Accelerad Documentation

You'll want to read this!

Welcome to Accelerad!

Accelerad is a free suite of programs for lighting and daylighting analysis and visualization. The suite uses physically-based backward ray tracing algorithms inspired by Lawrence Berkeley National Laboratory's popular Radiance software suite by Greg Ward. These algorithms are accelerated up to twenty times faster using OptiX™, a ray tracing engine built for the graphics processor unit (GPU). The acceleration factor scales with the number of available GPUs and is expected to increase on new generations of hardware. In order to allow for smooth adoption among Radiance users and software developers, Accelerad maintains compatibility with Radiance scene and output file formats and uses a subset of Radiance's material modifiers and command-line arguments.

Getting Started

Before you begin

The Accelerad programs are intended to replace the standard versions that come with the default installation of Radiance. Before you install Accelerad:

1. Verify that you have a CUDA-enabled GPU with compute capability 2.0 to 6.1. A list of currently-supported GPUs is available at https://developer.nvidia.com/cuda-gpus. If you are unsure what GPU you have:
    a. On Windows, right click on *Computer* and select *Properties* > *Device Manager* > *Display adapters*.
    b. On Linux, use the command `lshw -C display` to list devices.
    c. On Mac, choose *About this Mac* from the Apple menu and click on the *Displays* tab.
2. Download and install Radiance if it is not already present. The latest version of Radiance for all platforms can be found at https://github.com/NREL/Radiance/releases.
3. Download and install the latest graphics driver for your GPU from NVIDIA. Drivers can be downloaded from http://www.nvidia.com/Download/index.aspx.
4. On Mac, you will also need to download and install the latest CUDA driver, available from http://www.nvidia.com/object/mac-driver-archive.html.

Confirm that Accelerad runs on your computer

To confirm that Accelerad is successfully installed on your computer, try running the included Windows .bat or Unix .sh files in the *demo* folder. Running *test_accelerad_rpict* should produce a high dynamic range image file named *test_rpict.hdr*. Running *test_accelerad_rtrace* should produce a tabular set of data in a file named *test_rtrace.txt*. Running *test_accelerad_rcontrib* should produce a tabular set of data in a file named *test_rcontrib.txt*.

On Linux and Mac, it will first be necessary to add the Accelerad *bin* folder to the PATH and LD_LIBRARY_PATH environment variables, and to add the Accelerad *lib* folder the the RAYPATH environment variable. On Windows, this will be done automatically by the installer. On Linux or Mac, add the following to the end of your *~/.profile* file and edit the paths as necessary if you choose a custom install location:

```
export PATH=/usr/local/accelerad/bin:$PATH
export RAYPATH=/usr/local/accelerad/lib:$RAYPATH
export LD_LIBRARY_PATH=/usr/local/accelerad/bin:$LD_LIBRARY_PATH
```

Use Accelerad in your workflow

This suite contains Accelerad versions of several Radiance programs:

The Accelerad program ...	**… replaces the Radiance program ...**
accelerad_rpict	*rpict*
accelerad_rtrace	*rtrace*
accelerad_rcontrib	*rcontrib*
accelerad_rfluxmtx	*rfluxmtx*
accelerad_genBSDF	*genBSDF*

For Windows, these programs are installed by default to *C:\Program Files\Accelerad\bin*. However, different Radiance tools have different expectations about where Radiance programs will be found. Depending on the tools you use, **you may need to move or rename some files**. You can run Accelerad in one of three ways:

Within an application: In your applications that one of the above Radiance programs, change the paths to refer to the corresponding Accelerad program.

Replace the original programs: If your application does not allow you to edit the Radiance application names:

1. Copy the contents of the Accelerad *bin* folder into your Radiance *bin* folder.
2. In the Radiance *bin* folder, rename the Accelerad programs with the names of the corresponding Radiance programs and replace the original program files. You may want to keep backup copies of the original Radiance files elsewhere.
3. Copy the .ptx files from the Accelerad *lib* folder to the Radiance *lib* folder.

From the command line: If you use Radiance from the command line, you can simply address your commands to the Accelerad programs instead of the corresponding Radiance programs. Be sure that the Accelerad *bin* and *lib* folder are found in your PATH and RAYPATH environment variables, respectively. On Linux, the Accelerad *bin* folder must also be found in the LD_LIBRARY_PATH environment variable.

Information and Help

The Accelerad homepage can be found at:
http://mit.edu/sustainabledesignlab/projects/Accelerad/

A forum for questions, bug reporting, and update announcements can be found at:
https://groups.google.com/forum/?hl=en#!forum/accelerad-users

Release Notes

Radiance versions

Accelerad is designed to mimic the behaviour of Radiance and has been tested in comparison with Radiance version 5.0.a.11. A number of other sources exist for the Radiance executables, including DIVA, DAYSIM, and IES VE. However, these versions may not be updated as frequently and have not been tested with Accelerad.

Missing files from RAYPATH

Accelerad locates certain files using Radiance's RAYPATH environment variable. The message "File <filename> not found in RAYPATH" indicates that the folder containing the required file has not been added to the RAYPATH.

In Windows, this error may occur when Accelerad is installed for all users but another Radiance installation has been performed for the current user. In this case, the current user's RAYPATH will override the system RAYPATH created for all users by the Accelerad installer.

To edit the RAYPATH in Windows, right click on *My Computer* > *Properties* > *Advanced system settings* > *Environment Variables*. The RAYPATH variable may be in either the user variables or system variables set, but should not be in both. On Linux and Mac, environment variables can be edited in the hidden *~/.profile* file. Make sure that the RAYPATH variable contains paths to both the Accelerad *lib* and Radiance *lib* folders. The exact appearance will vary depending on the Radiance installations present, but a typical RAYPTH value looks like:

- Windows
  `.;C:\Program Files\Accelerad\lib;C:\Program Files\Radiance\lib`
- Linux and Mac
  `.:/usr/local/accelerad/lib:/usr/local/radiance/lib`

Missing library files (.dll, .so, and .dylib)

If you do not have Visual Studio 2013 installed on your computer, you may see a message that "The program can't start because MSVCR120.dll is missing from your computer." If you see this error, visit http://www.microsoft.com/en-us/download/details.aspx?id=40784 to download an installer for the missing file.

Linux and Mac computers may complain of missing .so or .dylib files if the Accelerad *bin* folder has not been added to the LD_LIBRARY_PATH environment variable. The path can be added by editing the *~/.profile* file.

Graphics driver crashes

If you use a GPU that is not in Tesla Compute Cluster (TCC) mode with Windows, you may experience an unresponsive screen leading to timeout detection and recovery (TDR) while

running the software. This will be accompanied by a message saying “Display driver stopped responding and has recovered.”

By default, the Windows OS will end processes on the GPU (essentially rebooting it) after two seconds of unresponsiveness, which can happen when the GPU is processing a large amount of data. Possible remedies include:

- Use a smaller non-zero -t argument (1.5 suggested). After the number of seconds indicated by -t, Accelerad briefly stops scheduling work on the GPU and waits for previously scheduled work to finish, allowing the GPU to be more responsive to the OS.
- If more than one graphics card is available, set CUDA_VISIBLE_DEVICES to use only the GPUs that are not connected to monitors.
- Increase the TDR delay by adding a registry key. This is not recommended for novices.
- Test on a more powerful graphics card or a card which is not connected to a monitor. Tesla cards or other cards with TCC mode enabled are not affected by TDR.

Older graphics hardware

Accelerad is targeted toward newer graphics hardware with large numbers of compute cores; however, it is compatible with older graphics hardware with CUDA compute capability 2.0 or greater. On older hardware, Accelerad will not necessarily run faster than Radiance and may even run slower due to setup time on the GPU.

Disabling GPU acceleration

You can reproduce the normal behavior of Radiance on the CPU in Accelerad by providing the argument *-g 0* to any Accelerad program. When this argument is given, Accelerad will not use the GPU for any calculations, and normal Radiance algorithms will be used.

Distance to origin

Because ray intersection calculations on the GPU use single-precision floating point arithmetic, some impression is likely for geometry that is located far from the origin. This can lead to missing geometry or stack overflow errors, which appear as warning messages. Additionally, warnings will appear as red pixels in *accelerad_rpict*. Accelerad makes an effort to avoid these issues, but the result is that surfaces far from the origin may appear incorrectly. As a solution, scale the model in meters and position the model so that relevant geometry is less than 100 meters from the origin.

Unexpected high and low values

Accelerad flags computation errors with brightly-colored results. If you see results with much higher radiance values than expected, check the command-line output for error warnings. Most commonly, these indicate stack overflow and can be corrected by increasing the *-g* parameter.

Low or patchy radiance results may indicate insufficient ambient coverage. You may correct this by varying the ambient parameters, including the additional Accelerad command line arguments.

Irradiance caching in *accelerad_rtrace*

This release features an experimental version of parallel irradiance caching in *accelerad_rtrace*. This algorithm works well in outdoor environments, but is known to undersample diffuse lighting in many-bounce paths. To turn off irradiance caching in either *accelerad_rtrace* or *accelerad_rpict*, set the ambient accuracy *-aa* to zero.

BSDF Use

Accelerad includes *accelerad_rcontrib*, which can be used to create bidirectional scattering distribution functions (BSDFs) by *rtfluxmtx* or *genBSDF*. However, the current versions of *accelerad_rtrace*, *accelerad_rpict*, and *accelerad_rcontrib* do not take BSDF files as input. Three-phase method simulations using Accelerad produce accurate results because this simulation type does not cast rays through the BSDF material. However, five-phase method simulations may produce lower illuminance results in Accelerad because interreflection of the direct solar component within the BSDF material is not taken into account.

Command Line Arguments

In addition to the command line arguments typically used by *rpict*, *rtrace*, and *rcontrib*, Accelerad introduces optional new command line arguments required for the GPU implementation of certain algorithms. These new parameters are summarized below:

Argument	Description	Default
-g *size*	Hint to set the GPU thread stack size to *size* bytes. A value of zero will cause the Accelerad programs to revert to normal Radiance behavior without using the GPU.	4096
-al *stride*	Set the spacing between seed point pixels for ambient sampling to *stride* in rtrace only. A value of zero will cause all pixels to be considered. This option is ignored when the *-az* option is used.	0
-ag *N*	Set number of ambient divisions for final gather infill to *N*. When *-aa* is non-zero, *N* ambient samples will be taken at points not covered by the precomputed irradiance cache. A value of -1 will cause the value to be copied from *-ad*.	-1
-az *res*	Set the number of seeds points for ambient samples to take around the circumference of a sphere based at the view point to *res* in rtrace only. A value of zero will cause view-dependent seeding to be used instead. Thus, zero should not be used in combination with the *-S* option in which a view file changes the view direction between frames.	0
-ac *N*	Set the number of k-means clusters for ambient calculation to *N*.	4096
-an *N*	Set the maximum number of k-means iterations to *N*. Larger values can cause k-means calculation to take longer but will generate more accurate ambient results.	100
-at *thresh*	Set the k-means threshold to *thresh*. This is the fraction of seeds that must change cluster in order for k-means iteration to continue. Smaller values can cause k-means calculation to take longer but will generate more accurate ambient results.	0.05
-ax *wt*	Set the weighting factor for position in k-means error calculation to *wt*. Small values concentrate more ambient calculations around edges where ambient gradients are likely to be large.	1.0
-t *sec*	Set the time between progress reports to *sec*. Fractional values of *sec* are supported and support is added for rtrace. A value of zero turns automatic reporting off.	0.0

Updates

Since Version 0.5 beta

- Added Accelerad versions of *rcontrib, rfluxmtx*, and *genBSDF* programs.
- Added support for Russian roulette ray extinguishing with negative *-lr* arguments. Using Russian roulette may have a negative effect on speedup and require a larger *-g* argument.
- Added support for src_phi2 and src_phi4 to be used in brightdata calls with *source.cal*.
- Slight performance improvement by ignoring materials that are defined but not used.
- Fixed bug in shadows in antimatter regions.
- Fixed bug in ambient calculations at low angles.
- Fixed bug that allowed ray weights greater than one in irradiance calculations.

Since Version 0.4 beta

- The ray tracing engine is updated to OptiX™ 3.9.1 final release using CUDA 7.5, which provides support for the Pascal architecture.
- Includes bug fixes from Radiance version 5.0.a.11.
- Added support for antimatter, although first string argument must be void.
- Added support for ambient inclusion and exclusion (*-ae*, *-aE*, *-ai*, *-al*).
- Added support for *.cal* files created with GenCumulativeSky.
- Added support for Mark Stock's *utah.cal*.
- Fixed bug in coordinate mapping for data from .dat files.
- Fixed bug in rendering glass with nothing behind it.
- Fixed bug in rendering shadows of trans materials.
- Fixed bug in rpict images with odd numbers in dimensions.

Since Version 0.3 beta

- The ray tracing engine is updated to OptiX™ 3.8.0 final release using CUDA 7.0.
- Includes bug fixes from Radiance version 5.0 final release.
- Added support for ambient supersampling (*-as*) so long as *-aa* > 0.
- Added support for Monte Carlo random seeds (*-u*).
- Added support for motion blur in rpict (*-pm*).
- Added support for depth of field blur in rpict (*-pd*).
- Added *-ag* parameter for faster final gather step for debugging.
- Added support for cone, cup, cylinder, tube, and ring objects in .rad files.
- Added support for illum materials with non-opaque or void string arguments.
- Added support for *corr*, *boxcorr*, and *cylcorr* functions from *source.cal*.
- Improved handling of normals modified by texfunc objects.
- Brightdata from *source.cal* can now be applied as a material to sources.
- Ignored triangulation errors with a warning.
- Printed an extra line in rpict to handle a bug found in IES<VE>.

- Fixed bug in display of GPU global memory size.
- Added support for Linux and Mac.

Since Version 0.2 beta

- The ray tracing engine is updated to OptiX™ 3.8.0 beta.
- Includes bug fixes from Radiance version 5.0a.
- Major speed improvements for ambient calculations in rpict and rtrace.
- Added support for spheres and bubbles in .rad files.
- Added support for alias objects in .rad files.
- Function objects no longer need to be direct parents of the objects they modify, but still only one function object is supported per modified object.
- Fixed issues in creation of non-square images that caused fatal errors.
- Renamed *-am* argument to *-an* to avoid conflict with new Radiance version 5.0a parameter.
- The *-x* and *-y* arguments are no longer required for rtrace.
- Removed limit on *-ad*.
- Removed limit on the number of TCC-enabled graphics cards that can be used.
- Reduced default OptiX stack size hint *-g* to 4096 bytes due to smaller stack size requirement.
- Improved reporting of errors.
- Fixed incorrect reporting on number of rays cast by rpict when GPU is used.

Since Version 0.1 beta

- The ray tracing engine is updated to OptiX™ 3.7.0 beta 3, which provides support for CUDA 6.5 and SM 3.7 GPUs, including the Tesla K80.
- Users are no longer required to download and install CUDA.
- Includes bug fixes from Radiance version 4.3.a.2.
- Radiance mesh objects are supported.
- Changed new *-at* parameter to work independently of *-aa* and *-ar*.
- Fixed a bug that prevented caching of ambient values for certain specular parameters.
- Fixed a bug in Gaussian transmission for trans materials.
- Changed error messages and return values to match those of Radiance.

Since Version 0 beta

- The ray tracing engine is updated to OptiX™ 3.6.3, which provides improved support for Maxwell-based SM 5.2 GPUs, including the GeForce GTX 980.
- Includes bug fixes from Radiance version 4.2.2.
- Enabled an experimental algorithm for parallel irradiance caching in rtrace.
- Implemented -I and -Id arguments from rtrace.
- Improved handling of not-a-number (NaN) errors in both GPU and standard implementations of rpict.
- Fixed a bug in Gaussian transmission for trans materials.

- Added ability to assign geometric transformations to skies defined with skybright.cal or perezlum.cal.
- Fixed a bug that caused out-of-resources error (cuda error 7) on some machines.
- Fixed a bug that caused stack overflow errors in some cases for geometry located far from the origin.

Current Limitations

The table below lists currently supported parameters. Unsupported parameters will be ignored.

Category	Supported Parameters	Unsupported Parameters
Direct	-dj, -ds, -dv	-dt, -dc, -dr, -dp
Specular	-ss, -st	
Visibility	-bv, -i, -I	
Ambient	-av, -aw, -ab, -ar, -aa, -ad, -as, -af, -ae, -ai, -aE, -aI	-ap, -am
Medium		-me, -ma, -mg, -ms
Rays	-x, -y, -lr, -lw, -ld	
View (rpict only)	-vt, -vp, -vd, -vu, -vh, -vv, -vo, -va, -vs, -vl, -vf	
Pixel (rpict only)	-pa, -pj, -pm, -pd	-ps, -pt
Modifiers (rcontrib only)	-m, -M, -b, -bn, -f followed by klems.cal, tregenza.cal, or reinhart.cal	
Control	-e, -u, -w rpict only: -S, -o, -z, -t rtrace only: -h, -o followed by o, d, v, w, l rcontrib only: -h, -o, -c, -V	rpict only: -r, -ro rtrace only: -n, -o followed by V, W, L, c, p, n, N, s, m, M, t, T

The following Radiance primitive types are currently supported. Unsupported surface types, textures, and patterns will be ignored. Surfaces assigned unsupported materials will also be ignored. Alias types are supported for all categories.

Surfaces

- Polygon
- Sphere
- Bubble
- Cone
- Cup
- Cylinder
- Tube
- Ring
- Mesh
- Source

Materials

- Plastic
- Metal
- Trans
- Glass
- Light
- Illum
- Glow
- Spotlight
- Antimatter

Textures and Patterns

- Texfunc using *tmesh.cal*
- Brightfunc using *skybright.cal*, *perezlum.cal*, *utah.cal*, *isotrop_sky.cal*, or output from GenCumulativeSky
- Colorfunc using *utah.cal*
- Brightdata using *corr*, *flatcorr*, *boxcorr*, or *cylcorr* from *source.cal*