Radiance + OpenDX

Radiance

Radiance + Unity3D

OpenDX + Octave + Radiance
Visualisation with Open DX

Experiences on the use of OpenDX in the Radiance workflow
Bank Facade Irradiance
Radiance Workshop 2008

UV Mapping / Unity3D
This is a 3D model with textures.

This is a UV map.
Complicate geometry...
Radiance Workshop 2014

Ambient files in OpenDX
lookamb -d model.amb > model.data
for each point:

Position, vector
Direction, vector
Lvl, scalar
Wt, scalar
Rad, scalar
Value, vector
Pos.Grad: vector
Dir.Grad: vector

data.amb
## RADIANCE

rpict -av 0 0 -aw 0 -ab 5 -aa 0.2 -ad 512 -as 128 -ar 64 model.oct
SOFTWARE= RADIANCE 3.7.2b patch release 28 Aug 2005 by G. Ward
FORMAT=sccl

| #  | Value 1   | Value 2   | Value 3   | Value 4   | Value 5   | Value 6   | Value 7   | Value 8   | Value 9   | Value 10  | Value 11  | Value 12  | Value 13  | Value 14  | Value 15  | Value 16  | Value 17  | Value 18  | Value 19  | Value 20  |
|----|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 0  | 1.44468920| -0.0003800| 1.143601  | 0.000000  | 1.000000  | -0.000000 | 4          | 0.062500  | 3.228661  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  |
| 1  | 2.842533  | 0.000565  | 3.603000  | 0.000000  | -0.000000 | -1.000000 | 0.062500   | 0.099496  | 3.228661  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  |
| 2  | 2.417429  | 0.491166  | 3.603000  | 0.000000  | -0.000000 | -1.000000 | 0.062500   | 1.031604  | 1.238456  | 1.238456  | 1.238456  | 1.238456  | 1.238456  | 1.238456  | 1.238456  | 1.238456  | 1.238456  | 1.238456  |
| 3  | 2.403000  | 0.900999  | 3.676275  | -1.000000 | 0.000000  | 0.000000  | 0.062500   | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  |
| 4  | 2.112890  | 1.859396  | 3.603000  | 0.000000  | -0.000000 | -1.000000 | 0.062500   | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  |
| 5  | 2.127900  | 1.859396  | 3.603000  | 0.000000  | -0.000000 | -1.000000 | 0.062500   | 3.228661  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  |
| 6  | 2.161944  | 1.714096  | 3.603000  | 0.000000  | -0.000000 | -1.000000 | 0.062500   | 3.228661  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  | 0.000000  |
| ... | ...       | ...       | ...       | ...       | ...       | ...       | ...       | ...       | ...       | ...       | ...       | ...       | ...       | ...       | ...       | ...       | ...       | ...       | ...       | ...       |
bringing all in OpenDX
bringing all in OpenDX
2014 Radiance Workshop

Summer Sale

Giulio Antonutto

Arup
Article #1

Script to create grids from rectangles
Prepare an obj file:
* containing materials
* with only rectangles
* with no orphan vertexes

Copy file `grigliata.m` into working folder

Open Octave

To use type: `grigliata('name of the file.obj', pass i in world units, pass j in world units)`

As example like this:
`grigliata('grids.obj',2,2)`

Note: All tested and working with Cinema4D obj files, not sure it is the same with Rhino…
Note: triangles will be ignored
Did you check if a rectangle?

Grid: 001; punti_i = 17; punti_j = 17; v_i = 0.000000 10.000000 0.000000; v_j = -10.000000 0.000000 0.000000; N = 0.00 0.00 1.00;
Grid: 002; punti_i = 17; punti_j = 17; v_i = 0.000000 10.000000 0.000000; v_j = -10.000000 0.000000 0.000000; N = 0.00 0.00 1.00;
Grid: 003; punti_i = 17; punti_j = 17; v_i = 0.000000 10.000000 0.000000; v_j = -10.000000 0.000000 0.000000; N = 0.00 0.00 1.00;
Grid: 004; punti_i = 17; punti_j = 17; v_i = 0.000000 10.000000 0.000000; v_j = -10.000000 0.000000 0.000000; N = 0.00 0.00 1.00;
Grid: 005; punti_i = 16; punti_j = 16; v_i = 0.000000 10.000000 0.000000; v_j = -10.000000 0.000000 0.000000; N = 0.00 0.00 1.00;
Grid: 006; punti_i = 16; punti_j = 16; v_i = 0.000000 10.000000 0.000000; v_j = -10.000000 0.000000 0.000000; N = 0.00 0.00 1.00;
Grid: 007; punti_i = 16; punti_j = 16; v_i = 0.000000 10.000000 0.000000; v_j = -10.000000 0.000000 0.000000; N = 0.00 0.00 1.00;
Grid: 008; punti_i = 16; punti_j = 16; v_i = 0.000000 10.000000 0.000000; v_j = -10.000000 0.000000 0.000000; N = 0.00 0.00 1.00;
Article #2

Script to convert obj files to OpenDX
Prepare an obj file:
  * containing materials
  * with no orphan vertexes

Copy file `obj2odx.m` into working folder

Open **Octave**

To use type: `obj2odx('name of the file.obj')`

As an example try like this: 
`obj2odx('model.obj')`

*Note: All tested and working with Cinema4D obj files, not sure it is the same with Rhino...*
Article #3
Script to plot polar diagrams out of radiance light sources
Go to working folder containing the file `flux_calc.m`

Copy at this location your radiance light sure (for example an `ies2rad` converted file)

Open **Octave**

To use type: `flux_calc('file_name.rad')`

*Note that the scale is in meters*

When the script outputs the light source flux, go to the dx folder and launch **OpenDX**.

Within **OpenDX** select the **Photometry** network.

Run it

You should get a nice chart and flux plotted
(Can export the photometry as `vrml` surface to check `replmarks`, etc).

As an example try:

`flux_calc('Mundial_1KW_V1.rad')`

*Note: All tested and working with Cinema4D obj files, not sure it is the same with Rhino…*
Luminous flux: 1406 lm (0 lm upward component, 1406 lm downward component)
Article #4

Script to use UV maps from obj files to create textures in radiance (radiance baking)
Concept:
* use a obj file to derive UV mapping
* use an octree file to bake based on UV mapping

Note: All tested and working with Cinema4D obj files, not sure it is the same with Rhino...
You need to:
* prepare an obj file with triangles, no orphan vertexes, UV mapping and materials

* prepare an octree of a scene with the light, colours and materials of your liking

* edit the `batch_tex` script to indicate name of files and materials

* run the `batch_tex` script in Octave

* This will generate images to map back in radiance or in the 3d software of your choice

Note: All tested and working with Cinema4D obj files, not sure it is the same with Rhino…
Usage:

Edit the `batch_tex` file
(indicating name of obj and oct files, resolution, etc)

You can run several obj files out of a single octree for baking different maps

Note: All tested and working with Cinema4D obj files, not sure it is the same with Rhino...
Done triangle: 5749
Starting triangle: 5750
Done triangle: 5750
Starting triangle: 5751
Done triangle: 5751
Starting triangle: 5752
Done triangle: 5752
Starting triangle: 5753
Done triangle: 5753
Starting triangle: 5754
Done triangle: 5754
Starting triangle: 5755
Done triangle: 5755
Starting triangle: 5756
Done triangle: 5756
Starting triangle: 5757
Done triangle: 5757
Starting triangle: 5758
Done triangle: 5758
Starting triangle: 5759
Done triangle: 5759
Starting triangle: 5760
Done triangle: 5760
Saving Grid for Calculation
mv: rename grid.all to tif/model.grid: No such file or directory

Now Rendering Grid on 2 Cores
rm: ill.tmp: No such file or directory
[1] 14553 14554
[2] 14555 14556
[2] Done
rtrace -w -h -ab 1 -oa 0 -ad 256 -as 128 -u -ds .1 -dj .6 model.oct < grid.2.tmp | rcalc -e "$1=$1+0.0001;$2=$2+0.0001;$3=$3+0.0001; "$1 $2 $3 > ii.2.tmp
[1] Done
rtrace -w -h -ab 1 -oa 0 -ad 256 -as 128 -u -ds .1 -dj .6 model.oct < grid.1.tmp | rcalc -e "$1=$1+0.0001;$2=$2+0.0001;$3=$3+0.0001; "$1 $2 $3 > ii.1.tmp

Loading back simulation results...
Reassembling Grid with zero values
Filtering texture
Working on RED channel
Working on GREEN channel
Working on BLUE channel
Creating image and finishing

>>
Bonus, how to put back textures in radiance:

How to:
* prepare a material file
* convert obj with obj2mesh and use material file
* load mesh into radiance scene with mesh primitive
* check with objview

Example of material file:

```plaintext
void colorpict baked
7
red green blue
baked.pic
.
1-A1*((Lu-A3)/A5-floor((Lu-A3)/A5))
1-A2*((Lv-A4)/A6-floor((Lv-A4)/A6))
0
6 1 1 0 0 1 1

baked glow material
0
0
4 1 1 1 0
```
Common known issues

* no triangulated polygons or other errors (such as no UV map or no material)

* there may be issues with normal listed on obj (rhino?)

* -rx 90 and Y=Z swap Cinema4D style

* image is swapped -h -v (pflip can help)

* some triangles are left black, this is because they are too small

* memory error, too many pixels! I used 4096 x 4096 max.

* all back! did you check normals? did you check the octree?

* takes a long time to process polygons, maybe is a good idea to save intermediate steps (requires coding)

* generally slow, maybe a volunteer to port it to c#?
Suggested uses

* can use UV to generate regular grids on fluid forms

* can use baking to bake geometry or small detail (for example the facade of a building)

* can use baking for animations in radiance and in other software

* can use baking for video games! such as in the Unity3d platform…
Article #5

Script to create a parametric potato into a radiance scene
Copy **genpotato** into a working directory, provide appropriate permissions.

Explore the following settings:

-x value [PF along x, for details contact Carsten Bauer]
-y value [PF along y, for details contact Carsten Bauer]
-z value [PF along z, for details contact Carsten Bauer]

-u value [u subdivisions used by gensurf for maximum accuracy]
-v value [v subdivisions used by gensurf for maximum accuracy]

or simply run defaults with
**genpotato**
or
**./genpotato**

*Note, never forget to add a potato to your renderings!*
Article #6

OpenDX map to visualise ambient files
Terms and Conditions
Use as you wish.

The only condition is to share your improvements to the code with the community.

“Take but give back.”

For warranty refers to warranty terms.
Warranty
There is NO warranty!
Thanks!