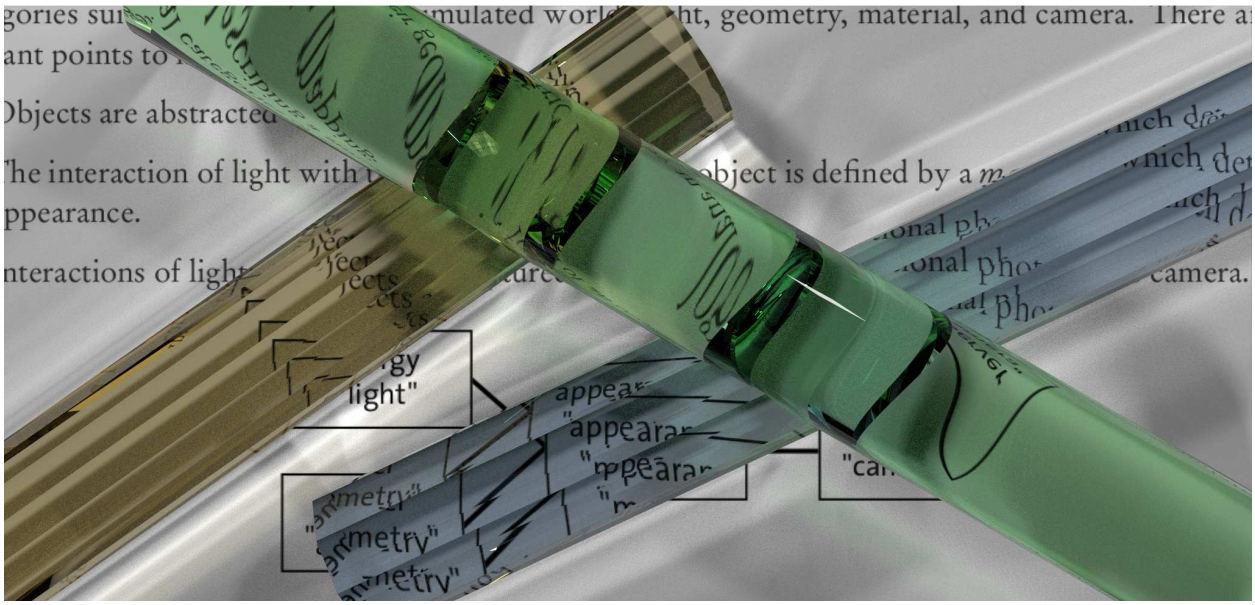
NVIDIA Iray

Whitepaper

Document version 1.0
6 December 2012



NVIDIA Advanced Rendering Center
Fasanenstraße 81
10629 Berlin
phone +49.30.315.99.70
fax +49.30.315.99.733
arc-office@nvidia.com

Document build number 185852

Contents

Iray — Summary of key benefits

NVIDIA Iray® 2013 **rendering technology** represents a comprehensive approach to state-of-the-art rendering **for design visualization**. Iray provides a **C++ API for seamless integration** of the renderer in applications running **on Mac OS X, Windows, and Linux**. No specialized programming skills in rendering technology and networking are required. **Detailed documentation, code examples, and an example application** in which Iray is integrated, are provided.

The following sections list key benefits that you can realize by integrating Iray into your 3D design visualization applications.

Rendering

Iray provides three rendering modes, which support a wide range of design needs, workflows, and content complexity:

- *Iray Photoreal* — Production-final rendering with full global-illumination support
- *Iray Interactive* — Interactive ray tracing and editing
- *Iray Realtime* — Large display capability and real-time editing (optional)

Because rendering modes share the same high-level scene description and the same materials, Iray is able to support seamless image blending when switching between modes. The result is a uniform user experience.

Scalability and networking

Iray provides distributed networking capabilities necessary to support high-performance rendering that can scale for massive scenes, multiple users, and remote, interactive collaboration. Key networking features include:

- Efficient rendering performance that scales for multiple GPU/CPU usage on a single host or multiple hosts
- Fault-tolerant clustering
- Cloud-ready design for rendering and collaboration

Materials

Iray provides an example material catalog implemented in NVIDIA Material Definition Language. Materials are easy to use and display a consistent appearance across all rendering modes.

Materials are created using a layered, modular concept. Material parameter values can be edited and materials themselves can be re-used to create new materials of varying complexity.

1 Executive overview

NVIDIA Iray® 2013 rendering software defines a comprehensive approach to state-of-the-art rendering for design visualization. This chapter presents the motivation for Iray, the Iray Unified Rendering Model, and a summary of key benefits.

1.1 Why Iray?

The NVIDIA Advanced Rendering Center (ARC) is the recognized leader in visual computing software, offering a wide range of rendering solutions for the design and entertainment industries. Our technologies are also used for geophysical visualization in the oil and gas industry. Our long term relationships with companies and user communities in these industries help NVIDIA ARC to better understand enterprise and user needs and how new technologies can help its customers maintain their competitive edge.

Iray is a response to the changing needs of companies where design is an integral part of their business and where "yet another renderer" is insufficient to satisfy the challenges they face. Iray provides a plugin architecture to integrate new technologial innovations and satisfy new customer demands, while providing the stability needed to protect existing technological investments.

Iray addresses four areas that can benefit from innovative, high-performance computing by providing:

- One advanced renderer to satisfy a wide spectrum of workflow and customer needs. This renderer generates convincing, physically-based visual simulations and supports interactive frame rates for design evalution and editing.
- A single method to describe surface and volume appearance that can support physically-based imagery.
- Scalable, network-based computing that enables more efficient use of the computing power of local networks and a simple interface to access Software as a Service (SaaS) offerings for cloud-based rendering.
- Support for collaborative workflows that enable imagery to be easily shared, discussed, and modified in real time.

Tight integration of these advanced visualization and compute technologies enables Iray to greatly accelerate professional design workflows, thereby reducing design cycle costs and time to market.

1.2 What you get with Iray

Iray addresses the major areas for improvement identified by NVIDIA. It provides one advanced renderer with multiple rendering modes to target a wide range of tasks, from real-time navigation through the scene to accurate preview of high-resolution imagery for print:

- *Iray Photoreal* — Production-final rendering with full global-illumination support
- *Iray Interactive* — Interactive ray tracing and editing
- *Iray Realtime* — Large display capability and real-time editing (optional)

To enable easy integration with existing design visualization software, as well as other applications that have rendering requirements for 3D content, Iray provides a heterogeneous set of tools, resources and instructions.

1.2.1 Tools

The C++ libraries provide the following APIs for integration and customization tasks:

- The C++ programmer's API provides a single access point to the library classes and methods used for integration and customization tasks. This API supports the import and customization of materials written in NVIDIA Material Definition Language (MDL).

- The plugin API allows programmers to extend the capabilities of Iray to suit application needs. Example plugins with limited capabilities are shipped with Iray as instructional aids. Software engineers may find it helpful to refer to these examples when developing plugins for their own applications. Example plugins shipped with Iray include importers and exporters for 3D content and plugins for cloud-based rendering.

1.2.2 Resources

The C++ libraries provide the following resources to support high-performance, interactive rendering over a network of hosts:

- The advanced renderer with multiple rendering modes. If you have special requirements, NVIDIA can work with you to develop additional rendering modes. (For more information, contact NVIDIA ARC at `arc-office@nvidia.com`.)
- NVIDIA Distributed Computing Environment (DiCE™), which provides:
  - Scene database creation and editing
  - Multi-threading of rendering processes
  - Scalable clustering using multiple GPUs for rendering on a local area network (LAN) or in the cloud
  - Node failure detection and data recovery based on redundancy

An example material catalog written in MDL is shipped with Iray. The materials are supported by all Iray rendering modes and can serve as examples for creating your own materials.

1.2.3 Instructions

Comprehensive documentation explains the concepts underlying Iray, its capabilities, and how to integrate it into design visualization applications:

- A Programmer's manual, which includes example programs
- A Technical Principles manual, which explains important concepts underlying Iray and the use and capabilities of the rendering modes
- An API reference, which provides detailed descriptions about the library classes and methods
- A specification for MDL
- A demo viewing application, which integrates a core feature set of Iray

1.3 Key benefits of Iray Unified Rendering Model

Iray is unified by a common in-memory representation of the scene and a common language for describing surface and volume materials. These features are key to the uniform user experience that you can offer users of your applications when they edit, manipulate, and model scene content and switch between rendering modes.

- *One renderer – multiple rendering modes*
  - Designed to make optimal use of GPU compute power
  - API support to provide a seamless user-interface experience for designers when switching between rendering modes
  - Tight integration with the DiCE to ensure consistent, optimized performance for rendering tasks
  - Consistent appearance for MDL-based materials across all rendering modes

- *NVIDIA Distributed Computing Environment (DiCE)*
  - A single in-memory representation of scenes shared by the rendering modes and by application users
  - A parallelization model and optimized networking components to scale for massive scenes and large numbers of users when using a cluster of hosts
  - Optimized for the most efficient use of GPUs for network rendering, either locally or in the cloud
- *NVIDIA Material Definition Language (MDL)*
  - A layered, modular concept to create materials:
    - Materials can be re-used to create new materials of varying complexity
    - Material parameter values can be edited
  - A set of generic material components and material classes, which serve as building blocks to create custom materials suited to your application requirements
  - An example catalog of surface, volume, and emissive materials:
    - Customizable
    - Shared and supported by all Iray rendering modes

Examples of rendered surface material definitions

Note: For more detailed information about MDL, see *Material Definition Language: Technical Introduction*. This document describes the key features of this language and how it compares to conventional shading languages.

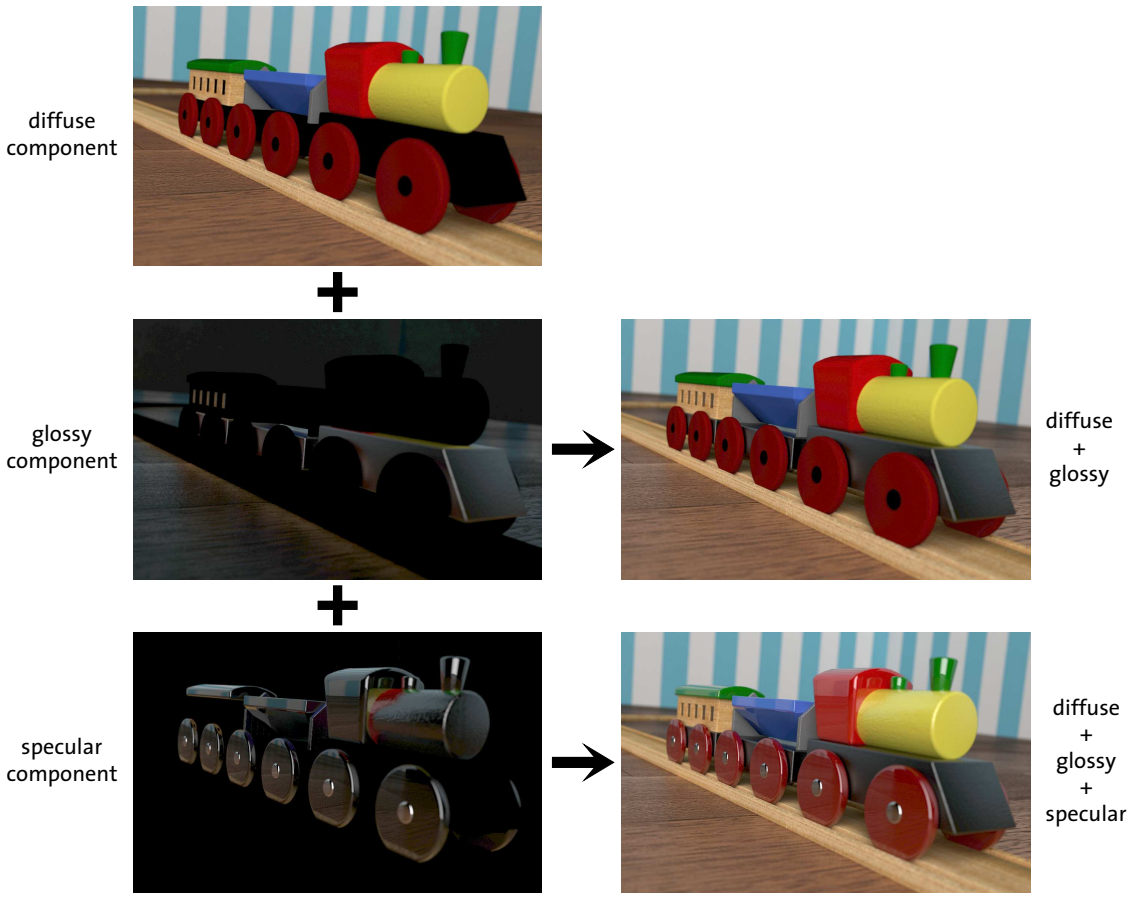
2 Unique capabilities of Iray

This chapter describes unique capabilities of Iray: How light path expressions can be used to easily isolate light components and store these components as output files at rendering time, how blending provides a uniform user experience across rendering modes, and how Iray support efficient use of GPU compute power on a single computer or a network of computers.

2.1 Using light path expressions

When a final image is rendered, lighting components can be isolated using a technique called "light path expressions" and saved to output buffers. Rendering of output buffers is done in parallel to the main rendering task. The buffers can be edited in post-processing using standard imaging or compositing software and the results can be integrated into the final output image without time-consuming and possibly expensive re-rendering.

The following image shows the diffuse, glossy (blurred reflection) and specular (perfect mirror refection) components of an image. The components are identified by light path expressions and are produced during a single rendering by Iray. When the corresponding pixels of these images are added together, the result is the complete rendering of all light interactions.

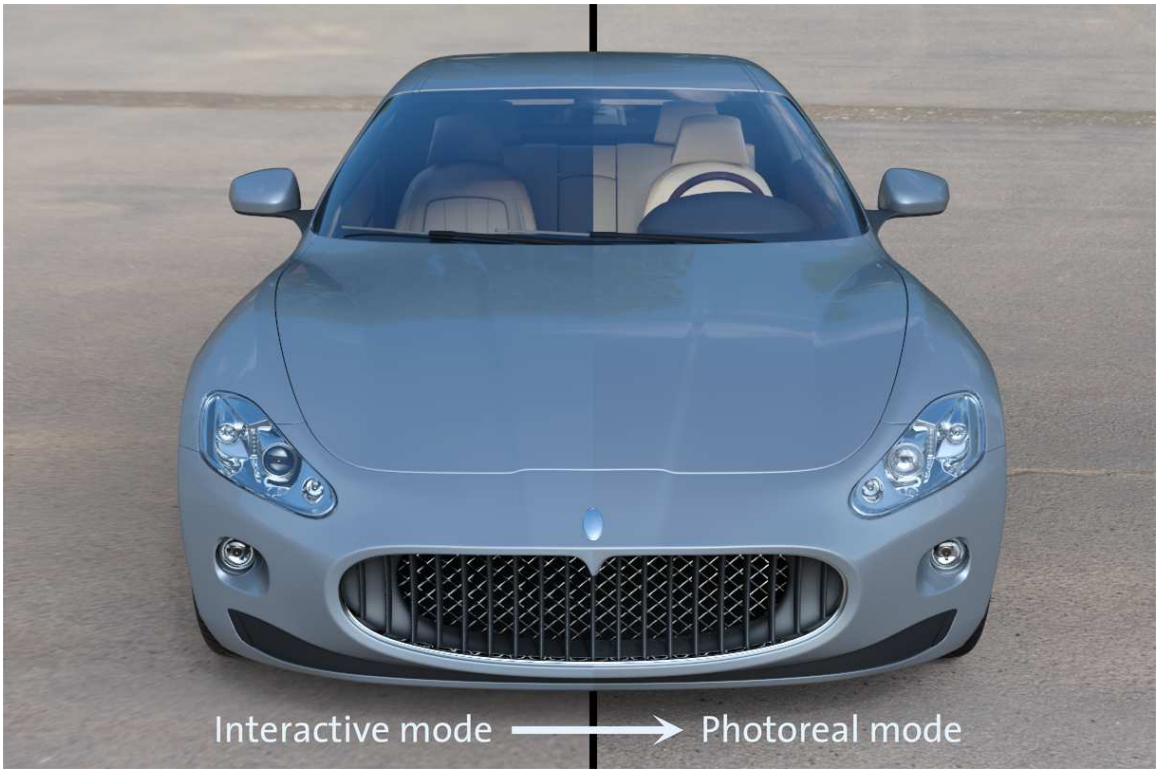


Adding the components produced by light path expressions

Light path expressions are a symbolic representation of the path that light takes between one or more light sources and the eye. Because the syntax is simple but expressive and precise, designers can more easily provide the specific light components required for post-processing tasks.

2.2 Enabling a uniform user experience across rendering modes

Iray provides a blend mechanism to enable a smooth transition between rendering modes in an application. The following figure illustrates one possible user-interface scenario: A designer explores a scene in Interactive rendering mode (shown in the left half of the figure). After a mouse-up event the entire image is blended to Photoreal rendering mode (right half of figure).

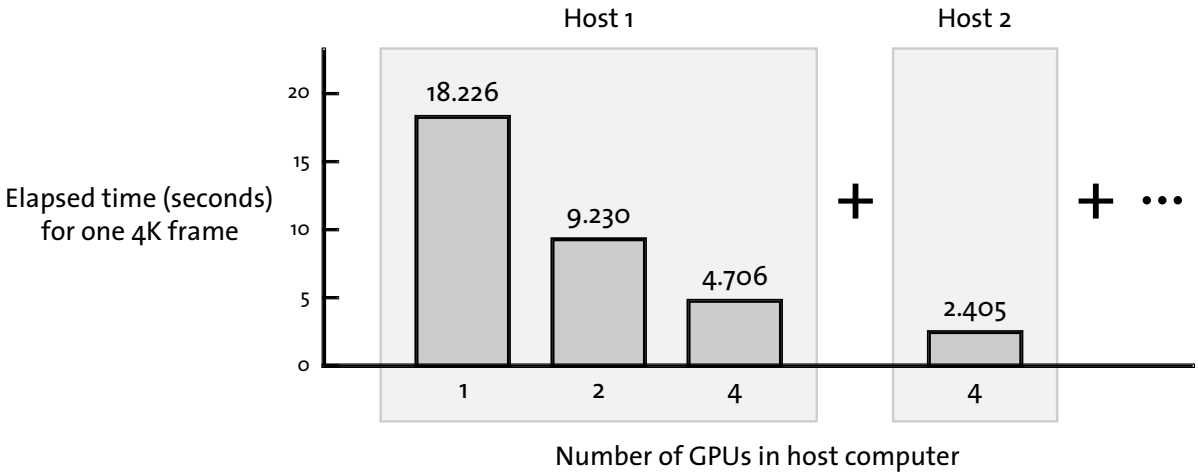


Using the same MDL materials across render modes to support a uniform user experience

Blending is possible because all rendering modes derive their appearance description from the same MDL materials. The MDL materials describe a complete model of appearance that each rendering mode implements based on its capabilities.

2.3 Harnessing GPU compute power

Iray is designed to efficiently maximize the massive parallel compute capabilities of NVIDIA GPU processors for a single computer or a network of computers. The following figure shows timing results for rendering one frame of a standard 4K resolution image.



Doubling the number of GPUs to generate a near-linear performance improvement

In a single computer, doubling the number of GPUs provides a near-linear performance improvement. Adding the processing of additional GPUs from a computer connected over a network also results in near-linear performance. As additional hosts are added, reasonable scalability across the network can be expected.

2.4 Contact us for a new perspective on rendering

Iray offers a new perspective on rendering. It is a comprehensive, physically-based, unified rendering solution. Multiple rendering modes use a common scene representation and a common material catalog. This means that Iray can seamlessly blend imagery when switching between modes and designers can remain confident that the appearance of objects remains consistent. Because Iray incorporates the latest advances in GPU and network technologies, it can support high-performance computing and facilitate collaboration with colleagues and customers in remote locations. Most importantly, the Iray C++ API assumes no special rendering knowledge and is designed to enable you to incorporate the Iray features suitable to your application needs.

To learn more about Iray and its unique capabilities, contact `arc-office@nvidia.com`.

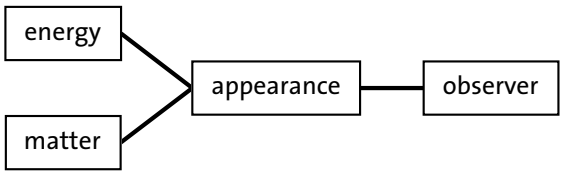
3 The conceptual process model underlying Iray

This chapter develops an intuitive model of the natural world of appearance and its mapping to categories suitable for a physical simulation model. It describes how Iray uses this simulation model to represent the data and processes required to implement:

- A flexible rendering system that can support a renderer characterized by multiple, special-purpose rendering modes
- Single user and collaborative workflows across local networks and the Internet

3.1 Modeling the world

At a fundamental level, the natural world may be modeled as the interaction of energy and matter. This interaction can, in turn, be viewed by an observer through the visible *appearance* of the world.

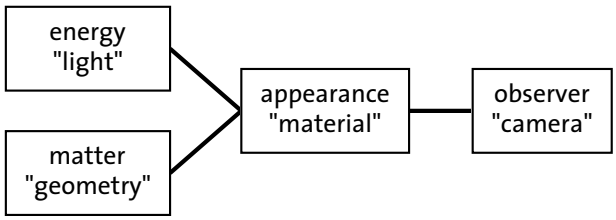


A conceptual process model of the natural world

In this model of the natural world, the appearance of an object is a consequence of the matter that constitutes it; in other words, appearance is an *intrintsic* quality of an object. However, to facilitate the mapping of intuitive categories of the natural world to a simulated world, a formal model can separate the appearance of an object from the object itself. Appearance is then an *extrinsic* category between the observer and the interaction of energy and matter. Defining appearance as an extrinsic property also has a very useful side-effect: the separation of the look of an object and its geometric structure facilitates an experimental freedom fundamental to the role that rendering plays in design and visualization tasks.

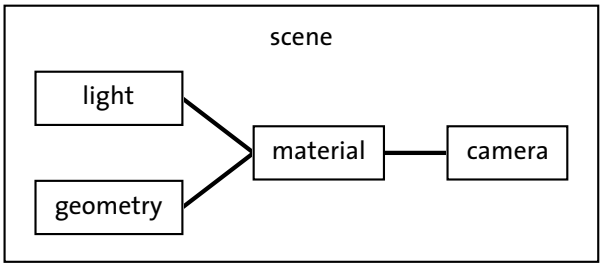
Iray maps the conceptual categories of the natural world model — energy, matter, appearance, and observer — to categories suitable for describing a simulated world: light, geometry, material, and camera. There are three important points to note in this mapping:

- Objects are abstracted to their geometric form.
- The interaction of light with the geometric form of an object is defined by a *material*, which determines appearance.
- Interactions of light and objects are captured by an analogy to a traditional photographic camera.



Mapping categories of a conceptual process model of the natural world to categories of a simulated world model

Lights, geometries, materials, and cameras in a simulated world and their spatial relationships constitute a *scene*. In Iray, the scene is the simulated world to be rendered.



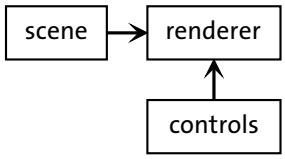
The scene as a model of the world to be rendered

3.2 Creating a record of the world

The virtual camera specified in a scene defines attributes analogous to those that characterize a real-world mechanical camera, for example, viewing direction and focal length. But, unlike a real-world camera, a virtual camera cannot create a record of a scene; this task falls to the renderer.

Strategies for creating a record of a rendering can differ between renderers. In ray tracing, for example, lines drawn from the camera's "eye" position into the scene determine the color values of the corresponding position in the resulting record. In other words, the camera determines the geometry of the relationship and the renderer controls how the color values are calculated.

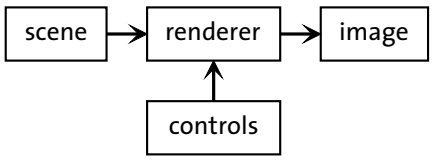
Control of the renderer is highly dependent on how the renderer is implemented in software. Developing an intuition for controls of the renderer requires an understanding of the rendering process itself.



The renderer and its controls are responsible for creating a record from scene data

3.3 Storing a record of the world

The data generated by the renderer can be stored in some manner, for example, as a file on disk. This data is often a picture of a scene, like a photograph of the real world. But a renderer can also collect other types of data such as surface position or orientation and structure this data to correspond to the scene from which it was derived. In this case, the "image" is a mapping from attributes of the scene to a two-dimensional position in a rectangular grid of data. Knowing what kind of data is contained in the image is necessary to correctly interpret the pixels of the rendered image.

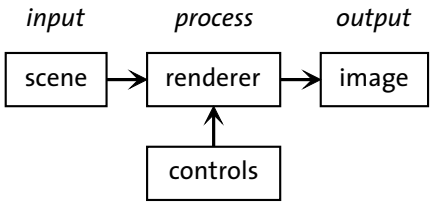


Images are records of the rendering operation

3.4 The rendering process

The preceding sections described the input, process, and output that map to the phases of the rendering process:

1. *Input* — The scene, which contains the description of the world and a position and direction from which it is viewed. The more comprehensive and detailed the description, the more realistically light interaction with surfaces and volumes can be modeled.
2. *Process* — The renderer, which processes the scene description. The algorithms the renderer implements determine its abilities to record and render the behavior of light and its interaction with objects.
3. *Output* — The output of the rendering process, which is an image — the record of what the camera "saw." The quality of the image is determined by the sophistication of the file storage format and the capabilities of the output device used to display the image.

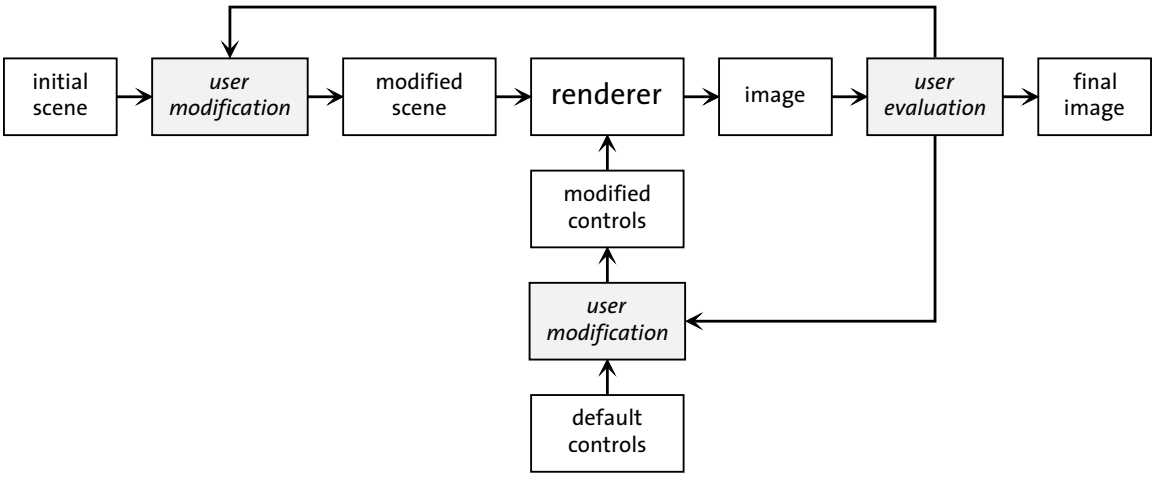


The fundamental phases of the rendering process

3.5 The rendering workflow

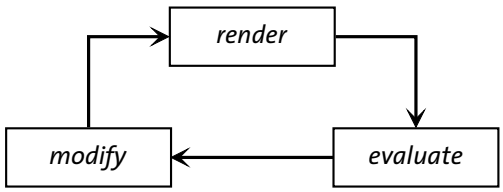
In practice, designers and other users rarely render a scene a single time, but rather explore the visual possibilities of a scene through repeated modification and subsequent rendering. The basic model of the rendering process in Iray can be augmented with moments of user interaction with the system:

- Modification of scene elements
- Modification of renderer controls
- Evaluation of the image as a basis for further modification



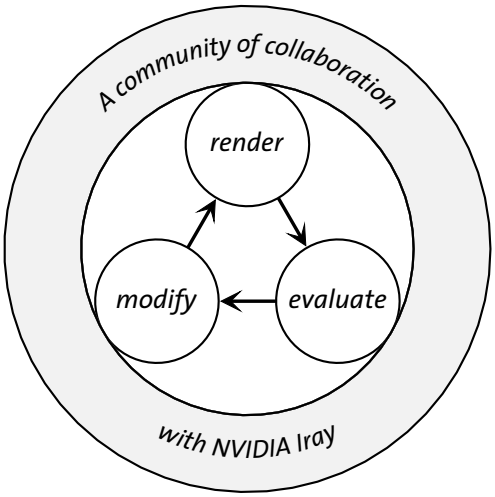
Successive evaluation, modification and re-rendering

At a higher level of abstraction, the rendering workflow for a single user can be reduced to a basic operations loop: render, evaluate, modify. This workflow loop clearly shows that rapid, physical simulations are a critical contributor to a designer's productivity.



The workflow loop for a single user

Many design and evaluation tasks are collaborative in nature. Iray supports workflow strategies that enable evaluation and modifications by large groups of artists, designers, and managers across local networks as well as the Internet. However, it is interesting to note that the underlying workflow remains the same as for the single user: render, evaluate, modify.



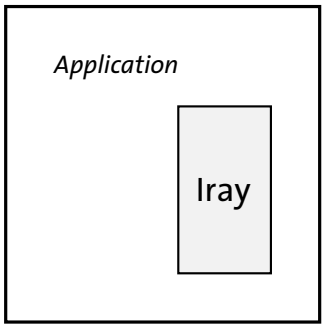
The workflow loop for collaborative tasks across local networks or the Internet

4 The implementation of the conceptual process model

This chapter describes the Iray implementation: the integration of Iray into an application, access to scenes and rendering modes through the Iray API, an example catalog of physically-based materials, parallel processing using GPUs and CPUs, local network and cloud rendering, and support for collaborative design and evaluation tasks.

4.1 Integrating Iray in your application

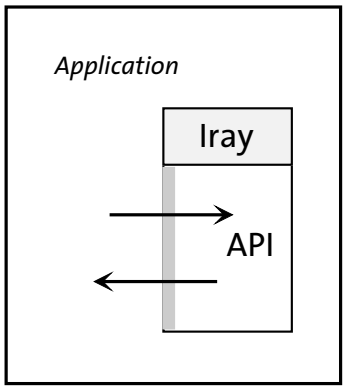
Iray is an implementation of the rendering process described in the previous chapter. Iray includes a set of C++ libraries that implement all the functionality needed to manage rendering in an application: configuration of your rendering hardware, support for loading, editing, manipulating, and modeling scenes and scene elements, rendering operations, and generating output. The C++ classes defined in the libraries support a wide range of application uses and configurations. For example, Iray can provide the rendering component in a desktop or web-based application or can be integrated as part of a dedicated rendering utility that services rendering calls from other applications.



Iray is a component of your application

4.2 Using the Iray API to access its class libraries

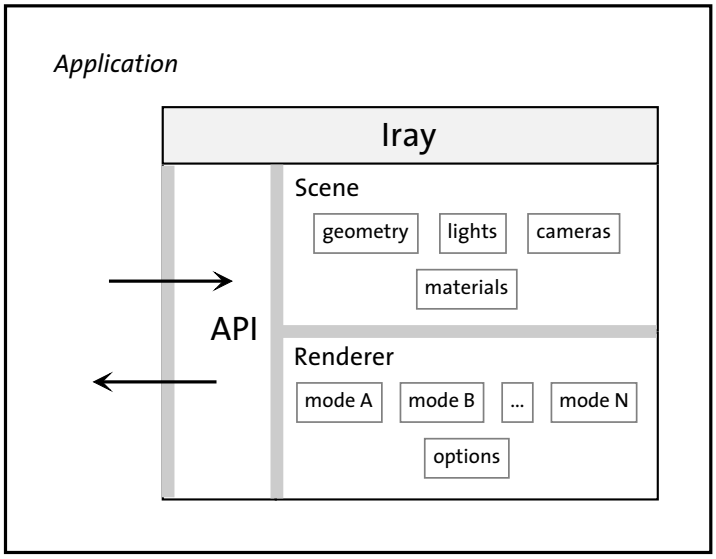
You access the classes defined in the libraries by using the Iray application programming interface (API). For example, when you want to provide functionality in your application to enable an end-user to load or edit a scene or render an image, you create instances of the appropriate classes available through the API and call the functions defined by those classes (their methods).



The application communicates with Iray through the Iray API

4.3 Accessing and modifying the scene and renderer components

The Iray library provides a wide range of classes to support the standard tasks that end users need to perform when working with the scene and the renderer. There are classes that support the modeling, editing and manipulation of traditional scene elements such as cameras, lights, and objects as well as rendering. Other classes support general process and programming tasks such as database transactions and memory management for data.



The elements of the scene and the renderer

4.3.1 Scene management operations

The Iray API for scene operations can be divided into three types: creation, editing, and storage.

- *Scene content* — Iray supports:
  - Scene content that is generated at runtime. The API provides class methods to create geometry, lights, cameras, and materials. Geometry construction uses a set of classes that implement geometric primitives (triangle meshes, polygon meshes, and subdivision surfaces) as well as various connectivity structures. Class instances are also used to construct the relationship of elements in the scene.
  - Scene content that already exists. Iray assumes that this content is stored in a file system or content management system and is loaded at runtime. To enable the addition of new scene file formats, scene file import is implemented through a plugin library mechanism. Loading plugins at runtime is defined though the API as part of the general mechanism for user-defined classes. Example plugin source files are included as part of the Iray software distribution.

  Content is stored in memory in a scene database for editing and rendering operations. The database may contain content for a specific scene, fragments of scenes, or a combination of both. The hierarchy of scene content in the scene database can be modified to suit the task at hand. For rendering operations, the renderer needs a root node, a camera, and the options.
- *Scene editing* — From your application, you can make API calls to edit content in the scene database. Common edit operations include:
  - Selecting one or more objects
  - Moving, scaling, or rotating objects

- Adding or removing objects
- Changing the shape of objects
- Changing the visual properties of surfaces and volumes

Iray provides mechanisms so that you can allow multiple users to access and edit the same content. You can also allow users to hide their changes from or share their changes with other users.

- *Scene storage* — Iray provides an export mechanism to save in-memory scene data to file. Exporting scene data can be implemented as a library plugin in a symmetric manner to the importer plugin mechanism.

4.3.2 Lights and lighting

Iray supports the following types of lighting:

- Light elements in the scene database
  - Directional lights
  - Point lights, including spot lights
  - Area lights: cylinders, spheres, discs, and rectangles
- Image-based lighting
  - Environment maps
  - Procedural definition of environmental illumination values
- Geometry that has emissive properties defined by a material

All sources of light in the scene are used in a material's definition of light interaction with surfaces and volumes.

4.3.3 Rendering mode selection

Rendering mode selection is implemented as part of the API. The following rendering modes are supported:

- *Iray Photoreal* — Production-final rendering with full global illumination support
- *Iray Interactive* — Interactive ray-tracing and editing
- *Iray Realtime* — Large display capability and real-time editing (optional)

The rendering modes produce varying degrees of physically-based and photorealistic imagery with different performance characteristics. Each rendering mode is designed to take full advantage of the latest advances in GPU technologies with a CPU-based fallback where appropriate. Because all rendering modes operate on scene data stored in memory in the scene database, your applications can offer a seamless user experience when switching between rendering modes.

4.3.4 Appearance definition through materials

Materials define the visual properties of objects, for example:

- The color, reflection and refraction, and light emission properties of surfaces
- The scattering and absorption properties of volumes
- Modified geometric properties of surfaces such as displacement and normal perturbation (bump mapping)

Iray provides an example catalog of physically-based materials to apply to geometry, lights, and cameras. These materials are defined using NVIDIA Material Definition Language (MDL). Materials are defined based on distribution functions for reflection and refraction, for volume scattering effects, and for light emission properties.

Material definitions define "what to compute". The "how to compute" task is considered the domain of the renderer. This split of responsibilities enables all Iray rendering modes to use the same materials and is key to the Iray Unified Rendering Model.

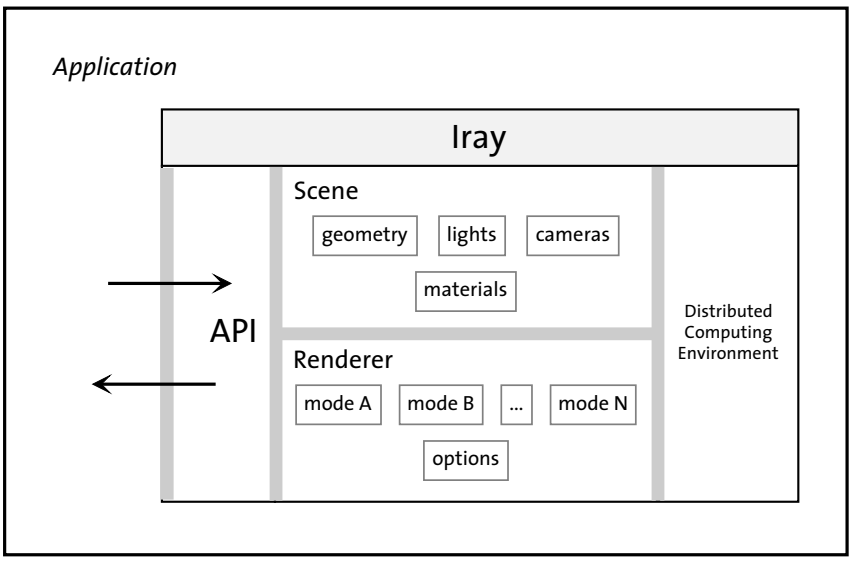
4.3.5 Runtime settings for the renderer

Runtime settings can be set in the scene description and in the renderer itself:

- The scene description may contain additional information about how the camera, lights, and other objects should be used in the scene beyond their simple geometric definitions. This "meta" information is defined in an options element that is also part of the scene. The values for these options provide parameter values for the various operations of the renderer, for example, how the geometry of the scene should be optimized for efficient rendering.
- Additional settings may be defined for the renderer itself, independent of a particular scene. Typically, settings are used to inform the renderer about where to find images and other file-based resources, whether to use GPUs or CPUs for rendering, and if GPU rendering is used, which hosts and GPUs should then be allocated for rendering.

4.4 Parallel computing on a distributed system

Iray uses NVIDIA Distributed Computing Environment (DiCE) technology to enable high-performance rendering and efficient data store distribution over a large cluster, multi-user operation, data redundancy, and dynamic cluster configuration.



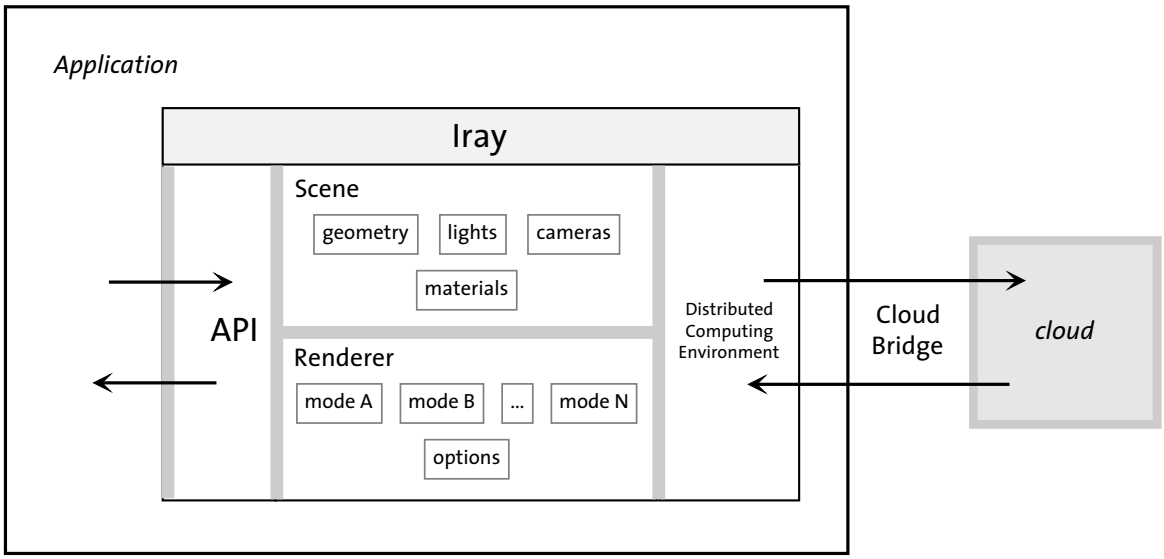
Computing on multiple hosts

- *Scalability* — DiCE distributes rendering over multiple GPUs on a single machine or in a cluster. DiCE uses a parallelization model and optimized networking components to enable effective use of large clusters of hosts even for interactive rendering modes. The result is faster, more efficient rendering and a better user experience.
- *Cluster rendering of large scenes* — DiCE includes a distributed, in-memory data store, which is used to store the complete representation of the scene and to efficiently distribute the scene data to all hosts in a cluster. This eliminates the long startup times typically needed to distribute data for massive scenes to a cluster.

- *Multi-user operation* — DiCE is specifically designed for multi-user operation, which enables Iray to be used by multiple users who share a single host or a cluster. Users working on the same scene can share the same scene data. DiCE can store variants of data objects, which allows users to use different versions of one scene. All parts of the scene which are the same for multiple users are shared between them, which can dramatically reduce memory requirements.
- *Dynamic cluster size* — The data store can be configured to provide redundant storage when Iray is running on a cluster, If one or more hosts in a cluster fail, for example due to hardware failure, Iray can continue operation. This feature is important for interactive applications which run for long times or are mission critical. In addition. hosts can be dynamically added to a cluster while Iray is running, which enables compute capacity to be increased without the need to restart the software.

4.5 Using the cloud

Iray uses Cloud Bridge to manage rendering with servers accessed through the Internet — in the "cloud".



Using Cloud Bridge to access rendering resources from the Internet

With Cloud Bridge, your application can push a local scene database to a remote server over a relatively low-bandwidth, high-latency connection such as the Internet.

Cloud Bridge uses caching and automatic change detection to efficiently transmit data not already on the remote server. Caching is done both in memory and on disk. When switching between scenes, or when restarting clients or the server, the cached data on the server is still available. For a server-side scene that has been rendered previously and is still cached, only minimal amounts of data need to be pushed to the server.

When Cloud Bridge is activated, Iray rendering modes are made available. For cloud rendering:

- No local resources are required for rendering
- All necessary database changes are automatically transmitted to the remote server
- Rendered images are automatically streamed to the application

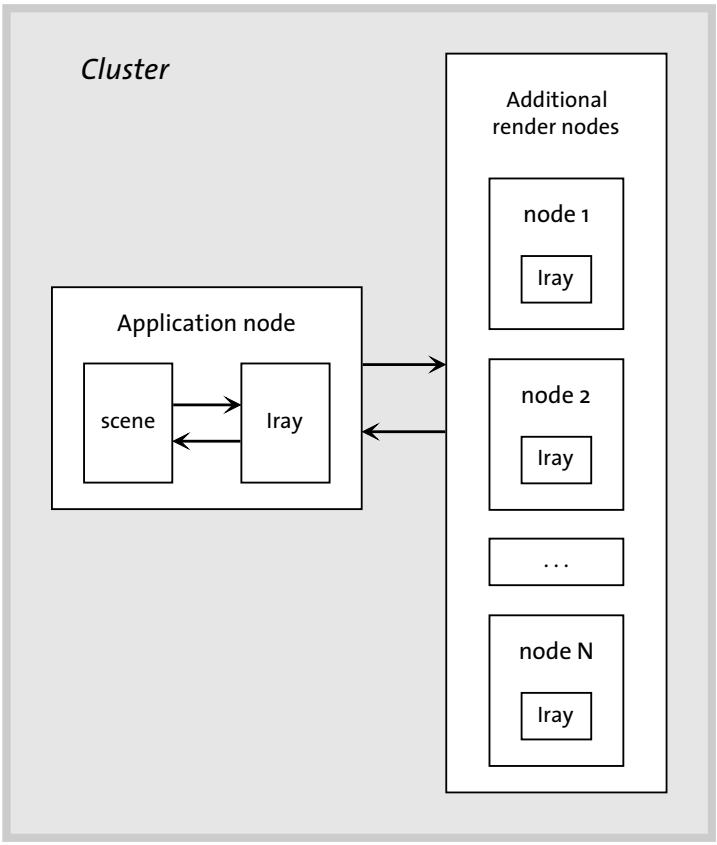
5 Cluster-based rendering with Iray

As previously described, distributed networking features and parallelization techniques built into Iray support efficient and scalable rendering using a set of processing units called a *cluster*. This chapter describes example configurations for local and remote cluster-based rendering.

5.1 Cluster-based rendering on a local area network (LAN)

A LAN supports high-bandwidth, low-latency network connections, which is ideal for interactive and realtime rendering.

The following figure illustrates a possible LAN configuration. Note that Iray must be installed on every node in the cluster that will be used for rendering.



Creating an Iray-based configuration for cluster-based rendering on a LAN

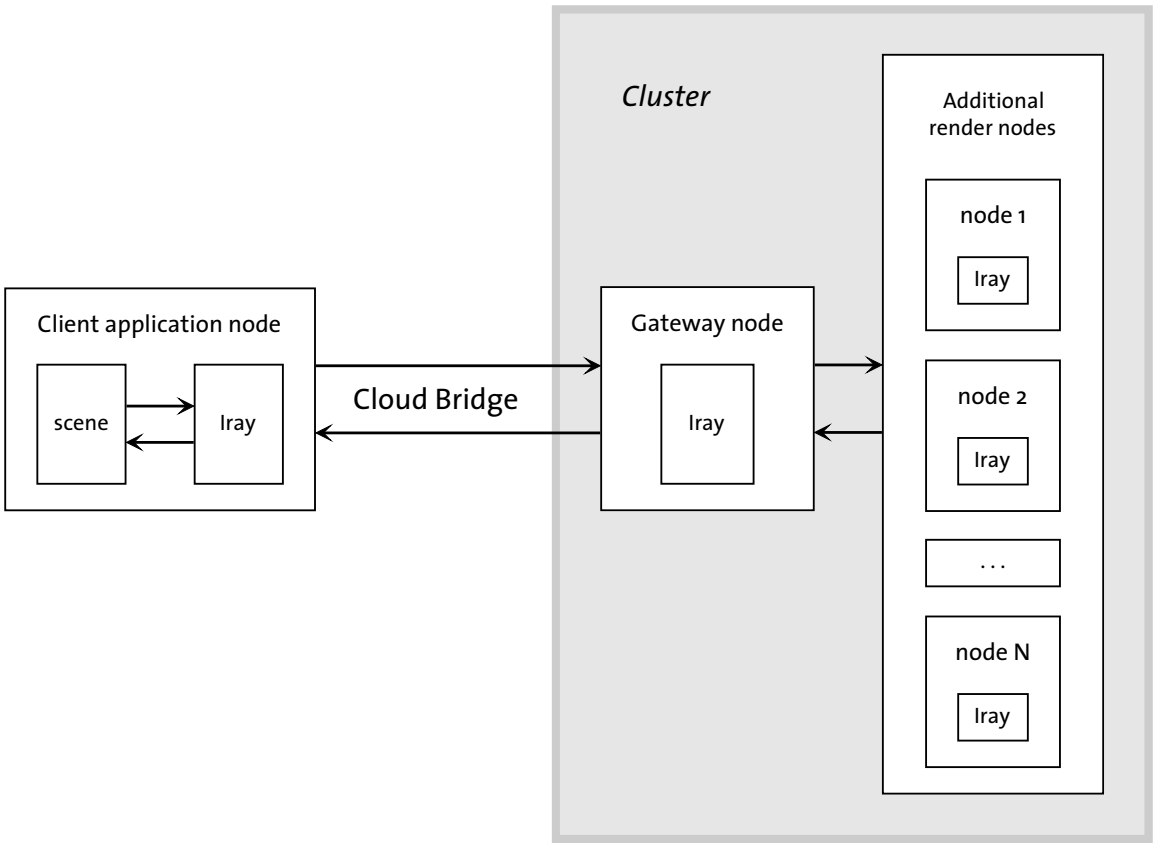
A cluster of rendering nodes consists of an application node and additional rendering nodes. The application node also functions as rendering node.

At runtime, a copy of the scene created or loaded in the application is stored in the Iray scene database on each rendering node in the cluster. Incremental changes to a scene database on one rendering node are automatically shared with all rendering nodes. The Iray API provides C++ classes to enable the original scene to be updated.

5.2 Cluster-based rendering in the cloud for Software as a Service (SaaS)

Cloud Bridge supports relatively low-bandwidth, high-latency connections. It is ideal for off-loading compute-intensive rendering tasks to a 3D visualization service on the Internet or in the cloud.

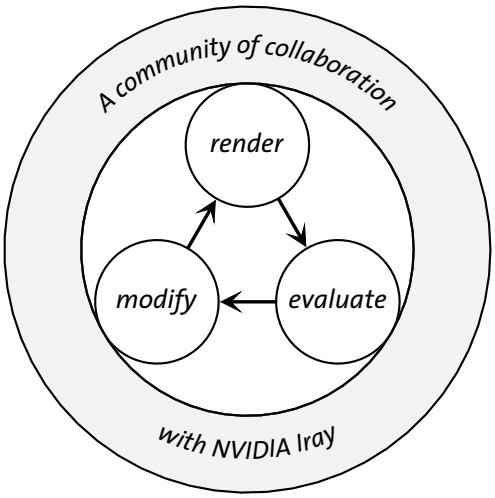
The following figure illustrates a possible configuration for rendering on a remote cluster. Iray must be installed on the client node and every node in the remote cluster that is used for rendering.



Creating an Iray-based configuration for remote, cluster-based rendering in the cloud

The application node is a client of the rendering service that is installed on the remote cluster. Cloud Bridge, described in the previous chapter, is used to enable communication between the client and the remote cluster.

At runtime, a copy of the scene is created or loaded into the client application. This scene is then stored in the Iray scene database on the client and on each rendering node in the remote cluster. Incremental changes made to the scene database on the client are sent to the remote cluster where they are automatically shared with all rendering nodes. The Iray API provides C++ classes to enable the original scene on the client to be updated.


A community of collaboration
render
modify
evaluate
with NVIDIA Iray