Sketchpad.

(To be continued….)