RenderWare Graphics

White Paper

Updating Your RenderWare Graphics Assets

Contact Us

Criterion Software Ltd.

For general information about RenderWare Graphics e-mail info@csl.com.

Developer Relations

For information regarding Support please email devrels@csl.com.

Sales

For sales information contact: rw-sales@csl.com

Acknowledgements

With thanks to RenderWare Graphics development and documentation teams.

1. RenderWare Graphics Versioning: What, Why and How

RenderWare Graphics maintains a version number that is split into two major components, the SDK version and the binary stream version. These version numbers are returned in a concatenated state by the function `RwEngineGetCurrentVersion`. An example of the version number being returned might be 34005, which can be broken down into SDK version 3.40 and a binary stream version of 05. The SDK version is only incremented when a new release of the RenderWare Graphics SDK is shipped. The binary stream version may, however, increment in between major releases to allow bugs to be fixed whilst maintaining backwards compatibility.

The version numbering is stored inside RenderWare Graphics streams when they are written to indicate which version of the SDK was used to generate the stream. This versioning is crucial to allow RenderWare Graphics to maintain the ability to read streams written out with previous versions of the SDK. This information within RenderWare Graphics streams is used to determine the data formats stored in the stream and the code paths required to successfully read the data from the stream.

RenderWare Graphics streams are a chunk-based format and, as a result, the version numbers are held within the chunk headers and are used to describe the version number format of the data maintained within that chunk. As a result old unsupported data (or new unsupported data) will not cause an application to crash. Instead RenderWare Graphics performs a number of checks during the process of finding a chunk with `RwStreamFindChunk` or reading chunks with `RwStreamReadChunkHeaderInfo`.

Based on the version number found during the chunk reading processes, RenderWare Graphics will follow one of three main code paths. The code paths are chosen based on a set of version numbers stored within the SDK being used. These are as follows:

1. `BaseVersion` – This is the lowest version number from which the current SDK can still read data.

2. `WarnVersion` – Any versions prior to, and including this version, are calling non-optimal code and will be removed in future versions. This version will often be the same as the `BaseVersion`.

3. `CurrentVersion` – The current version of the SDK. Any data streamed out will be written using the formats of this version number and will have this version number stored in their chunk headers.

For any chunks found with a version less than the `BaseVersion` or greater than the `CurrentVersion`, RenderWare Graphics will refuse to read the chunk and fail in the current operation. This is to protect the user against an attempt to read a chunk of data in an unknown format that may cause the application to crash.

Any chunks found with a version number greater than or equal to the `BaseVersion` and less than or equal to the `WarnVersion`, RenderWare Graphics will successfully load the chunk of data performing any necessary conversion of the data during the streaming process. This process can therefore be quite a bit slower than reading the data if it were stored in the current versions format. RenderWare Graphics will also send out a warning in a debug build of the application to indicate that it has found a chunk considered to be in a legacy format. This warning indicates that we strongly advise you to update the assets containing this data to the current RenderWare Graphics version to ensure more optimal streaming performance and the ability to upgrade to future versions of RenderWare Graphics without needing to re-export the data from scratch.

Any chunks found with versions greater than `WarnVersion` and less than or equal to the `CurrentVersion` will be streamed in without warning. We still however recommend that updating the assets to the latest version of RenderWare Graphics may improve streaming performance. However an upgrade to a future version of RenderWare Graphics would be possible with this data without the need to re-export.

The stepping of the various versions is set up to ensure we always support reading assets from the `CurrentVersion` and previous version without warning. Hence, when upgrading to the latest version you can do so without receiving constant warnings about potentially out of date assets. We will also usually support at least one more version prior to that, with warnings that those assets are non-optimal and may become non-streamable in the following release. At a release, the `CurrentVersion` will usually be incremented up by a single release unit, and the `BaseVersion` and `WarnVersion` will remain equal and step up a release unit. Hence we will normally support a `BaseVersion` with warnings and two versions beyond that without warnings.

As mentioned before there is also a binary stream component in the version number that is separated from the major version number. It should be noted here that increments only in the binary version number **do not** constitute a new version.

2. Why Update Assets

Although one of the primary reasons for the versioning data held in RenderWare Graphics streams is for backwards compatibility streaming, this does not mean that it is advisable to use assets streamed out via older versions of RenderWare Graphics. Many of the reasons for incrementing the stream version number are to incorporate more optimal ways to describe or store data required for an asset. When a more optimal solution is incorporated into the RenderWare Graphics SDK, a version increment will happen and we will guarantee to support reading of the old stream data for a period of time (as outlined in *1 RenderWare Graphics Versioning: What, Why and How*).

Although we give this guarantee of supporting older assets for a period of time the use of such assets may give non-optimal performance in your application. Although, in general, the rendering process will still be optimal, the streaming process will need to perform a conversion of the data at runtime as the asset is loaded. This conversion process can potentially be a slow process and as a result would give an application poor performance of asset loading. This may present itself as slow level loading in a game.

By taking the time to update your assets to the current RenderWare Graphics version you are ensuring that you final application will follow the most optimal code path for asset streaming and will also have stored the assets in their most optimal format on your storage media. This will result in the best performance in your application.

Updating your assets also gives the added protection that should you wish to upgrade to a future major RenderWare Graphics release, this will also guarantee the ability to convert without any re-exporting of assets. This will enable the fastest possible upgrade path for your game. However, again, we strongly recommend that following the upgrade process you should then go about upgrading all your assets to that new version of the RenderWare Graphics SDK.

3. How to update to the latest version

Updating assets in RenderWare Graphics is a relatively straightforward process once working practices are set up to cope with it. There are two real ways to update artwork to the latest stream format in RenderWare Graphics. The first is applicable only to developers making use of the `rf3` format and is detailed further in *4 The Way Forward - rf3*. Because `rf3` files are RenderWare Graphics version agnostic, the process is to simply reconvert the `rf3` files with the current RenderWare Graphics SDK.

The second process will allow you to convert assets between RenderWare Graphics versions simply by streaming them into an application and then streaming them back out again. Because RenderWare Graphics always holds assets in the most optimal form, it can memory stream in an old format asset, which will perform conversion during the streaming process. Once held in the optimal form, memory streaming back out to disk will result in a copy of the asset in the most optimal streamed format on disk with a version number matching the version in the SDK used for the conversion. The conversion process will therefore be performing the non-optimal streaming, as part of an offline process that will ensure the streaming in with a final runtime application is as optimal as it can be.

The process of updating assets through streaming into and out of an updated application is as straightforward as opening two streams and, for the example of an `RpClump`, calling `RpClumpStreamRead` followed by `RpClumpStreamWrite`. The only complication is introduced by any plugin data stored within the objects you are trying to convert. In order to ensure plugin data is correctly maintained during the update process, all plugins used by the object should be attached in the application, this includes both RenderWare Graphics plugins and any third party plugins or custom plugins that a developer may have written.

An example of a function to update an `RpClump` might be,

```
RwBool
UpdateClump(RwChar *oldClumpName, RwChar *newClumpName)
{
        RwStream *stream;
        RpClump *clump = NULL;

        stream = RwStreamOpen( rwSTREAMFILENAME,
                               rwSTREAMREAD,
                               oldClumpName );
        if (stream)
        {
              if (RwStreamFindChunk(stream, rwID_CLUMP))
              {
                    clump = RpClumpStreamRead(stream);
              }
              RwStreamClose(stream, NULL);
        }
        if (clump)
        {
              stream = RwStreamOpen( rwSTREAMFILENAME,
                                     rwSTREAMWRITE,
                                     newClumpName );
              if (stream)
              {
                    RpClumpStreamWrite(clump, stream);
                    RwStreamClose(stream, NULL);
              }
              RpClumpDestroy(clump);
              return(TRUE);
        }
        return(FALSE);
}
```

This process can be followed for any types of RenderWare Graphics assets to move them from any version above the current libraries `BaseVersion` up to the `CurrentVersion`.

If you are required to move assets from a version less than the `BaseVersion` of the current library then you will first need to convert them using an older version of RenderWare Graphics. Any older version with a `BaseVersion` less than the asset version and a `CurrentVersion` greater than the current libraries `BaseVersion` will suffice. In extreme circumstances it might be necessary to have more than two version steps to upgrade. However we don't envisage this happening throughout the life cycle of a single application. By keeping assets up to date you can also avoid any of these problems should you wish to share assets across multiple development cycles.

4. The Way Forward - rf3

RenderWare Graphics 3.5 has introduced and new file format for use in the art tool chain. This new format can also be seen as the way forward in maintaining version compatibility.

The `rf3` files store all the data required to generate RenderWare Graphics assets in a fully extendable format (XML). By storing the data in a format that is RenderWare Graphics version agnostic it will be possible to regenerate all of your RenderWare Graphics assets targeted to any version of RenderWare Graphics. This regeneration would ensure that all the assets are stored in an optimal format for both streaming into an application and rendering at run time.

One of the major benefits of using the `rf3` files is that you can export your data from modeling packages and then perform the conversion into RenderWare Graphics assets as part of the build process for you application or offline on machines without needing modeling packages to be installed. This allows developers the ability to keep assets permanently in the most up to date optimal format, removing the need to ever worry about upgrading RenderWare Graphics versions and needing to re-export/convert any current assets.

The process of converting `rf3` files into RenderWare Graphics assets can be done either through provided command line tools or by building your own conversion tools using RenderWare Graphics supplied libraries. These topics are covered throughout the RenderWare Graphics Technical Artist Guide and the rf3cc tool user guide.