Changelog

- July 16, 2007: Initial Version – Luc Bolduc/SoftImage
- October 25, 2007: Version 1.1 – Peter Nagy
- January 31, 2008: Version 2.0 – Michael Endres
- February 15, 2008: Version 2.4 – Michael Endres
- February 18, 2008: Installation section update – Luc Bolduc/SoftImage

CryExporter – the XSI to Cryengine2 Exporter

Introduction

The CryExporter is a XSI Addon, which not only contains the Export functionality, but also provides custom objects and a custom shader to make maximum use of the Cryengine2 features.
The CryExporter prepares assets created in Softimage XSI for the Cryengine 2 Resource Compiler (RC). The assets are stored in the Collada format and within the RC converted to the final game asset.

Concept

- A single scene can contain more then one asset.
- Every asset has a CryExportNode as a root Node. This node, which is similar to a model node, stores some export settings and the filename of the cgf/cga/chr created afterwards.
- Link the geometry and its functional parts to the CryExportNode.
- Export one or all CryExportNodes from XSI – the resource compiler will create the target files automatically.

Possible types of assets in an XSI scene:

- static rigid geometry (i.e. buildings) (CGF)

Installation

Requirements
In order to work with the CryExporter, you need the following software installed and working:

- Softimage XSI Version 6 and up
- Alternatively, XSI Mod Tool
- Crosswalk 2.5 and up

You also need the Crytek_XSI_vX.X.zip archive. This archive contains:

- The resource compiler (Crysis\Bin32).
- A set of compiled example asset (Crysis\Game\Objects\XSI_Assets).
- A test level (Crysis\Game\Levels\XSI_Testobjects).
- An XSI database containing the XSI scenes of the corresponding example asset (Crytek_XSI_Database).
    - Note that texture images are not provided and not assigned on the example objects materials. To make sure that those objects are textured in Sandbox 2 or in the game, use the already compiled material files (.mtl).
- The Crytek XSI Addon (XSI_Addon\Crytek Addon.xsiaddon).
- The documentation (XSI Plugin Documentation_VXX.doc).

Installation procedure

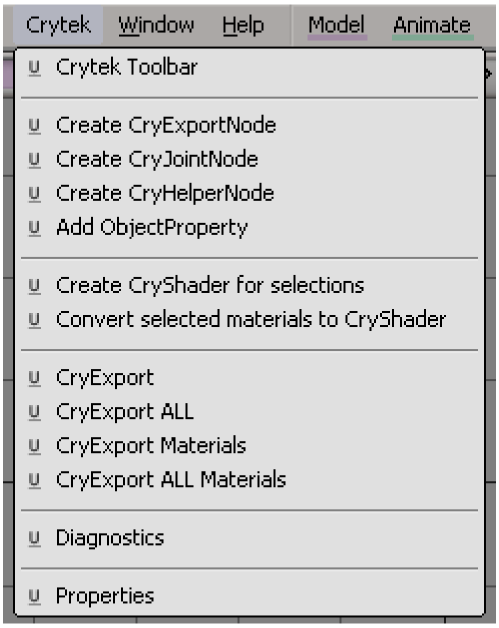
- Install Crysis.
- Install Sandbox2 (from the game dvd, Sandbox2 folder).
- Install the resource compiler, example asset and test level.
    - Copy the Crysis folder from the Crytek_XSI_vX.X.zip over the game install Crysis folder. This will add the Bin32\rc folder, some already compiled example asset in Game\Objects\XSI_Assets and a test level (Game\Levels\XSI_Testobjects\XSI_Testobjects.cry).
- Install the XSI Crytek Addon.
    - Drag n drop (install) the Crytek Addon.xsiaddon in XSI.
- In XSI, Set the project path: Crytek > CryProperties.
    - Bin32 Path

        - Ex.: C:\Program Files\Electronic Arts\Crytek\Crysis\Bin32
    - Export Path
        - This is the working folder. When exporting, a COLLADA file (.dae) is created and the resource compiler is automatically executed with it as an input. The resource compiler generates a .cfg or a .mtl file, depending on your export settings.
        - Ex.: C:\Program Files\Electronic Arts\Crytek\Crysis\Game\Objects\XSI_Assets

UI

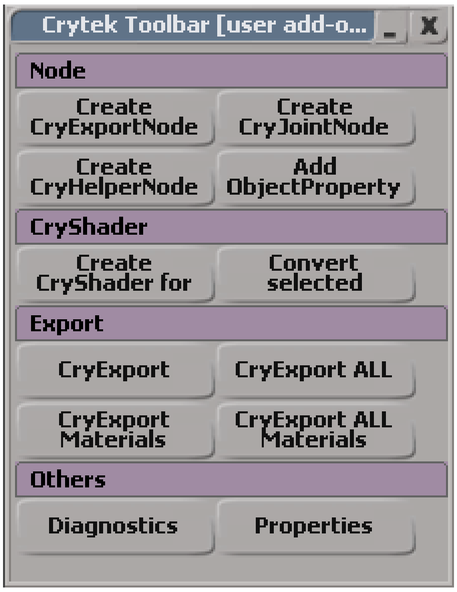
Crytek Menu

After the installation of the Crytek Addon, you will find the Crytek menu in the XSI menu bar.



Crytek Toolbar

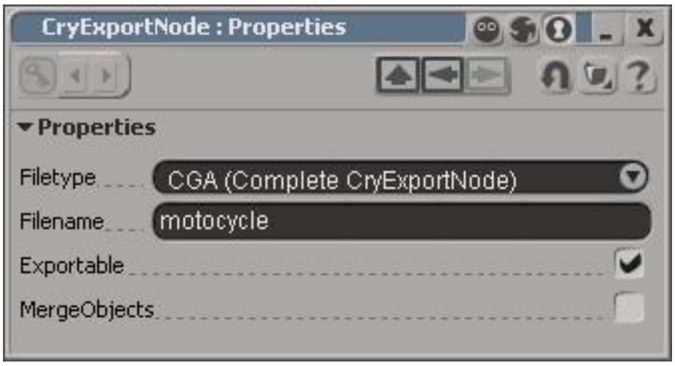
The Crytek Toolbar gives you quicker access to all functions of the Crytek Menu.
You can place it wherever you like.



Nodes

CryExportNode

After creating a CryExportNode model, you are asked in the CryExportNodeProperties window to specify a name for the final asset.
The name will be added to the beginning of the CryexportNode name. i.e. motocycle_CryExportNode.



CryExportNode : Properties

Properties

Filetype CGA (Complete CryExportNode)

Filename motocycle

Exportable ☑

MergeObjects ☐

You can choose the type of asset, i.e. .cgf, .cga or .chr file (refer to CryExport for more details).
Additionally you can choose whether this particular asset should be exported during when using the **Export all** command.
Check merge, if the asset has no functionality and is composed of many separate geometry nodes. The exporter will automatically remove the hierarchy and compress the data for the target asset.
The values are stored in custom properties called **ExportProperties**. You can easily access them from the Explorer.

CryJointNode

This node is a box shaped null helper which serves as a joint in your object.
The Default name is **CryJointNode**.
Link it to the **CryExportNode** model.

The joint needs to be placed between two geometry nodes, so that its bounding box is intersecting with the bounding box of the two objects it should keep together.
The null has a text property in which you can adjust the physical parameters for the joint, i.e. limit=10000

Joint parameters:

- limit – this is a combined parameter from different joint parameters (bend, twist, shift…). A limit of 1000 is a good starting point.

CryObjectProperty

Use this function to add a custom text field to a geometry node.
It enables you to add engine properties to a node, like mass/density, LowSpec Settings, etc.

CryObjectProperty settings

- Mass - defines the mass in kilograms; do not use together with density; i.e. mass=50
- Density - defines the density of the material, the mass will be calculated based on the volume of the bounding box, i.e density=2
- Joint Parameters (see CryJointNode)

- Pieces – defines the pieces that are spawned (and replaces the original object) when a physical impulse on a joint exceeds its limits, i.e. pieces=_wall01_piece01,_wall01_piece02,_wall01_part
- LowSpecLod0 – use this on the lod you want to use as the base lod on a lowspec machine, i.e in the properties of a Lod1 or Lod2
- Entity – turns a piece into an entity, which the player can interact with. The
- Count – spawns the piece x-times, i.e. count=50 spawns 50 pieces within the volume of the intact object

NOTE: Either use Mass or Density, not both together.

CryShader

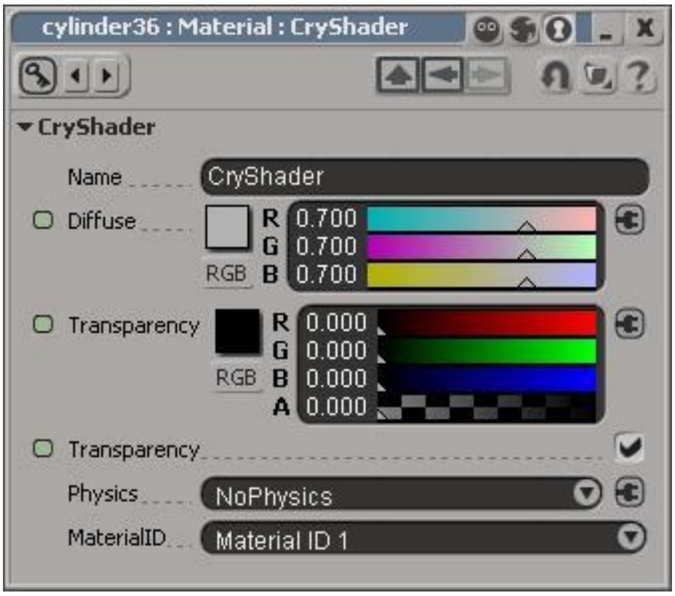
The most important info the Cryshader stores within XSI is the physics parameters of polygons the shader is assigned to.

To assign a CryShader to an object, select it use **Create CryShader for selections**.
This creates a new material without Physics settings (NoPhysics).
If you want to assign cryshader to single polygons, you need to create a cluster before you can assign a new Cryshader to them.

Second you can select existing materials in the Explorer and use **Convert selected materials to Cryshader**.
As the name suggest, selected materials are converted to the CryShader format.

The plugin automatically adds prefix numbers to the materials to keep the order of the materials consistent. CryEngine2 applies the materials in the order of the .mtl file this order will not change creating the .mtl file (CryENGINE2 Material file).

If you convert an existing set of materials to the Cryshader, make sure there are no other unused Cryshader materials in the same library. This would interfere with the numbering process.
Collect all materials of one asset in a Material library and rename it. The name will be used to create an mtl file, or to connect the asset with an existing .mtl.



cylinder36 : Material : CryShader

CryShader

Name: CryShader

Diffuse: R 0.700 G 0.700 B 0.700

Transparency: R 0.000 G 0.000 B 0.000 A 0.000

Transparency: ☑

Physics: NoPhysics

MaterialID: Material ID 1

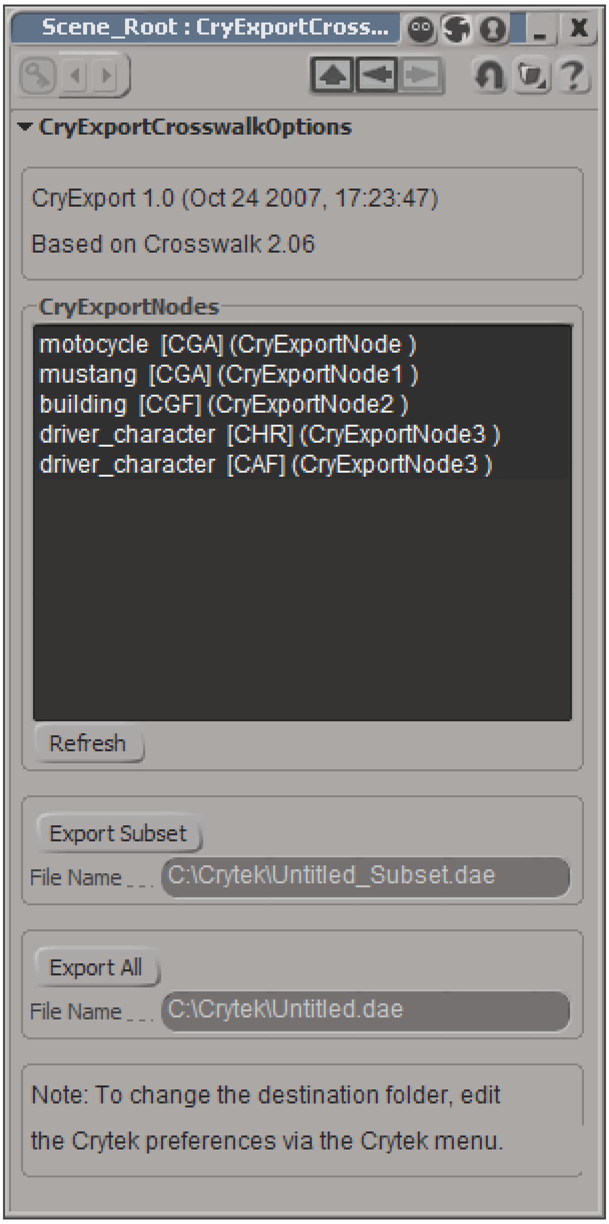
Physics Parameters:

- NoPhysics – Default Value, the polygons are not physicalized

- Physics – the polygons are physicalized to block the player and vehicles and can get a shader assigned, i.e a tree trunk
- Physics (NoDraw) – the polygons are physicalized to block the player and vehicles but are invisible and can not get a shader assigned
- No Collide – use this for very simple proxies that do not block the player, but are used for detecting bullet shots. The calculation will then detect the render mesh below.
- Obstruct – the polygons are physicalized in order to block the AI view; player, AI and vehicles can pass through, i.e. used for vegetation cover objects

CryExport

Once ready to export, the artist can open the following UI via the Crytek menu or toolbar:



CryExport 1.0 (Oct 24 2007, 17:23:47)

Based on Crosswalk 2.06

motocycle [CGA] (CryExportNode)
mustang [CGA] (CryExportNode1)
building [CGF] (CryExportNode2)
driver_character [CHR] (CryExportNode3)
driver_character [CAF] (CryExportNode3)

File Name _ _ _ C:\Crytek\Untitled_Subset.dae

File Name _ _ _ C:\Crytek\Untitled.dae

Note: To change the destination folder, edit the Crytek preferences via the Crytek menu.

All the asset and their corresponding Crytek compiler out filename and file type are listed in the CryExportNodes list.
If you want to export only some assets of the scene, you can select the assets in the list and use **Export Subset**. The specified “<filename>_Subset.dae” COLLADA file will be exported.

Export All creates a COLLADA file with the name you specify, containing all the assets from the scene. The RC will then create separate files in the target format.

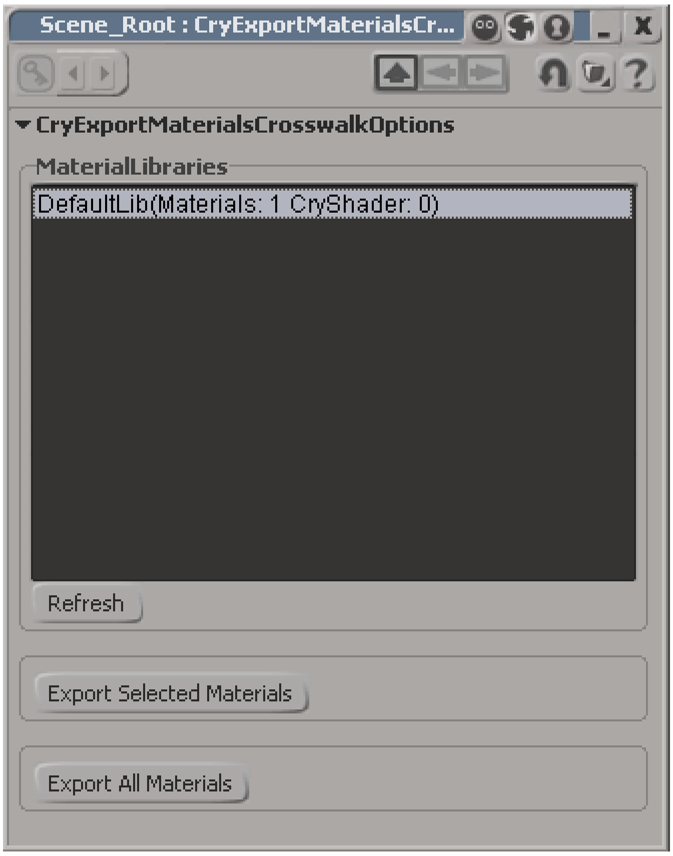
The CryExport exporter takes into account every asset individual property and will only export the required elements for that particular asset/file type in the intermediate COLLADA file.

You can also access a shortcut to the **Export All** function through the Crytek Menu.
This save some time during the testing and learning phase.

Here’s what gets exported for the possible file types:

CGF : Triangulated mesh geometry, hierarchy, material and image

CryExport Materials


Scene_Root : CryExportMaterialsCr...
CryExportMaterialsCrosswalkOptions
MaterialLibraries
DefaultLib(Materials: 1 CryShader: 0)
Refresh
Export Selected Materials
Export All Materials

Similar to the CryExport function, this window let’s you select the available material libraries and export them to the export folder.
This is useful to create an initial version of the material. The fine tuning needs to happen in the Sandbox Material Editor anyway.

You can choose between exporting the selected materials in the list or exporting all the materials at once.

You also have a shortcut to create a material file for all the available Material libraries.

Here’s what gets exported for the possible file types:

MTL : material file, taking into account the diffuse colours and texture paths.
They will be created relative to the Game folder.

Others

Diagnostic

Diagnostic is a validation process, which checks settings and properties for the selected assets.
It checks for valid filenames, proper material and material library names, whether pieces are available, etc.

CryProperties

Here you need to set the properties of export process:

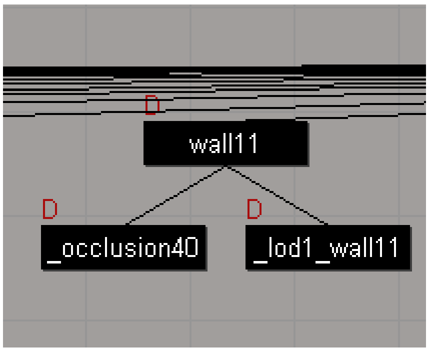
- The path to the Bin32 folder which contains the rc folder, where the Resource Compiler is stored.
- The export path for all assets.

The properties are saved in the XSICryExport preferences. (<path of crytek addon>/XSICryExport.xsipref)

Special Nodes

Occlusion objects

These are simple geometry objects which are placed inside non see through walls to indicate the renderer that everything behind it does not need to be rendered.
This is helpful in busy scenes to ensure high render performance.
The name of the geometry object needs to start with “_occlusion”. Link it to the geometry object which serves as the “visual barrier” in the game.

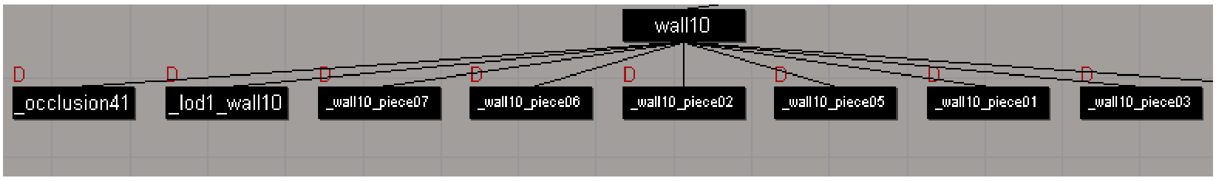


You can check them with various console commands:

- e_cbuffer_draw_occluders 1 : debug draw of occluders in coverage buffer.
- e_debug_draw 11: Render occlusion geometry additionally
- e_debug_draw 12: Render occlusion geometry without render geometry
- e_debug_draw 13: Render occlusion ammount (used during AO computations)

Pieces for breakable objects

Pieces geometry nodes can be placed anywhere in the hierarchy.
They can be named to your liking, but need an underscore as the first character.
i.e. _piece01, _wall01_part01, _blah



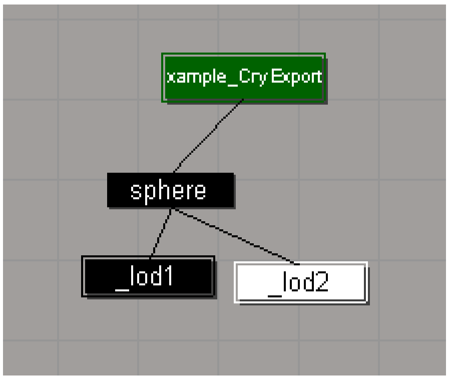
Branches

The branches nulls are needed for the setup of bending vegetation leafs.
The nulls are used as joint in the engine, through which the physics engine interpolates a rope spline, with which the player can interact.
The branches have a special and strict naming convention: branch<#>_<#>
i.e branch01_01, branch01_02, branch02_01, branch02_02, branch02_03

The system is unconventional, but very efficient. The bounding boxes of the branch nulls needs to enclose vertices which are influenced by the spline. Ideally you choose the vertices lying in the center line of a branch (it works especially well for trees with large leafs, like palm trees).
The engine then makes a lookup in the UV coordinates and checks for all vertices, which have connected UVs. All the connected vertices are taken into account when the physics calculates the deformation.
The cool thing about it is, that you do not need to create weighting on the vertices and you only need to set up one (!) leaf and all other instances in the asset will automatically work, if they share the same UV coordinates. You can even deform the other leaves and still have the effect applied.

LODs

You can make use of the Level of detail functionality of CryENGINE™2 by linking the LODs to the Base object, add an UNDERSCORE to the name and number themaccordingly.
The rule of thumb is to reduce the polycount for every LOD step by 50%.
If you only remove 20% of the polygons, the LOD will not load for being insufficient.
There is a exception, though. If you use less different materials than on the previous LOD level, the LOD will still work, even if there are not 50% of the polygons saved.



You can also add a postfix with the name of the LOD0 (in this case “sphere”) to the name of the LODs, i.e. _lod1_sphere or _lod2_justforfun.
Important are only the first 5 characters.