Corona Standalone usage (2013/06/18)

Command line parameters

- All inputs are from command line arguments/text files, there is no GUI for them
- Corona can take zero or more arguments with the following syntax:
    - 1st argument: path to a .scn file with scene description. This file can include other input files. If the argument is not present, a default one is substituted: “export.scn”
    - 2nd argument: path to the output file without the file extension. After the successful rendering PNG and EXR files are saved. If not present, a default one is substituted: “output”
    - 3rd, 4th, … arguments: filenames of additional configuration (.conf) files to parse.
- All command line arguments are processed in the order specified on command line (additional includes in parsed files are processed immediately). If single settings parameter is specified multiple times, the value appearing later overrides the earlier one.

File formats syntax

- There are 4 types of input files allowed:
    - Scene configuration files: .scn
    - Geometry files: .obj
    - Material files: .mtl
    - Settings files: .conf
- All input formats are ASCII text files, line-based, where each line is parsed separately and can contain 1 command/entity declaration. Comments start with #
- [filename] in the following sections substitutes a string without the brackets nor quotation marks. [XYZ] and [RGB] are simply a triplets of floating point numbers, again without any brackets. [int] is single integer and [float] is single floating point number without any brackets.

SCN file syntax

Following declarations are allowed:

- `include scn [filename]`
- `include obj [filename]`
- `include mtl [filename]`
- `include conf [filename]`
    - Immediately parses an SCN/OBJ/MTL/CONF file
- `light point position [XYZ] color [RGB] attenuate [int]`
    - Puts a point light into the scene at position [XYZ] with color [RGB] and square falloff if [int] is nonzero (no attenuation is used otherwise)
    - Note: It is recommended to use area lights, not point lights. Area lights are created automatically from materials with emission color/texture.
- `camera perspective origin [XYZ] target [XYZ] roll [XYZ] fov [float]`
    - Sets the current camera to a perspective one with given origin point, target point, roll vector (direction pointing “up”), and horizontal field of view (in degrees). Disables DOF.
- `camera perspective origin [XYZ] target [XYZ] roll [XYZ] fov [float] focalDist [float] fstop [float] filmWidth [float]`
    - Same as above, plus enables DOF effect with given focal distance, f-stop number, and width

of the camera film/sensor

- `camera ortho origin [XYZ] target [XYZ] roll [XYZ] width [float]`
  - Sets the current camera view to a orthogonal one with given origin, target, roll vector (direction pointing “up”), and vertical width of the captured image
- `sun dirTo [XYZ] color [RGB] halfAngleSize [float]`
  - Activates the sun in the scene and sets its position (specified as dirTo, vector pointing towards the sun from the scene origin), color in RGB, and half angle size of the sun projected onto a unit hemisphere (specified in radians. Physically correct value is 0.00445).
- `enviro [name] color [RGB]`
  - Sets the constant environment color to a given RGB value. [name] is the name of background type to set - “main” for regular background, “direct”/”reflect”/”refract” for overrides (without quotation marks).
- `enviro [name] texmap [filename]`
  - Enables the use of the textured environment. [filename] specifies a path to a valid bitmap with the environment map rendered in a latitude-longitude projection. [name] is the name of background type to set - “main” for regular background, “direct”/”reflect”/”refract” for overrides (without quotation marks).
- `portal [XYZ] [XYZ] [XYZ]`
  - Adds an environment portal triangle into the scene with three vertices in no particular order (normal does not play a role for portal lighting)
- `renderpass [type] [name] [additional params]`
  - Adds a single render pass to the frame buffer with given name. Some of them can have additional parameters. Allowed renderpasses (value of [type]) are:
    - ShadingNormalPass
    - GeometryNormalPass
    - DotProductPass
    - WireColorPass
    - DiffuseColorPass
    - ReflectColorPass
    - RefractColorPass
    - OpacityColorPass
    - TranslucencyColorPass
    - PrimitiveCoordinatesPass
    - UvwCoordinatesPass (additional parameter: int id)
    - PrimitiveIdPass
    - MaterialIdPass
    - ObjectIdPass
    - InstanceIdPass
    - VelocityPass
    - ZDepthPass (additional parameters: floats from, to)
    - NormalsDiscrepancyPass
    - DebugPass
    - DirectLightingPass
    - ShadowsLightingPass
    - EmissionLightingPass
    - GiDiffusePass

        - GiTranslucencyPass
        - GiReflectPass
        - GiRefractPass
        - WorldZPass
        - WorldPositionPass
        - AlphaPass
        - AoPass

MTL file syntax

- Corona uses the standard MTL format (http://people.sc.fsu.edu/~jburkardt/data/mtl/mtl.html)
- The format specifies multiple materials. Each new material has to be started with "`newmtl name`"; all lines after that one add to the description of the material with that name, until new newmtl is encountered.
- Corona uses a subset of the MTL properties, and adds its own.
- All lines can start with "#CORONA" tag; if they do, the tag is removed and the line processed normally. This is to "hide" the non-standard extensions of the format as comments from non-corona parsers to keep the file a valid MTL.
- Allowed material properties are:
    - `Kd [RGB]`
    - `Ks [RGB]`
    - `Ke [RGB]`
    - `refract [RGB]`
    - `opacity [RGB]`
    - `translucency [RGB]`
        - Colors of diffuse (Kd), specular (Ks), emission (Ke), refraction (refract), opacity (opacity), and translucency (translucency)
    - `Ns [float]`
    - `reflectGlossiness [float]`
        - Glossiness of reflection. There are two commands for it, one (Ns) uses the phong exponent, other (reflectGlossiness) uses a value between 0 and 1, identical to the one in Corona API/GUI
    - `refractGlossiness [float]`
        - Glossiness of refraction. Uses a value between 0 and 1 identical to Corona API/GUI
    - `Ni [float]`
        - Index of refraction
    - `reflectFresnel [float]`
        - Index of refraction for fresnel reflections
    - `attenuation [RGB]`
        - Direct input of the refraction attenuation value. In 3dsmax GUI this is computed from the color and distance user inputs as ln(color)/distance (ln = per-component natural (base e) logarithm)
    - `emissionExponent [float]`
        - Phong exponent describing the emission distribution. 0 produces standard diffuse lights, higher values produce spotlights.
    - `twosidedGlass [int]`

        - Sets the refraction behavior: 0 = onesided, 1 = twosided, 2 = hybrid (onesided for primary rays, twosided for secondary)
    - `NonShadedMtl [abcde]`
        - If this line is present, the current material is turned into Constant (nonshaded) one. In this mode only the emission color (Ke, map_Ke) is used. [abcde] are 5 zeroes or ones without spaces, that determine the visibility (1) or invisibility (0) of the material for various ray types (from left to right: for normal rays, direct rays, refracted rays, reflected rays, shadow rays).
    - `map_Kd [filename]`
    - `map_Ks [filename]`
    - `map_opacity [filename]`
    - `map_Ns [filename]`
    - `map_Ke [filename]`
    - `map_translucency [filename]`
    - `map_refract [filename]`
        - Texture bitmaps for the respective slots (see above).

OBJ file syntax

- Corona uses the standard OBJ format (http://en.wikipedia.org/wiki/Wavefront_.obj_file)
    - Vertices are specified by `v [XYZ]`, normals by `vn [XYZ]`, mapping coordinates by `vt [XYZ]`.
    - `usemtl [name]` causes all subsequent primitives to use material with given name (must be already specified in some of previous MTL files before use!)
    - `"f v//vn v//vn v//vn"` or `"f v/vt/vn v/vt/vn v/vt/vn"` declares a triangle face with vertices/mapping coordinates/normals at vertices specified by given indices. Indexing starts from 1 as specified in the OBJ format.
    - mtllib command includes a material library. Each material library must be included only once (if you include from OBJ, do not include again from SCN file)
- Relative indexing (negative indices) and quads are currently not supported
- There is one extension: `#CORONA sphere transform [tm]` outputs a sphere specified by a transformation matrix [tm] that transforms unit sphere (center in scene origin, radius 1) to the desired one. [tm] is a 4*3 affine matrix, specified by 12 floats in row-wise order.

CONF file syntax

- Syntax of each line: `[datatype] [name] = [value]`
- Datatype can have following values:
    - int: single integer example: 42
    - float single floating point number example: 42.123
    - bool either true or false example: true
    - Vec3 triplet of floats (either vector or rgb color) example: 0.2 0.5 0.7
    - string ASCII string in quotation marks example: “filename.bmp”
- Name must be one of the predefined types; see example data for the names of standard parameters. Each parameter has a predefined type, trying to use a parameter with different type results in error.
- Value for ints and floats has to be in the valid range for that parameter (same as the ranges for

spinners in 3dsmax)

Input bitmap specifications

- Corona currently supports these bitmap formats:
    - EXR
    - PNG
    - JPG
    - BMP
- All formats have to use appropriate gamma (2.2 for PNG/JPG/BMP, 1.0 for EXR)
- Numerical values of some algorithm choice parameters:
    - Renderer:
        - Progressive 2
        - Bidir/VCM: 3
        - Bucket: 0
    - GI solvers (both primary and secondary)
        - None: 0
        - Path tracing: 1
        - Photon mapping: 2
        - HD cache 3
        - VPL 4
        - Irradiance cache 5
    - Image filters:
        - None 0
        - Box 1
        - Gaussian 2
        - Tent 3
    - Frame buffers
        - None 0
        - wx 2
    - Bidir/VCM modes
        - Bidir: 4
        - VCM 5

General tips

- Each parameter must be specified at least in one configuration file. It is a good idea to have a “universal” .conf file with default values that gets included in every scene before scene-specific settings that override parameters you want to change.
- Set vfb.type = 0 to get no-GUI unattended rendering setup. If you use other VFB types, you will need to manually close the window with rendered image to make Corona stop running after the rendering.
- Try exporting from Corona for 3dsmax (Export scene button; saves results as export.scn/conf/obj/mtl somewhere in the appdata/local/Autodesk folder) to get example data, especially to see how different parameter values export to a .conf file.
- Use geometry with materials that have Ke set to non-zero to create lights