



Honey v0.2

Sunflower for Houdini

1. About Sunflow:

Sunflow is a open source raytracing engine written in Java.

More information, download and user forums can be found here: <http://sunflow.sourceforge.net>

2. Requirements:

To use Honey, you need:

- Houdini 8.1
- Python 2.4 (tested on 2.43)
- Java JDK 1.6.0
- Sunflow 0.07.1 (tested on 0.07.1-2).

3. Sunflow features supported in Honey v0.1:

- Polygons geometry export (via obj files).
- Point and Area lights.
- Sunflow surface and bump shaders included with Honey.
- Janino shaders code loaded from a source file.
- Motion blur and Depth of Field.
- Global Illumination (via light shaders).
- Sun Sky (via light shaders).
- Caustics Photons (via light shader).
- Geometry instances (with scale, rot and shaders attributes per instance).
- Light instances (via geometry instance pipe).

4. Installation and setup:

4.1 Make sure that all needed executables are present in your PATH.

The easiest way to do this is to open shell/command line and type: "python" then "java". Python and Java should answer respectively. If not, add proper paths to your global variable PATH.

4.2 Extract Sunflow archive to location of your choice (i.e. C:/sunflow-0.07.2/)

4.3 Open Houdini session and install **Honey.otl** (menu File/Install Operator Type Library) . Open Operator Type Manager (under Tools menu). You should see number of assets in Honey library.

4.4 Go to Outputs and add **Honey ROP**. It looks similar to Mantra ROP. Go to Specific Tab of Honey ROP and specify path to your Sunflow directory with sunflow.jar. If, for some reason, you don't like to add Java installation path to your PATH global variable, you can specify it here also (not recommended). See Honey help card for details about specific parameters (they look and behave similar to Mantra parameters as much as possible).

5. Workflow:

5.1 Few things to remember:

- Only **triangle polygons** are supported (bezier mesh comes with SF 0.7.3)

- UV should be point's not vertex's attributes.

- Sunflow doesn't accept Uvs nor Normals when renders geometry from *.obj files.

- Sunflow can not save alpha channel currently.

5.2 Only **/obj level shaders** are supported. That means that one object can have only one shader. It goes from Sunflow way of defining geometry. Maybe in a future realize of Honey I will take care of it. It would require to cut obj into pieces according to groups, than export them as separated objects with its own shaders. Can be done (?).

5.3 Please avoid using nested operators. Shaders should have unique names as **/shops/myshader** and **/obj/myobj/shops/myshader** will become one shader (and I have no idea which one).

5.4 There are (at least) two ways of working with Houdini/Sunflow. You can render pictures to standard SunFlow framebuffer. This way you have fast feedback but you can't save rendered images. Another way is to keep opened **Sunflow GUI** and point it to script file exported from Houdini. This way you can instantly rebuild your script (*.sc) after every export and render images inside Sunflow GUI itself.

5.5 Most of Sunflow build-in shaders exists in two versions: textured and not textured. They are different entities. More over Sunflow doesn't support empty parameter value. So whenever there is an image path parameter, image must be specified. That's why some shaders force you to choose between options. When you apply **Uber Shader** (with two image parameters) for example, Sunflow will hang without paths to both images.

5.6 There is a special class of Sunflow shaders called **Janino**.

Janino is a Java interpreter written in Java. Sunflow uses it to compile java code during execution. You are free to write your own shaders with Java and Sunflow API.

Some examples can be found on [user forum](#).

Sunflow Janino surface shader can load such a code from a file. It requires ascii file with Java code but **without** „<code>“ „</code>“ markups or Sunflow shaders specific statements usually present in examples on forum. I'm including one of Janino shaders (stained_glass.java) which has nothing to do with my self. Copyright note is included in the file. Renders, discussion and code is [here](#). Quite impressive, isn't it? (You must change texture path in shader code to use it). This shader uses also caustics photons. You can add them to your scene with **Sunflow Photons** light shader.

5.6 Another shader of special purpose is **Sunflow Inline** which is an empty shader. It lets you send to Sunflow anything you want. This can be diffuse shader like:

```
shader {  
    name $OS  
    type diffuse  
    diff 0.5 0.5 0.5  
}
```

or light of a type which is not currently supported by Honey, directional:

```
light {  
    type directional  
    source 1 4 3  
    target 0 0 0  
    emit 0.9 1.0 0.5  
}
```

Please note **\$OS** expression in diffuse shader. It's obligated as Sunflow stores shader names in geometry file. This way shader name will correspond to string saved in geo definition.

5.7 Area lights.

Honey exports Houdini's area lights as Sunflow's meshlights. It means that for every area light Honey creates polygonal geometry. It stores in obj directory with your objects. It does it in pretty crude way. Inside every light created by Houdini default light creation script there is a merge SOP „arealight2“. Honey simplie adds a few SOPs there to produce nice triangle mesh from a **grid**. Actually it fails with other shapes. I could improve it later but if you really need other area light shape now (like a long thin line) you can easily change network inside a light. Possibly it's enough to change primitive type on a sphere SOP for example from Nurbs to Polygon. Do whatever than Honey will find „arealight2“ SOP and takes its output, triangulate and subdivide it (optionally). Please note also that setting **Sunflow Constant** shader color above 1 1 1 with Global Illumination enabled will make your geometry shinnig.

5.8 Instances. To use instances with Sunflow you need:

Toggle on „Point Instancing“ on Render Tab of your point's object (value of „Point Geometry“ parameter is currently ignored). Create point string attribute **instance** with value of **local** name of an object you want to be instanced: **mymodel** but not **/obj/mymodel** (no nesting... sorry). You can create also other point attributes:

- **rot** (vector)
- **scale** (float) - this is uniform scale attribute
- **shader** (string)
- **modifier** (string) - this is an internal Sunflow name of bump shader

Classic Houdini trick with expressions easily can be recreated by putting this line:

```
shader { name my_instances_`$PT` type diffuse `rand($PT)` (...) }
```

into **shader** attribute.

You have an access to all Sunflow shaders which comes with Honey. You can copy/paste shader strings from SHOPS then modify them as you like. Again, please note **\$PT** expression in shader string. This is not mantra way of doing instances. Sunflow doesn't recognize point attributes internally. Thus every instance has to have its unique copy of shader with unique name.

Note: In Honey v0.2 you can mix point instancing with loading geo on demand. This way 1000 instances of heavy geometry can be rendered.

5.8.1 With Honey v0.2 you can also **instance lights** with a little hack. All you have to do is to prepare above setup as for geometry omitting **instance** attribute. In shader attribute place **point light string** as you can find in normal *.sc file:

```
light { type point color 1 1 1 power 5 p 0 1 0 }
```

Obviously it would be nice to copy point position on light position (**p**) with **`point()`** expression and vary color values.

5.9 **Scale x, y, z** on objects Transform Tab are ignored for now. Only **Uniform Scale** works.

5.10 Very handy trick is to turn off geometry export from Houdini:

Specific Tab of **Honey ROP** „Generate Obj files” toggle.

Honey will use than *.obj files already stored on disk. This can save lots of time in case of bigger meshes. When tweaking lights/shaders you should definitely go this way.

Second step would be to turn off *.obj files parsing by Honey python script:

Specific Tab of **Honey ROP / Geometry** option menu:
„Include reference to existing geo files”

This way Honey won't parse *.obj files and Sunflow will be pointed to *.sc files as they exist on disk. Keep in mind that Honey won't check if files really exist.

5.11 Global Illumination

Honey comes with a light shader called **SunFlow GI** which gathers all GI methods of Sunflow. One tricky part is that some of them needs photons to work. Not all type of light source emits photons. Both **Sunflow SunSky** atmosphere and **Sunflow IBL** light (Image Based Lightning) shaders don't emit photons thus **Sunflow GL** shader won't work and Sunflow will rise an error.

Note: It is possible to use Ambient Occlusion on surface shader also. You can easily apply AO shader with Sunflow Inline shader. Just find it on Sunflow forum or dig into examples scenes.

5.12 There is limited set of image formats supported by Sunflow. Not even sure of details. Some of them are: **jpg, png, hdr, exr**. Sunflow GUI can work with *.png files only. Others formats have to be rendered directly on disk.

Notes:

(a) Sunflow uses two types of color notation, standard:

color 1 1 1

and nonlinear palette:

{ "sRGB nonlinear" 1.0 1.0 1.0 }

I omitted the second one for now. Is it needed?

(b) In SunFlow documentation it is strongly recommended to start Java with "-server" option. However this can't be done calling Java from Houdini's csh. That's why "-server" command is missing in Honey ROP setting (see very bottom of Specific Tab). I hope someone can solve this puzzle.

(c) You can use python script **Honey.py** standalone as soon as you have a „*.honey" file on disk. *.honey is a temporary bridge between hscript and python. It is used by Honey.py to create *.sc scene description for Sunflow.

```
> honey.py -f myscene.honey
```

will create Sunflow scene from a file.

```
-o refer | referNoOverride | onDemand
```

„-o" option specifies how to deal with geometry (see 5.10 above and Honey ROP Help card).

```
> honey.py -f myscene.honey -r
```

This command will execute original render command as was specified in Houdini. Whereas command like this (-c option):

```
> honey.py -f myscene.honey -c „java -Xmx1G -jar C:/sunflow/sunflow.jar -bucket 32 spiral -nogui -o myrenderfile.exr D:/SF/sunflowexample.sc"
```

will override render command from *.honey file and use specified with -c „..."

(d) There is a new Sunflow edition on horizon along with revolution called Houdini 9. Sunflow 0.7.3 will bring new scene file format, Houdini 9 even more. All that means we live in fluid reality and so Honey does...

That's it for now,
bon appetit!
[Symek@2007](#)



All comments / suggestion / wishes are strongly welcome!