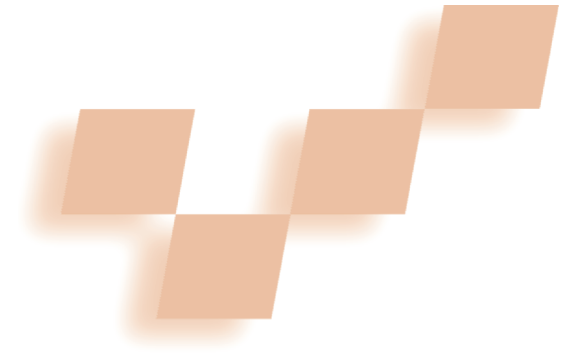


# A Distributed Rendering Environment for Teaching Animation and Scientific Visualization



Krishna P.C. Madhavan, Laura L. Arns, and Gary R. Bertoline  
*Purdue University*

**D**istributed rendering environments (DREs) are fairly common in the computer graphics and animation industries. They help create large-scale animations ranging from movies like *Shrek* to advanced scientific visualizations. Most DREs are deployed in tightly controlled animation production units or national labs where access is limited to a small group of people. While DREs are well known in the industry, no implementations are currently available for general student access. Furthermore, the challenges of setting up such a system for a large audience of students at colleges and universities are significant. Given this scenario, computer graphics students today have no exposure to real-world problem-solving techniques in rendering and animation development, as that would require access to an infrastructure that is absent in academia.

The course assignments given to students in rendering and animation programs do not reflect the actual magnitude and scale at which commercial and scientific computer graphics work is undertaken. Instructors are often limited to the computational resources available within the physical confines of the learning space. In effect, this leads to the complete dilution of the content taught within the rendering and animation curricula. The students graduating from such programs are, therefore, under-

prepared to deal with the complexities of the work environment they enter.

This article outlines the development of a DRE at Purdue University. Faculty and students are currently beta testing the DRE. The project's goal is to make DREs available to faculty and students throughout the US at no cost. The design addresses several pedagogical paradigms that are critical for enhanced positive learning outcomes (see the "Background and Rationale" sidebar). Critical to the development of Purdue's DRE is the notion of an enhanced positive user experience. To address this goal, the project team is focusing on engineering methods for one-touch job deployment; ease of job status monitoring; anywhere, anytime job submission; rendering and animation error correction; and reduced job wait times.

## Harvesting spare compute cycles

As rendering algorithms are continuously refined to maximize accuracy and clarity of animations and scientific visualizations, the computational resources required to achieve this objective have also increased exponentially. It's impractical for educational institutions to invest valuable dollars in infrastructure that can only be used for rendering. However, most universities support and maintain large computer laboratories that remain unused for a small part of each day. Our approach to designing and deploying DREs is based on harvesting and using processor cycles effectively during the brief periods when computer labs are closed. Our system splits one large animation into individual frames and distributes it to any available computer—we refer to this as a *rendering node*. As more computers become available,

---

This article outlines the development of a distributed rendering environment at Purdue University. The design addresses several pedagogical paradigms critical for enhanced positive learning outcomes.

## Background and Rationale

The pedagogical problems with current approaches to classroom instruction in animation and scientific visualization are rooted in the logistics associated with rendering. Rendering has long been an intrinsic part of traditional computer graphics instruction. However, students have always relied on single machines to complete rendering assignments. Rendering jobs on single machines is extremely slow and time consuming.

For example, let's examine a simplified scenario where students are asked to produce a 2-minute animation project. Total rendering time for this project is

$$t_{rt} = \text{fps} \times t_{al} \times t_{rf}$$

where  $t_{rt}$  is the total rendering time in seconds,  $\text{fps}$  is frames per second,  $t_{al}$  is the total length of animation (in seconds), and  $t_{rf}$  is the time for rendering one frame (in seconds).

If we assume that the animation runs at 30 fps and each frame takes approximately 2 minutes to render, the overall rendering time is 7,200 minutes or 120 hours. In real-world scenarios, a single frame could take substantially longer to render. Extending this example further, a single class of just 20 students attempting to complete a 2-minute animation assignment would tie up computers in a departmental laboratory for a total of 2,400 hours.

Because rendering is a processor-intensive operation, the computer on which the rendering job is running becomes unavailable for other activities. Interruptions might cause the student to restart the entire job, resulting in wasted resources and loss of time. In an educational setting where multiple courses might require rendering assignments and the use of computers for formal course lab sessions, the magnitude of this problem is immediately apparent. The situation is further complicated by the need for custom configurations on machines that usually perform rendering.

Due to this heavy dependence on resources completely local to the student's work environment and learning space, current pedagogical practices in the field of animation and data visualization focus on restricting the scope of rendering assignments. This has resulted in a situation where instructors consciously dilute the scope of the problem assigned within the classroom, thus leading to less than authentic learning experiences. Also, faculty members teaching animation have become accustomed to lower

qualities of class projects due to the logistical difficulties associated with getting the renderings correct.

The practice of diluting problems to suit the learning space runs counter to existing pedagogical paradigms that call for constructivist approaches to teaching and learning. Under the constructivist model, learners are encouraged to apply their knowledge to facilitate deep learning.<sup>1,2</sup> Kauchak and Eggen point out that "learning activities based on constructivism put learners in the context of what they already know, and apply their understanding to authentic situations."<sup>3</sup> In the context of learning and teaching animation and scientific visualization, the emphasis of the learning process should be on authenticity. Under the current model, learners do not have an opportunity to apply their knowledge in authentic, real-world scenarios. Resnick points out the distinct difference between knowing a certain concept and a student's ability to apply this knowledge.<sup>4</sup> The necessity for the learning process to incorporate tenets of the real world has well-established theoretical foundations.<sup>5,6</sup> The use of authentic learning contexts and scenarios is not only necessary to cultivate the learner's ability to apply the knowledge learned in class to a variety of contexts,<sup>7</sup> but also to create cognitive flexibility<sup>8</sup> that is critical to real-world problem solving. In the context of animation and data visualization teaching and learning, one area where the application of learned knowledge becomes critical is in error detection.

Several complications can arise when rendering animations during students' initial learning stages. It's difficult to complete a rendering accurately without doing several tests first. For example, in the movie industry it's not unusual to render a frame more than 50 times to get the desired results.

Shadows, lights, animation, and textures are just a few elements that could create undesirable results in the rendering process. For example, shadow errors are difficult to diagnose before rendering a full animation. A typical shadow error could be manifested as flickering, where the shadow associated with an object may have variability from frame to frame.

When working with lights and shadows, students need to test render at various stages of the animation. This will help ensure that all parts of the scene are lit and shaded correctly. Each time a lighting attribute is changed, a new render must be done to see the outcome, often resulting in several re-

*continued on p. 34*

the system creates a farm of rendering nodes called a render farm. The time to complete an animation is inversely proportional to the number of render nodes:

$$t_{rt} = \left( \frac{\text{fps} \times t_{al} \times t_{rf}}{n_r} \right) + E_{to} \quad (1)$$

where  $n_r$  is the number of render nodes and  $E_{to}$  is the extra time in rendering due to overheads.

Equation 1 is only an estimate based on an upper-bound approximation of the speedup with rendering time. This value is the theoretical speedup in rendering.

We can determine the observed or actual speedup by considering Amdahl's<sup>1</sup> and Gustafson's laws.<sup>2</sup>

In most types of rendering, each frame can be viewed as completely independent of the other. However, in some scenarios, where complex dynamics are computed, one frame might depend on ones previously rendered. This reduces speedup of the overall render job. To alleviate this problem, students usually must "bake" the dynamics component of their jobs prior to submission for rendering. Baking the dynamics first creates a key for every frame where the dynamic simulation is set to active. This removes the need for dynamic calculations and dependency upon previous

continued from p. 33

renders. This isn't always done under student workflow because students do not yet fully understand the process.

There is a workaround to doing full renders all of the time, as many rendering packages can generate a crude preview render that lets the user quickly see changes made with sights and materials. However, these previews do not always let the user see all of the potential problems with the animation. While several parameters might affect the rendering's accuracy, unless students are exposed to these scenarios as part of their learning experience, they won't have the cognitive flexibility needed to detect rendering errors in animations.

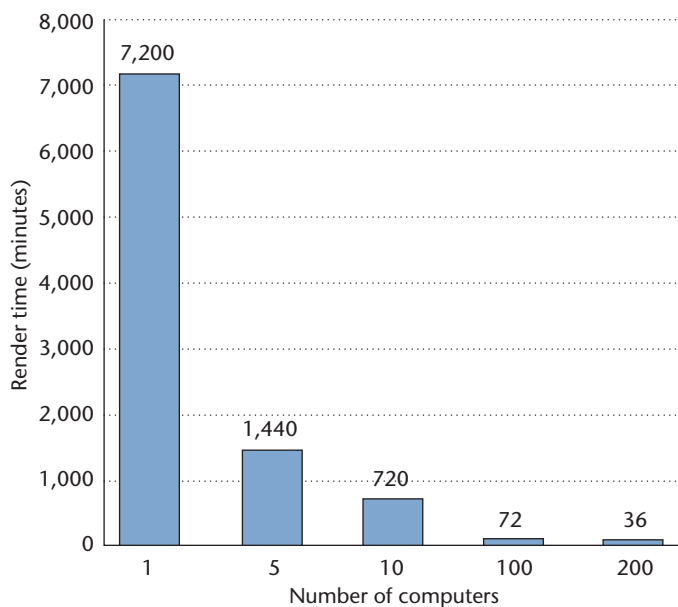
Students need full-scale, hands-on experience in dealing with the animation process, particularly given the large potential for errors in rendering. However, the logistical problems with the rendering process preclude any such educational experience for students.

The limited computational resources available for rendering within traditional learning spaces has forced faculty members and students to resort to pedagogical techniques that are far removed from real-world practice. To alleviate much of the resulting logistical and pedagogical problems, we need to extend the virtually unlimited computational power available through newly emerging

distributed computing paradigms into the classrooms. It is in this context that DREs, such as the one described in this article, can have significant impact.

References

1. J. Bruner, *The Process of Education*, Harvard Univ. Press, 1960.
2. J. Bruner, *Going Beyond the Information Given*, Norton, 1973.
3. D.P. Kauchak and P.D. Eggen, *Learning and Teaching: Researched-Based Methods*, Allyn and Bacon, 1998, p. 184.
4. L. Resnick, "Learning in School and Out," *Educational Researcher*, vol. 16, no. 9, 1987, pp. 13-20.
5. J.S. Brown, A. Collins, and S. Duguid, "Situated Cognition and the Culture of Learning," *Educational Researcher*, vol. 18, no. 1, 1989, pp. 32-42.
6. Cognition and Technology Group at Vanderbilt, "Anchored Instruction and Its Relationship to Situated Cognition," *Educational Researcher*, vol. 19, no. 6, 1990, pp. 2-10.
7. M.L. Gick and K.J. Holyoak, "Schema Induction and Analogical Transfer," *Cognitive Psychology*, vol. 12, no. 3, 1983, pp. 306-355.
8. R.J. Spiro et al., "Cognitive Flexibility, Constructivism, and Hyper-text: Random Access Instruction for Advanced Knowledge Acquisition in Ill-Structured Domains," *Educational Technology*, vol. 11, no. 5, 1991, pp. 24-33.



1 Theoretical reduction in rendering time due to adding render nodes to a 2-minute animation.

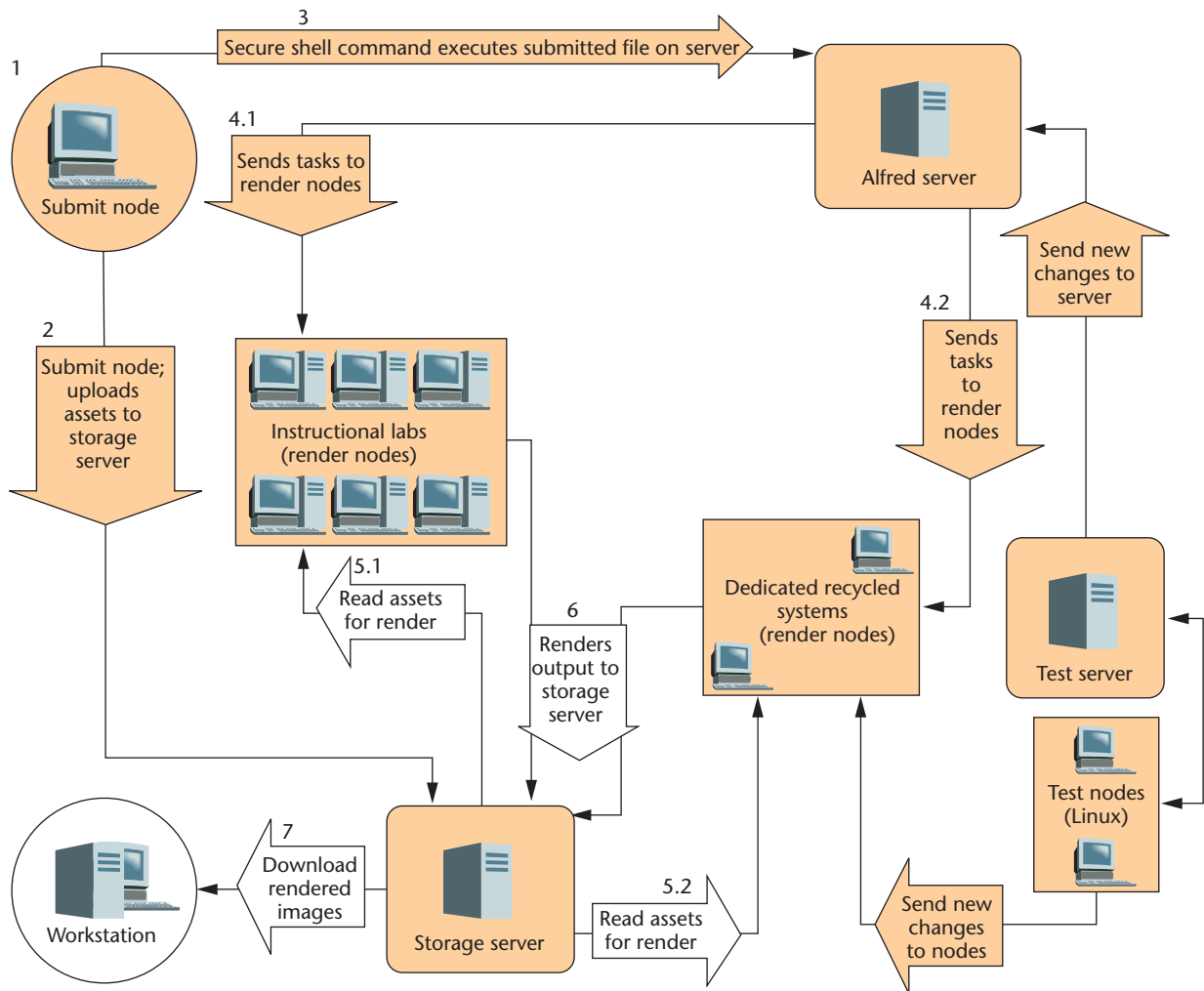
frames. Therefore, when considering multiple frames, the serial portion of a job is almost negligible. Rendering is not ideally suited for applying Amdahl's or Gustafson's laws, as the number of steps involved in rendering each frame might significantly differ depending on parameters such as light, texture changes, and shadow within a particular frame.

Equation 1 also assumes that all render nodes have similar hardware configurations. In actuality, the total

rendering time is not only affected by render node processor speeds and the amount of memory available, but also by network speed, the total time each render node is available, and time to distribute and manage multiple rendering jobs. For most practical scenarios, Flatt and Kennedy,<sup>3</sup> Karp and Flatt,<sup>4</sup> Sun and Gustafson,<sup>5</sup> and Van-Catledge<sup>6</sup> show that  $E_{to}$  is a function of number of processors (or in this case number of render nodes) and is usually a constant factor. In high-bandwidth, homogenous scenarios, this term approaches zero. In Equation 1, the term can be assumed as zero.

The introduction of the DRE into a course dealing with animation or scientific visualization reduces the bottleneck associated with the rendering process. As a result, faculty members can assign real-world projects as part of a course, therefore, significantly adding to a positive learning experience. With an increase in the size of the render farm, students can undertake increasingly complex projects approaching full-fledged industry scenarios. Figure 1 presents the theoretical speedup in a 2-minute animation (at 30 fps and a render time of 2 minutes/frame).

The notion of harvesting compute cycles for scientific purposes is not new. Several well-publicized projects like the Search for Extraterrestrial Intelligence at Home (SETI@Home) have employed this technique successfully. However, the process of harvesting compute cycles for rendering purposes within a university setting poses a unique set of technical challenges. Some of these challenges include the need to maintain a considerably large storage space, setting up a rendering management process, and a focus on security considerations consistent with all federal data security requirements.



2 Overall workflow and setup of Purdue University's DRE.

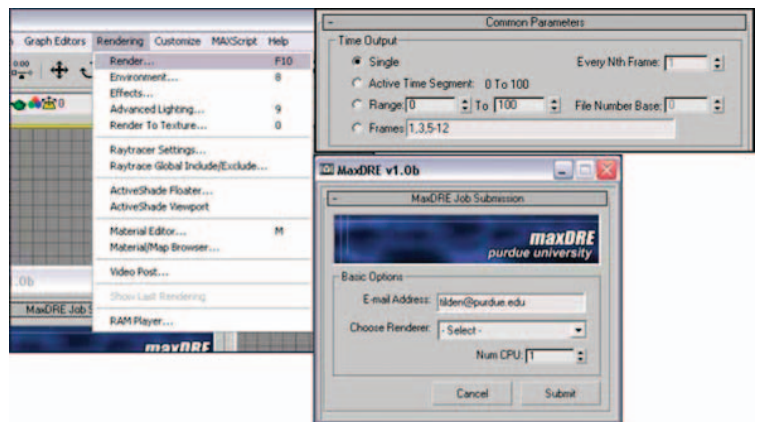
### Implementing the DRE

Purdue's DRE consists of several hardware and software resources all working together in a user-friendly workflow. Figure 2 presents the overall workflow in the DRE.

#### Submission nodes

Enhancing positive user experience is one of the main objectives of the DRE's design. From a user's perspective, one of the important aspects of the DRE is the easy-to-use interface for submitting render jobs. Because many of the render nodes are already in use as machines in instructional labs, it makes logical and practical sense that these machines should also be used as submission nodes (Step 1 in Figure 2). Because appropriate animation packages such as Autodesk 3ds max and Alias Maya are already installed on these machines, submission to the DRE can be greatly simplified. This is done through custom menus within the user interface of the front-end modeler. These menus integrate DRE access directly into the animation package interface (see Figure 3).

However, because all scripts necessary for DRE functionality are located on the central network's storage rather than on individual nodes, other machines on



3 Custom menu within the 3ds max software environment for submitting jobs to the DRE.

the campus network can also be used as submission nodes, even if they are not one of the DRE render nodes. In such cases, tools such as 3ds max and Maya might not be installed, or may be installed but not have the custom modifications. This requires some extra

Table 1. Composition of the DRE rendering nodes.

Primary Purpose	Number of Nodes	CPU Type	CPU Speed (MHz)	RAM (Mbytes)	Hard Disk Drive Capacity (Gbytes)	Available (Hours a Day)
DRE dedicated	48	Pentium 3	550	256	9	24
Instructional lab	22	Pentium 4	1,800	512	40	7
Instructional lab	22	Pentium 4	1,800	512	40	7
Instructional lab	22	Pentium 4	3,000	1,000	80	13
Instructional lab	22	Pentium 4	3,000	1,000	80	13
Instructional lab	22	Pentium 4	2,400	1,000	80	7
Instructional lab	15	Pentium 4	1,500	1,000	60	7
Instructional lab	24	Pentium 4	1,700	1,000	40	7
Instructional lab	22	Pentium 4	2,400	1,000	80	13

work on the user's part, as the files required for the render job need to be manually transferred to the network storage reserved for DRE use, and then submission scripts must be initiated. Currently, job submissions are accepted only through machines located on the campus network. However, we have started exploring allowing select off-campus submissions as our long-term goal is to provide a rendering resource available to a greater community.

#### Job server

We designed the DRE's implementation to shield users from the complexity of a distributed computing environment. To this end, one aspect of the DRE hardware to which users are not exposed to is the job distribution and management mechanism. This is the server responsible for receiving render job submissions and managing the distribution of these tasks.

Our server is called Alfred, as it uses Pixar's Alfred software for performing these tasks (see <https://renderman.pixar.com/products/tools/alfred.html>). When the job is submitted, the system immediately transfers all assets needed for completing this job to the central storage (Step 2 in Figure 2) and automatically transmits the job details to the Alfred server (Step 3 in Figure 2). The Alfred server acts as the resource broker and farms out the job to available render nodes (Steps 4.1 and 4.2 in Figure 2). Once a rendering job is submitted to the server, the server manages the job completely. This allows the user to log off the submission machine, making it available for other users or render jobs.

The server maintains a queue of all jobs submitted, checks the status of the render nodes to determine what machines are available, and then distributes smaller parts of the render jobs to each available render node. As the nodes complete their render tasks, the system sends the finished frames to the network storage area, and the server sends additional tasks to the render nodes as they become available. Some render jobs might have special requirements such as a preferred operating system or a minimum RAM requirement. The server also monitors these requirements and ensures that the render tasks are sent to appropriate render nodes. In some cases the server has to hold the tasks in a queue until necessary resources become available.

#### Render nodes

Purdue University's DRE is currently composed of 219 render nodes. Forty-eight of the render nodes are available throughout the day and are dedicated for rendering purposes, while the other nodes are lab machines available only during certain time slots such as evenings and weekends. The dedicated render nodes are Pentium 3, 550-MHz machines with 256 Mbytes of RAM. The other nodes are Pentium 4 class machines located in labs, in groups of 15 to 24 machines. Table 1 shows the specifications of all the machines, grouped by location, along with information about their availability for rendering during weekdays.

#### Software

The choice of an operating system will often dictate other software choices. As previously mentioned, most instructional lab machines are already running a Microsoft Windows-based OS. However, other machines such as the DRE distribution server are using a Unix-like OS. To facilitate secure communication between these machines, we installed Secure Shell (SSH) software on the Windows machines. All rendering nodes must have software rendering tools installed. We currently use Pixar's RenderMan software in our setup. These machines also run an additional service called Alfserver, which provides Alfred with information such as machine availability and receives task requests. Because availability to users is the highest priority for instructional lab machines, when a user logs in to a render node, the Alfserver process stops, preventing the machine from receiving new render tasks. Any currently running render tasks are also stopped and restarted when the machine frees up.

On most render nodes, we've installed common animation packages such as Maya and 3ds max, with the custom DRE modifications mentioned earlier. However, some submission nodes might have animation packages without the custom modifications or even simple communications packages such as SSH with no animation packages. This means that machines configured for a variety of different lab uses can all still be used as submission nodes for the DRE.

All of the DRE machines use some custom written Python scripts for communicating with each other and transferring required assets such as model files. All of these scripts required for job control and communica-

tion are located on the network storage so that all machines can use them without our having to install the scripts directly on every machine.

### **Network and storage**

Networking is a key component for any distributed rendering system, as this is the medium through which rendering tasks and assets are distributed to the render nodes as well as how completed renders are made available to the users. The DRE rendering nodes, storage, job management nodes, and submission nodes are connected through a Gigabit Ethernet pipeline. Also, all students, staff, and faculty receive a unique username that they can use for accessing email and network storage, creating a personal homepage, and logging into public lab machines on the network. This means that when users submit jobs to the DRE, their username and password can be authenticated against a centralized Lightweight Directory Access Protocol database. Any changes in usernames and passwords in any part of Purdue University's system are immediately propagated to the DRE system.

Also, completed renders can be placed on network storage so that the user can easily retrieve them from any other location on the campus network. We have also dedicated a portion of the central network storage to the DRE. In addition to storing completed renders, this area is also a shared space for storing other information such as the rendering source files, front-end software extensions, unique plug-ins, and DRE access scripts.

The render nodes read any required assets from the central storage location as indicated in Steps 5.1 and 5.2 of Figure 2. After the entire rendering job is completed, the results are delivered as rendered images back to a folder within the users' profile located on the central storage (see Step 6 in Figure 2). On the far right of Figure 2, we are also building provisions for test jobs. We incorporated these design variations into the DRE to permit early error detection and sampling frames in complicated rendering jobs. Upon completing the render job, the DRE system automatically sends out an email to the users indicating that they can download the rendered frames from the central network storage (see Step 7 in Figure 2). This marks the completion of the rendering process and workflow.

### **Impact and future plans**

One of the first major contributions the DRE has provided is the ability to incorporate significant curricular and pedagogical innovations in how we teach animation and data visualization. For example, students can now practice multipass rendering, something that was not possible before DRE implementation. Under this methodology, students can render each scene element (such as shadows, diffuse color, and specularities) separately. This allows them to practice postproduction operations, where each of these individual characteristics is run through compositing software into the original image. If there is an error in one of the characteristics, the error can be isolated and fixed without affecting other image parts. These types of alternate rendering methodologies are prevalent in the industry and students

can now practice this as part of their regular course work.

The DRE has positively affected users' experiences with the rendering process. Students no longer have to shepherd their individual jobs or concern themselves with the security of their work. The DRE also provides users with the ability to use one-touch job deployment for their rendering jobs. Email notifications at various points in the rendering process keep the users informed of job progress. Current plans for the DRE also include the ability to dispatch and track jobs from mobile devices such as PDAs and cell phones that will completely relieve users from the need to visit instructional labs or be physically connected to track and submit their jobs.

The impact that the DRE can have on the curriculum is significant. Previously, it was not uncommon for instructors to limit their expectations for student work based on the available computation resources in a lab. Students typically ran jobs on a single workstation, which often took days to render for a short animation scene. This tied up the workstation so others could not use it during rendering. Instructors were reluctant to give assignments or would lower their expectations based on the need to keep resources free in labs. With the DRE it's possible to raise expectations of student work in a class because more computation resources are available to the student. Students also choose to complete more challenging problems given the resources available with the DRE.

The next stage of the DRE development effort will link the system outside the physical boundaries of Purdue's campus and forge concrete links into the TeraGrid.<sup>7</sup> The National Science Foundation is investing significant resources into the development of the TeraGrid, which is a critical part of the national cyberinfrastructure. To achieve this, the rendering process needs to be implemented through an open source renderer—a process that adds substantial complexity to the design. As a first step toward this, we are currently testing the DRE over a Condor pool, which allows for job check-pointing. So, if a particular render node encounters critical errors during the rendering process, the work performed on that node is not completely lost. Rather, the render node running Condor will immediately transmit the render job's status to the central server, which will in turn reallocate the job to continue on a different node, without having to restart the complete frame.

A new security model that is consistent with the TeraGrid security framework will also need to be implemented. The focus of the DRE design will continue to emphasize positive user experience while lowering the threshold of entry for diverse fields ranging from visual and performing arts to nanotechnology. This dual focus on usability and easy access will eventually make the DRE available to students and faculty members nationwide at no cost. ■

---

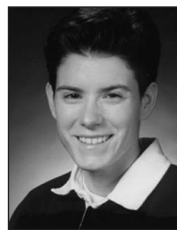
### **References**

1. G.M. Amdahl, "Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities," *AFIPS Conf. Proc.*, vol. 30, 1967, pp. 483-485.

2. J.L. Gustafson, "Reevaluating Amdahl's Law," *Comm. ACM*, vol. 31, no. 5, 1988, pp. 532-533.
3. H.P. Flatt and K. Kennedy, "Performance of Parallel Processors," *Parallel Computing*, vol. 12, 1989, pp. 1-20.
4. A. Karp and H.P. Flatt, "Measuring Parallel Processor Performance," *Comm. ACM*, vol. 3, no. 5, 1990, pp. 539-543.
5. X.H. Sun, and J.L. Gustafson, "Toward a Better Parallel Performance Metric," *Parallel Computing*, vol. 17, 1991, pp. 1093-1109.
6. F.A. Van-Catledge, "Toward a General Model for Evaluating the Relative Performance of Computer Systems," *Int'l J. Supercomputing Applications*, vol. 3, no. 2, 1989, pp. 100-108.
7. C. Catlett, "The Philosophy of TeraGrid: Building an Open, Extensible, Distributed TeraScale Facility," *Proc. 2nd IEEE/ACM Int'l Symp. Cluster Computing and the Grid (CCGRID)*, IEEE CS Press, 2002, p. 8.



**Krishna P.C. Madhavan** is a research scientist for teaching and learning applications with the NSF-funded Envision Center for Data Perceptualization and Rosen Center for Advanced Computing, Information Technology at Purdue University. His research interests include the integration of high-performance tools into regular curriculum. Madhavan received a PhD in computer-assisted assessment systems from Purdue University. Contact him at [cm@purdue.edu](mailto:cm@purdue.edu).



**Laura L. Arns** is an associate director and research scientist for the Envision Center for Data Perceptualization at Purdue University. Her research interests include applied virtual environments, distributed virtual environments, human factors in VR, and VR usability. Arns has a BA in computer science and mathematics from Wartburg College and an MS and a PhD in computer science from Iowa State University. Contact her at [larns@purdue.edu](mailto:larns@purdue.edu).



**Gary R. Bertoline** is an associate vice president and director of the Rosen Center for Advanced Computing and professor of computer graphics technology at Purdue University. His research interests include applying computer graphics; immersive interactive environments; and VR to visualize, interact, and analyze engineering and scientific data and information. Bertoline has a BS from Northern Michigan University and a PhD from Ohio State University in industrial technology. Contact him at [bertoline@purdue.edu](mailto:bertoline@purdue.edu).

For further information on this or any other computing topic, please visit our Digital Library at <http://www.computer.org/publications/dlib>.

Join the IEEE Computer Society  
online at



[www.computer.org/join/](http://www.computer.org/join/)

Complete the online application and get

- immediate online access to **Computer**
- a free e-mail alias — **you@computer.org**
- free access to 100 online books on technology topics
- free access to more than 100 distance learning course titles
- access to the IEEE Computer Society Digital Library for only \$118

Read about all the benefits of joining the Society at  
[www.computer.org/join/benefits.htm](http://www.computer.org/join/benefits.htm)