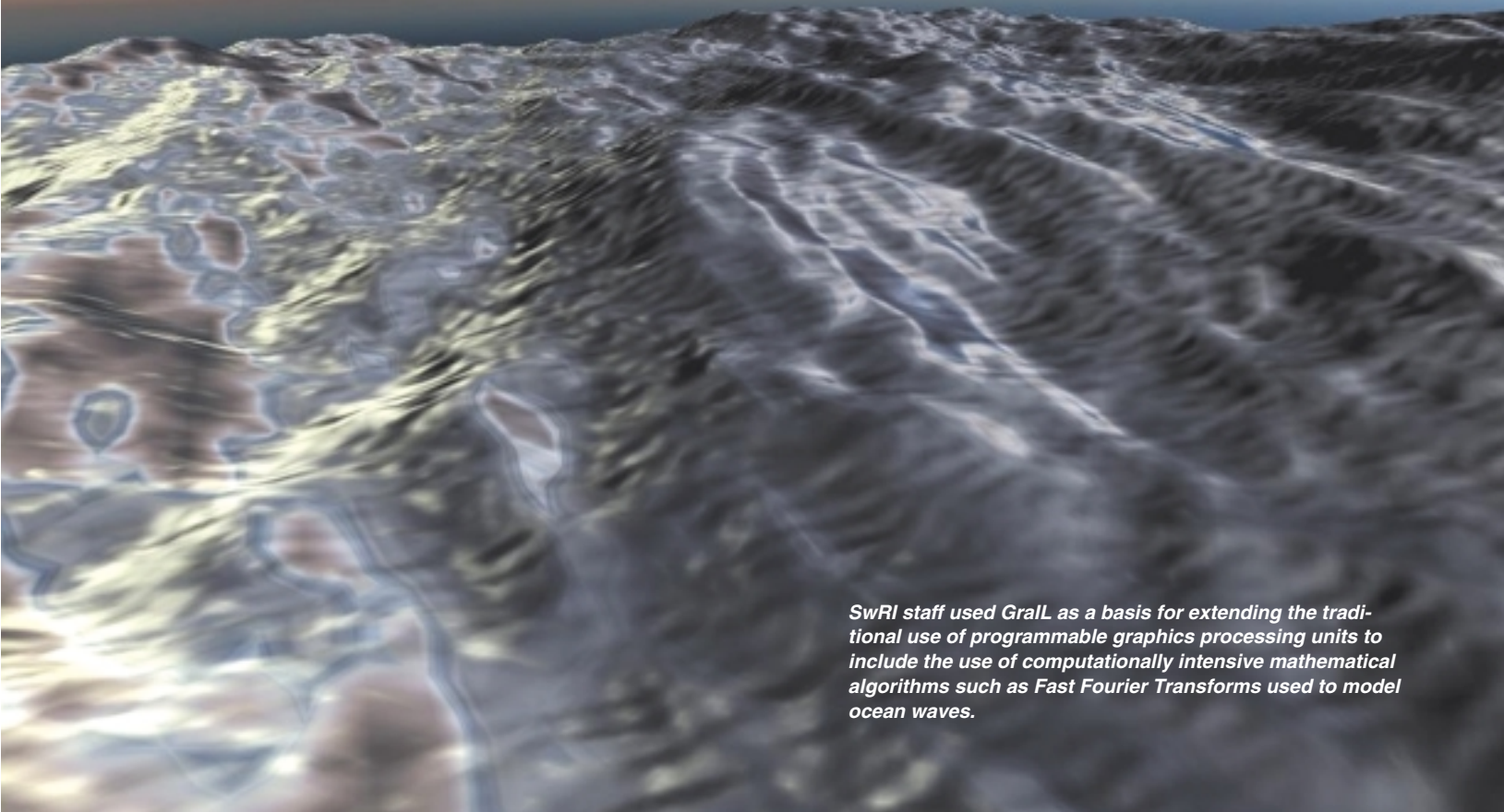


The Quest for Grall™

SwRI-designed graphics engine reduces cost, increases flexibility on graphics-intensive applications



SwRI staff used Grall as a basis for extending the traditional use of programmable graphics processing units to include the use of computationally intensive mathematical algorithms such as Fast Fourier Transforms used to model ocean waves.

By J. Brian Fisher

Real-time, interactive 3-D graphics such as those in video games and immersive virtual environments enhance computer applications ranging from entertainment to training and simulation systems. The advent of high-performance, low-cost, consumer-level graphics cards — personal computer hardware that renders 3-D graphics — allows software developers to generate these graphics in ways not previously possible. Software development tools, or graphics engines, generate the data processed on these graphics cards to produce highly realistic display images.

Focused on developing systems for training and simulation, SwRI software engineers have used a variety of proprietary commercial-off-the-shelf (COTS) software tools or “graphics engines” to produce a variety of applications. These graphics engines simplify the development of new software by providing a link

between high-level custom user applications and low-level graphics functions implemented in run-time libraries and on graphics cards.

While the use of a COTS graphics engine simplifies application development, the associated licensing requirements, restrictions and costs greatly limit the ability to develop and deliver cost-effective graphics applications that offer SwRI clients the flexibility they need. In addition, the lack of source code available to graphics engine users significantly complicates debugging and severely limits the ability to extend the capabilities of the engine. Staff members realized that an SwRI-owned graphics engine would offer better Institute control over distribution and licensing of software to clients and provide access to, and control over, the source code, significantly increasing development flexibility.



J. Brian Fisher is a manager for the Advanced Technologies Department in the Training, Simulation and Performance Improvement Division. He manages a team highly skilled in developing software programs for real-time 3-D graphics, modeling and computer systems. GraIL was designed to take advantage of recent advances in programmable GPUs to produce detailed visual effects such as real-time environment mapping.

Technical Approach

Institute engineers recently completed development of a graphics engine called the **Graphics Interface Library (GraIL™)**, providing the basis for current and future 3-D graphics application development. Initial efforts focused on identifying and developing a set of core function-

ality — software capabilities that provide a basic level of functions that, in turn, can be used to build a broad range of applications. Features in the initial version include a highly optimized math library, window and channel support, texture loading, model loading, shadows, scene graph implementation, callbacks, and lighting. The goal was to provide basic graphics capabilities in a flexible architecture, capitalizing on recent advances in computer graphics hardware and software.

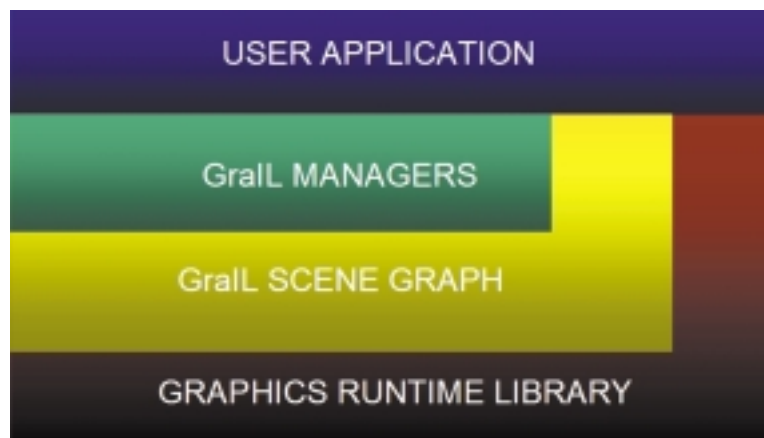
Designing for Flexibility

Developing a highly capable graphics engine is a significant effort that often requires years to complete. SwRI engineers wanted to emphasize flexibility in the initial design to support future GraIL development. Selecting the underlying graphics run-time library — the software that provides the link between a high-level 3-D graphics application and the graphics hardware — played a key role early in the design process. To ease future enhancement and maximize portability, SwRI engineers selected the industry-standard

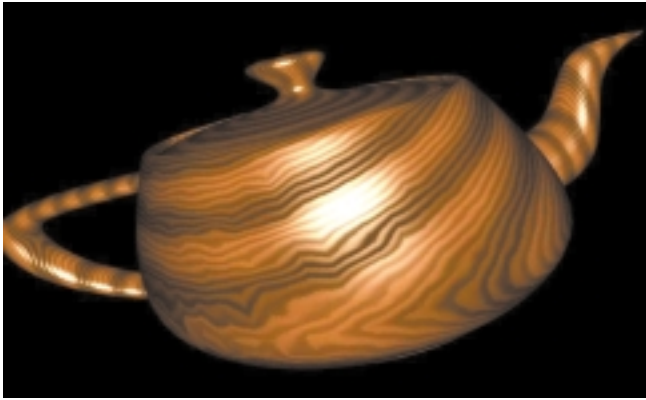
OpenGL® graphics run-time library that supports multiple operating systems including Microsoft® Windows and Linux®.

A flexible plug-in architecture allows users to extend GraIL's capabilities without modifying the core architecture, simplifying future development. In this approach, detailed interfaces are defined to help developers generate new modules and integrate them into GraIL with relative ease. These modules may be developed outside the GraIL core and dropped in as dynamically linked libraries. The SwRI development team used this approach to develop several GraIL modules — includ-

ing model and texture loaders and input device handlers. Planned enhancements using this approach include adding



The GraIL architecture provides multiple ways to access functionality to support developers with varying experience levels.



Following a long standing tradition for graphic developers, GraLL was used to model a 3-D teapot as one of its first applications.

new model and texture loaders and incorporating new input devices, such as position tracking systems, to support immersive virtual environments.

Software developers access GraLL capabilities through a robust application programming interface (API) which simplifies software application development by providing a collection of functions accessible at multiple levels. GraLL uses a manager-level approach to provide inexperienced users high-level access to engine functionality to develop basic 3-D graphics applications. The GraLL manager approach hides the low-level details of the graphics implementation allowing developers to use simple commands to perform basic operations. Low-level access is also provided directly to the GraLL scene graph, which stores data describing the graphical scene in a hierarchical structure, and the OpenGL run-time library, providing more advanced users the ability to control the low-level features of the engine to develop more sophisticated applications.

Designing for Speed

GraLL takes advantage of recent advances in computational algorithms and consumer graphics card hardware capabilities, which have significantly improved the way 3-D graphics are rendered. Rendering real-time computer graphics is a math-intensive process requiring numerous matrix operations during each rendered frame. Any optimization in mathematical computations has a significant effect on the overall performance of a graphics application. SwRI engineers implemented the GraLL math library by

GraLL has been used to support multiple internal and client-funded projects, including this effort to provide interactive 3-D models of the human anatomy for military training.

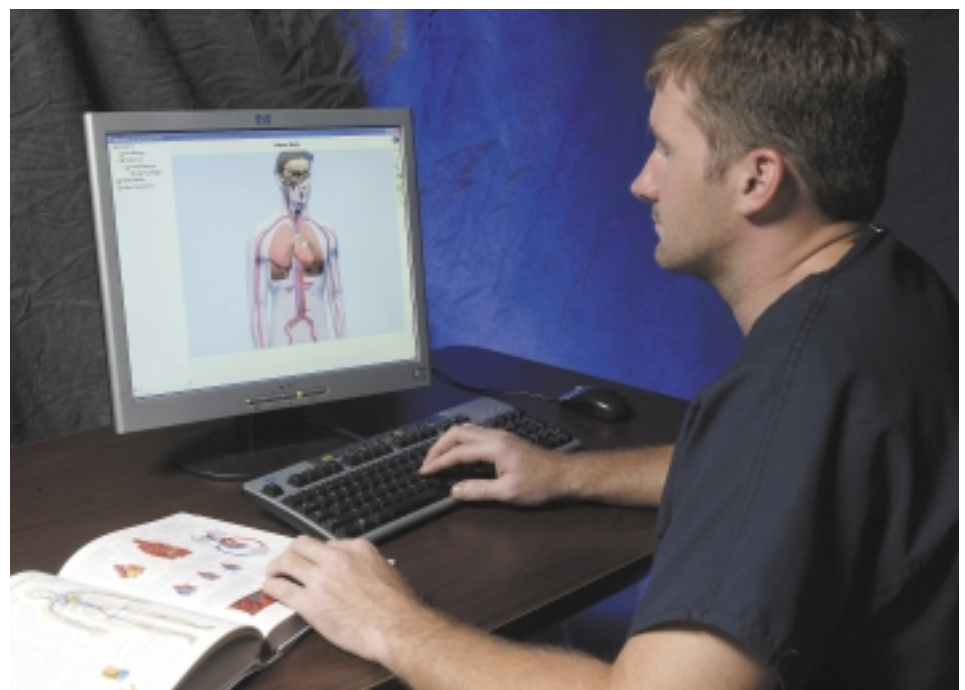
extending an optimized template approach to support additional mathematical operations specific to 3-D graphics rendering. This approach improved performance of the math computations by a factor of two to three.

The introduction of a programmable graphics processing unit (GPU) on consumer-level graphics cards has significantly advanced real-time 3-D graphics capabilities. GPUs are highly optimized graphics chips that accelerate 3-D rendering. These chips are now programmable using languages referred to as “shader languages” or “shaders.” These shaders allow programmers to implement graphics effects that significantly improve the realism of computer-generated imagery through highly optimized programs. The SwRI team designed GraLL to take advantage of these capabilities by building shader support directly into the core, simplifying the use of these techniques, and providing support for very detailed and powerful graphics capabilities. GraLL supports the traditional use of shaders to support visual effects such as surface materials, reflection, refraction, bump mapping, environment mapping and per-pixel lighting, which can dramatically improve realism, leading to more effective training environments.

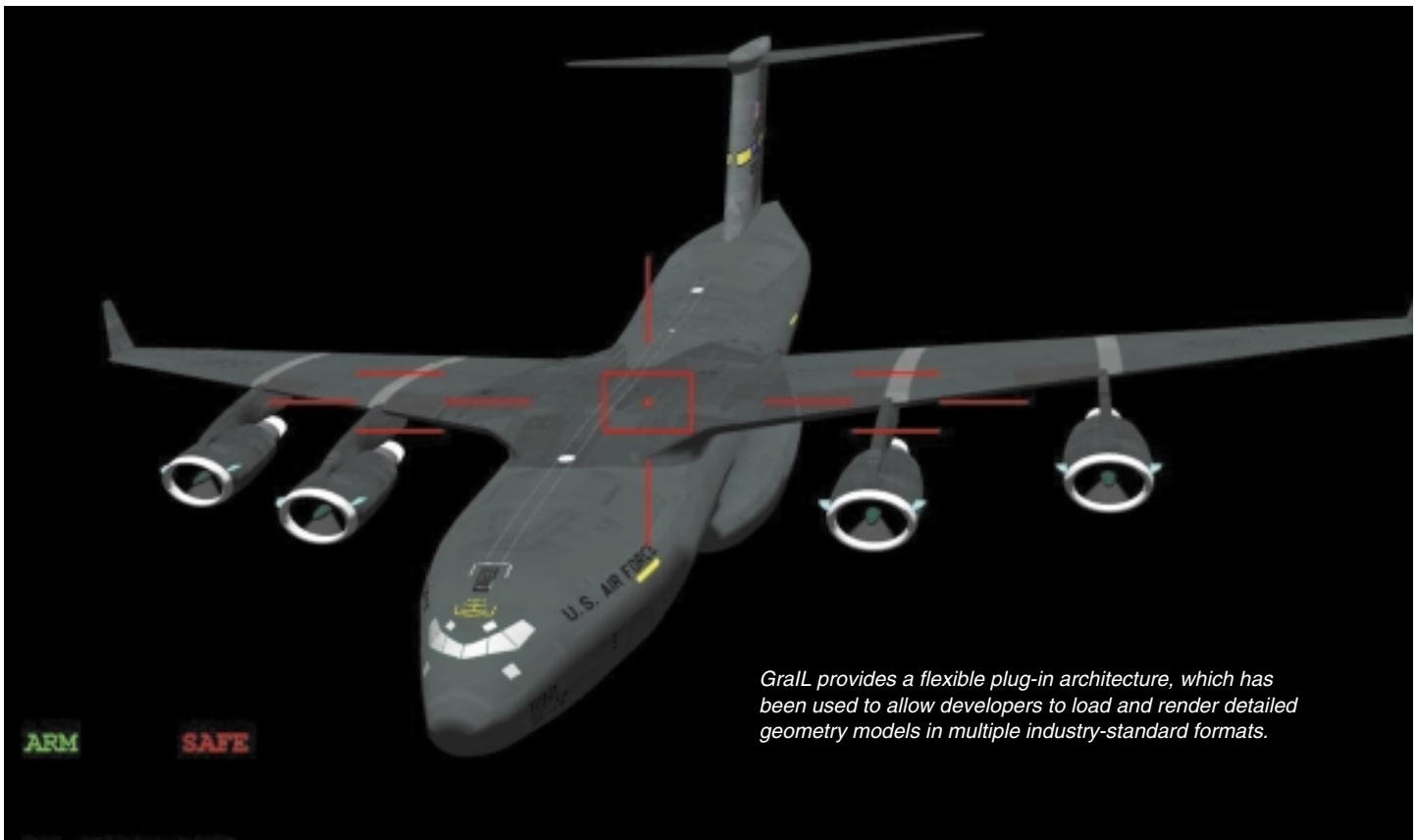
In addition, SwRI has developed methods in which shaders can be used in non-traditional ways to relieve some of the computational load on the central processing unit (CPU) by taking advantage of the GPU’s capabilities to rapidly perform mathematical operations in parallel. Specifically, the staff used GraLL to implement a shader program that computes fast Fourier transforms (FFTs) on the GPU to support the simulation of a dynamic height field to model waves on the ocean surface. This method of using the GPU will become increasingly valuable because GPU performance is growing at a rate much faster than that of CPUs. Because GraLL capitalizes on these capabilities, it is well positioned to provide significantly better processing speeds than other engines.

Extending the Core

Once the core was in place, SwRI engineers began enhancing GraLL by incorporating additional functions. These extensions include motion models, animated geometry, 2-D text overlays, transparency, basic collision detection and a particle system. Motion models make 3-D applications more interactive by allowing the user to move various graphical elements, including objects in the scene or the viewpoint, using an input device



D015016-0064



such as a mouse or keyboard. Animated geometry allows the programmer to incorporate models into the virtual environment which change over time to simulate objects such as a beating heart. Collision detection is used in many 3-D applications to illustrate the interactions of 3-D objects that come into contact with each other. Implementing collision detection can be very challenging as it often involves computationally intensive algorithms. A limited collision detection capability was implemented in Grall to provide basic detection while providing a framework for more detailed algorithms as required for future graphics efforts.

Particle systems can increase the realism of 3-D applications by creating dynamic geometry that can display programmable behavior. Particle systems are often used to simulate various effects such as fluid flow, smoke and explosions. The particle system implemented in Grall is based on algorithms that appear in sophisticated 3-D modeling tools typically used in generating detailed animations. The Grall particle system is physics-based and supports multiple standard particle types and texture mapping options, random, configurable, and age- and distance-based parameter variations, and integrated viscous, linear, nonlinear and collision-sensitive particle effects. SwRI

has used the Grall particle system to develop an interactive application that illustrates blood flow through the human heart in real time.

Conclusion

The Grall engine will enable SwRI to be more competitive in the 3-D graphics application market by increasing the staff's ability to develop and market innovative approaches to generating 3-D graphics applications. By having access to the source code, software developers who use Grall will have much better insight into the internal workings of the engine, allowing them to create unique features not available in commercial engines. It will also provide control over the licensing of applications.

Grall is already being used to support multiple internal research development efforts and client-funded projects. These efforts include the development of a web-enabled application for training in aircraft maintenance procedures using interactive 3-D graphics, an application for providing interactive 3-D instructional material for medical training and an application for simulating ocean surfaces in real-time. The quest for a stronger, more flexible graphics engine will continue with future development,

enhancement and extension of the Grall capabilities. ❖

Comments about this article? Contact Fisher at (210) 522-3762, or bfisher@swri.org. To discuss this article online see www.swri.org/forums



Acknowledgments

The author gratefully acknowledges the dedication and contributions of the Grall development team including Principal Engineer Eric Peterson, Senior Research Engineer Billy Couvillion, Senior Research Analyst Malachi Wurpts and Senior Research Analyst Ryan Logan, all of the Training, Simulation and Performance Improvement Division.