FLEXPORTER FAQ

Last update : February, 25, 2001

Why do you use funky comments in your code ?

Because I use DOXYGEN to automatically create my docs….
Please note there's no online Flexporter doc anyway ! The DOXYGEN doc is the ICE doc, and that one is way too big (and way too incomplete) to be of any use to Flexporter users.

Why is it so slow to export a Character Studio skin ?

Because Character Studio provides a Vertex Interface, that I must use. For each vertex I have to get it, use it, release it, and it happens to be slow. Here is the code I use, if someone knows a faster way, I would be happy to hear about it :

```
IPhyVertexExport* vtxExport = pcexport->GetVertexInterface(i);
if(vtxExport)
{
	// Need to check if vertex has blending
	if(vtxExport->GetVertexType() & BLENDED_TYPE)
	{
		// Here, each vertex is linked to multiple bones.
		IPhyBlendedRigidVertex* vtxBlend =
			(IPhyBlendedRigidVertex*)vtxExport;

		// Number of driving bones
		udword NbBones = vtxBlend->GetNumberNodes();

		for(udword n=0;n<NbBones;n++)
		{
			INode* Bone	= vtxBlend->GetNode(n);
			Point3 Offset	= vtxBlend->GetOffsetVector(n);
			float Weight	= vtxBlend->GetWeight(n);
		}
		pcexport->ReleaseVertexInterface(vtxExport);
		vtxExport=null;
	}
	else
	{
		….other code path…
	}
}
```

How do I use the Character Studio's offset vectors, weights, etc ?

Here’s some code samples.
For non-blended vertices, this is very easy :

```
// Compute mesh deformation for Character Studio's non-blended vertices
for(udword i=0;i<NbVerts;i++)
{
	// Each vertex is linked to a single bone, whose ID has been exported
	udword ID = BonesID[i];

	// We also have one offset vector per vertex. The offset vector
	// must be multiplied by the skeleton's bone's matrix. Note : the
	// absolute matrix, not the relative one.
	Point P = Offsets[i] * Matrix[ID];

	// The vertex might need further transformation to match D3D frame !
	Skin[i].p.x = P.x;
	Skin[i].p.y = P.z;	// Note y and z are swapped here !
	Skin[i].p.z = P.y;
}
```

For blended vertices, this is almost as easy :

```
// Compute mesh deformation for Character Studio's blended rigid vertices
for(udword i=0;i<NbVerts;i++)
{
	// Each vertex is drived by N bones
	udword NbBones = BonesCounts[i];

	Point Blended(0, 0, 0);
	float TotalWeight = 0.0f;
	for(udword j=0;j<NbBones;j++)
	{
		// Get the current bone’s ID for that vertex
		udword ID = BonesID[j];

		// Get current weight for that bone
		float Weight = BonesWeights[j];

		// Update total weight
		TotalWeight+=Weight;

		// The skin's offset must be multiplied by the
		// skeleton's bone's matrix…
		Point P = Offsets[j] * Matrix[ID];

		// …then weighted…
		P *= Weight;

		// Take bone’s contribution into account
		Blended += P;
	}

	// D3D transform + final division
	Skin[i].p.x = Blended.x / TotalWeight;
	Skin[i].p.y = Blended.z / TotalWeight;
	Skin[i].p.z = Blended.y / TotalWeight;
}
```

NB : the code for blended vertices can be rewritten in a more efficient way (you can get rid of the final division by modifying the weights once and for all). The code here just shows how to use Character Studio data as you get them from MAX.

Gosh ! It crashed MAX !

Actually, it shouldn't (…) But I experienced some really strange crashes indeed. Sometimes it helps to manually collapse all your geometries before exporting.

Why did you choose a Utility plug-in ?

Because it allows me to add extra stuff in Flexporter if I want to. For example if I suddenly need to burry a lightmapper as a Flexporter option, I can. It would be difficult to do the same with an Export plug-in….

Where should I put all the DLLs ? Where's Flexporter in MAX ?

First of all, be sure you read the Installation notes in Flexporter.doc.

Then :

- You must have MAX 3.x or MAX 4, put the main DLU file in MAX's Plugins directory, and the other DLL files in a reachable place, for example in the MAX root directory with the other DLLs. Flexporter plug-ins (.FLX files) must be in MAX root directory as well.
- Run MAX.
- To check the plug-in has correctly been loaded, look in: File->Summary Info...
- Then click on Plug-In info... and look for Ice Exporter.
- To access the Plug-In, you must click on the Utilities property page, the last one on the right, in the Command Panel. If there's no "ICE Flexporter" in the list of utility plug-ins, click on "Configure Button Sets" and add an entry for it.
- That's the standard procedure to access Utility Plug-Ins (.DLU files).

Is there a ZCB reader / importer available ?

There is one in Panard Vision (http://pvision.planet-d.net/) but I didn't code it and I don't know how well it works, and even if it's up to date.

At first ZCB was just a little private test format, to check all the features were correctly exported. I wasn't expecting everyone would use that. That's the whole point of Flexporter : you're supposed to be able to code your own format in a very easy way, without even needing the MAX SDK.

I think recoding a ZCB importer is a lot more complicated than writing a new format plug-in for Flexporter.

Now, if you badly need it, there's some code in the Flexporter SDK to load a ZCB file. It has not been written on distribution purpose, and then you might have some minor modifications to do in order to even compile it. Nonetheless this is the code I'm currently using, and it does the job.

New :
Check out the little ZCB reader here : www.codercorner.com/ZCBReader.htm

I can't recompile the exporters, help !

- I only tested with VC++ 6.0, not the previous versions
- MAX plugs (and Flexporter plugs) should be compiled in Release mode only.

To debug Release builds, go to the Settings panel, then :

- In the C/C++ sheet, select *Optimizations :Disable (Debug)*, *generate browse info*, and *Debug info : Program Database*.
- In the Link sheet, select *Generate debug info*.

How do I use the cropping values ?

Here's some code to handle the texture matrix and the cropping values :

```
Point TextureVector;
TextureVector.x = ExportedU;
TextureVector.y = ExportedV;
TextureVector.z = ExportedW;  // Or 0.0f
Point CoordMaps = TextureVector * TexMat; // Multiply by the texture matrix
float u = CoordMaps.x;
float v = CoordMaps.y;
FinalU = (u*ScaleU)+OffsetU;        // Use the cropping values
FinalV = ((v-1.0f)*ScaleV)-OffsetV;
```

I know the equations look weird, but I'm not responsible for that….

See ZCBHelp.cpp in the Flexporter SDK for actual source code.

Character Studio export doesn't work with my scene !

I forgot to expose some rather severe limitations indeed. My mistake.

1) Only use real BIPED parts. Do not use a PHYSIQUE skin linked to non-BIPED parts, it won't work. To check whether your bones are BIPED parts or not, click on *Summary Info*. Valid bones are marked as *Biped Objects*.

2) Use the same wordlist as Character Studio ! BIPED parts are for example named *Bip01 Head, Bip01 L Calf*, and so on. You can change the *Bip01* part only. For

example the name *MyCharacter L Calf* is a valid name. *Dragon L Wing* is **not**, because *L Wing* doesn't appear in the original Character Studio wordlist.

Why do those limitations exist ? For a single reason : *motion blending*. I used to do motion blending with my characters. I then needed to link a given bone from a given motion to the *corresponding* bone from *another* motion. That's why I used what I called the *CSID* (Character Studio ID) in Flexporter. CSIDs are tags between 0 and 104, because the complete Character Studio 2.0 wordlist contains 105 bones at most. That way, I assign a unique CSID to each bone, and that value is later used to blend motions.

The alternative is to use the *Node ID* (as saved by the ASCII Exporter for example) as a bone ID. It always work, but that Node ID depends on the scene organisation ! Export the scene once, merge another object in it, export the scene again : the Node ID for your bones may have changed. Worse : you have a given character, plenty of motions, you export them. Nice. Then your favorite artist decides a given bone is finally useless, and deletes it. Guess what ? Now you must re-export all the previously exported motions, since the Node IDs have changed. With my method, there's no need to re-export, and I can even blend two motions exported from two different skeletons (as Character Studio does, actually). In short, I couldn't use the NodeID for Motion Blending, hence those rather painful limitations.

For those who don't care about motion blending, I'll expose an option to use the Node ID as a bone ID nonetheless. As soon as possible.

How do I use the camera quaternion ? Can I export the roll angle ?

The roll parameter is not explicitely exported as, say a floating-point roll angle, because I don't think that information is available from MAX. Now, this is implicitely encoded in the quaternion part of the camera's PRS structure. If you use that PRS structure to rebuild a view matrix, you get the exact same view as in MAX.

Use the piece of code below. Feed it with an exported camera PRS, and directly send the resulting matrix to D3D. Works like a charm.

```
//////////////////////////////////////////////////////////////////////
/**
 * Compute a view matrix.
 * This method computes a view matrix compatible with Direct3D. Once the
matrix
 * is computed, you just have to send it to the renderer.
 * \param  prs   a PRS structure, directly exported from 3D Studio MAX.
 * \return  Self-Reference
 *//////////////////////////////////////////////////////////////////////
Matrix4x4& Matrix4x4::View(PRS& prs)
{
     *this = prs.Rot;
     Transpose();
     Scale(prs.Scale);

     HPoint Col1b = GetCol(1);
     HPoint Col2b = GetCol(2);
```

```
      SetCol(1, Col2b);
      SetCol(2, -Col1b);

      Point Col0 = GetCol(0);
      Point Col1 = GetCol(1);
      Point Col2 = GetCol(2);
      SetTrans(
        -(prs.Pos|Col0),
        -(prs.Pos|Col1),
        -(prs.Pos|Col2));

      return *this;
}
```

What is ICE ?

My own development framework. Details are available here : www.codercorner.com/Ice.htm

How do I use the consolidation results ?

Please have a look at ZCBHelp.cpp in the Flexporter SDK.

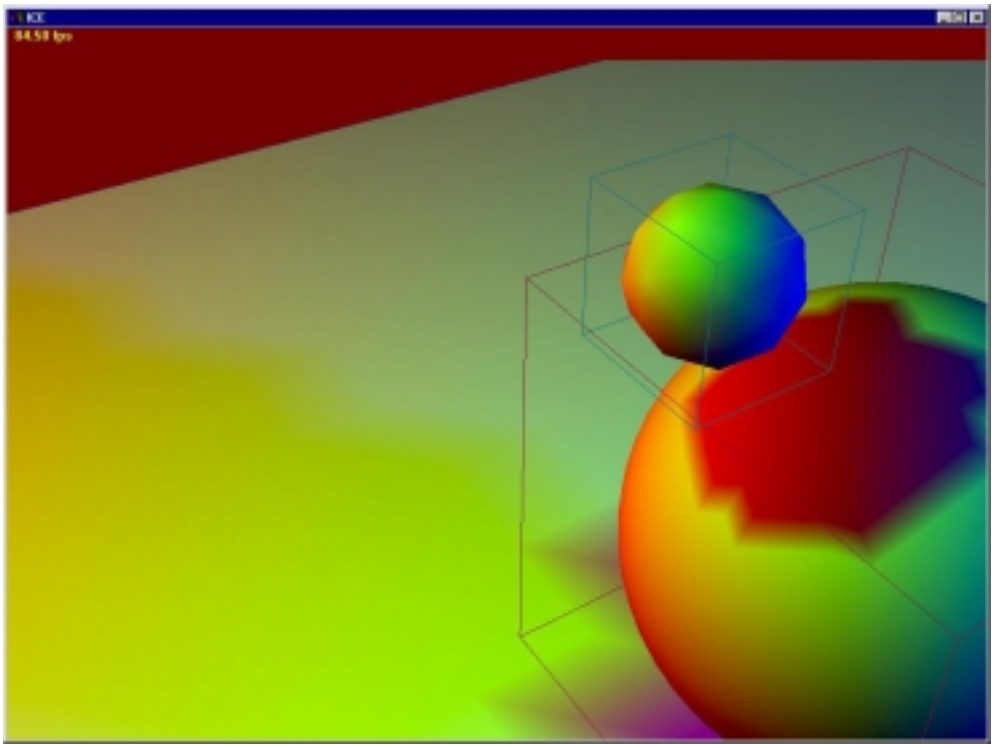
What ? It crashes on exporting a shape !

You must manually collapse your shapes before exporting them. For example if you create a text-shape, bend it with a modifier, then export it…. MAX will crash. This is not a Flexporter bug, this is MAX crashing as soon as I try to transform the shape into poly-lines. Apparently the usual way of automatically collapsing the objects (as used with meshes) doesn't work for shapes, and I don't know how to detect non-collapsed shapes, which would at least prevent MAX from crashing – but still wouldn't export the shape.

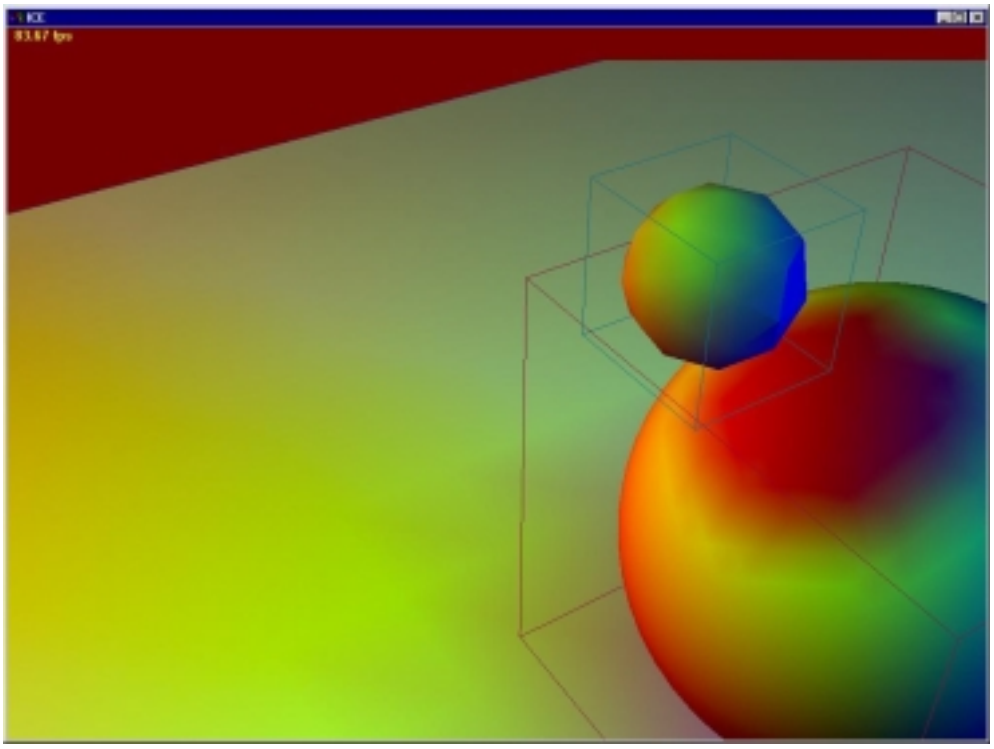
Any idea, someone … ?

What's this color smoothing thing ?

Here's a scene exported without color smoothing :



And here’s the same scene with color smoothing :



Can I use Flexporter for a commercial game ?

Yes, sure.

Is Flexporter still available for MAX 2.5 ?

No, definitely not.

Are you going to support MAX 4 and Character Studio 3 ?

Since Flexporter 1.08, it runs on MAX4 and exports Character Studio 3 (not the new features anyway).

Can I contribute to Flexporter ?

You can contribute in two ways :

- by providing code for a new option you would like to see included
- by providing new custom exporters to well-known formats (D3D x files, OBJ files, etc)

How do I convert a quaternion to a matrix ?

This is a very general question which has actually little to do with Flexporter. If you're not familiar with quaternions, you really should learn about them first, then come back to MAX.

Anyway here's some code to convert a quaternion to a 3x3 matrix :

```
////////////////////////////////////////////////////////////////////////
/**
 *    Operator to cast a quaternion to a 3x3 matrix.
 *    Only works with unit quaternions, pay attention to what you do...
I don't normalize quat before proceding for speed reasons.
 *    \return           a 3x3 matrix built from the quaternion
 */
////////////////////////////////////////////////////////////////////////
__forceinline Quat::operator Matrix3x3() const
{
      Matrix3x3 Ret;

      float xs = p.x + p.x;   float ys = p.y + p.y;   float zs = p.z + p.z;

      float wx = w * xs;      float wy = w * ys;      float wz = w * zs;
      float xx = p.x * xs;    float xy = p.x * ys;    float xz = p.x * zs;
      float yy = p.y * ys;    float yz = p.y * zs;    float zz = p.z * zs;

      Ret.m[0][0] = 1.0f -    yy - zz;
      Ret.m[1][0] =           xy - wz;
      Ret.m[2][0] =           xz + wy;

      Ret.m[0][1] =           xy + wz;
      Ret.m[1][1] = 1.0f -    xx - zz;
      Ret.m[2][1] =           yz - wx;

      Ret.m[0][2] =           xz - wy;
```

```
	Ret.m[1][2] =           yz + wx;
	Ret.m[2][2] = 1.0f -    xx - yy;

	return      Ret;
}
```

What is the MAX-to-D3D transform ?

MAX uses a right-handed frame, D3D uses a left-handed one. When you ask Flexporter to perform a MAX-to-D3D transform before exporting, several things happen :

- mesh parity is flipped
- y and z are swapped for all vertices
- y and z are swapped for all PRS components (i.e. for the position vector, the quaternion, and the scale as well)

What about mirrored objects ?

Mirrored objects weren't correctly exported before version 1.09, because the code responsible for removing the scaling did not take negative scaling into account. In short, it has been fixed and should now work as long as your meshes are not instances.

For mirrored instances, you need the scale vector (which will be negative). You can't mix mirrored instances and the « Remove scaling » option, since the « mirror » part is captured in the scaling. This is not a problem for non-instances meshes, since the actual geometry is transformed when removing the scaling factor…