Proceedings of the ASME 2011 World Conference on Innovative Virtual Reality
WINVR2011
June 27-29, 2011, Milan, Italy

WINVR2011-5575

DEVELOPING THE PLANCK MISSION SIMULATION AS A MULTI-PLATFORM IMMERSIVE APPLICATION

Gerald A. Dekker Jr.
John Moreland
Purdue University Calumet, CIVS
Hammond, Indiana, USA

Jatila van der Veen
University of California at Santa Barbara
Santa Barbara, California, USA

ABSTRACT

Planck is an international mission led by the European Space Agency with significant contribution by NASA, designed to measure the anisotropy of the Cosmic Microwave Background (CMB), the oldest radiation of the universe, with the greatest accuracy and precision of any such CMB experiment to date. The present work was completed as part of the Planck Education and Public Outreach (E/PO) effort to communicate the results of Planck science to the public. The Planck Mission Simulation is a multiplatform, interactive visualization of the mission, from launch to orbital insertion to data gathering operations. The simulation was developed for a number of hardware and software configurations. Originally designed for a multi-screen virtual reality system, the scope of project grew to include other systems, including 3D kiosk displays, stereoscopic televisions, and domed-roomed systems. Implementation factors, technical details, and lessons learned from deployment on various platforms are discussed.

INTRODUCTION

The Planck Mission and Cosmic Microwave Background

Launched on May 14, 2009, the international Planck Mission is currently in its second year of mapping the microwave sky in nine frequency channels (30 to 857 GHz), with a temperature sensitivity of a few microKelvin and spatial resolution as fine as 5 arc minutes. The main scientific objective of Planck is to measure the spatial anisotropies of the temperature of the Cosmic Microwave Background (CMB), with an accuracy set by fundamental astrophysical limits. Planck will extract essentially all the information contained in the CMB temperature anisotropies, with which to constrain models of how the universe originated and evolved. Planck will also measure to high accuracy the polarization of the CMB, which encodes not only a wealth of cosmological information, but also provides a unique probe of the thermal history of the Universe during the time when the first stars and galaxies formed. In addition, the Planck All-Sky surveys will produce a wealth of information on the properties of extragalactic sources and on the dust and gas in our own Galaxy [1].

Planck is the third generation of satellite to map the CMB, after CoBE (launched in 1989) and WMAP (launched in 2000). Planck is an international mission, led by the European Space Agency with significant contributions from NASA, designed to improve our understanding of the origin and evolution of the universe. According to our current understanding, the universe began some 13.7 billion years ago in an unimaginably hot, dense, compact state, from which it suddenly expanded after an unknown period of dormancy. Within the first unimaginably short time span (on the order of 10-30 seconds), the universe underwent a tremendous stretching, called inflation, which stretched the universe by some 40 orders of magnitude. Due to the inherent quantum uncertainties on small scales, this period of inflation ended at slightly different times throughout the infant universe, producing small fluctuations or inhomogeneities in the matter-radiation field. As the universe continued to expand and cool, going through a series of phases of particle production, it became a fluid of tightly coupled matter and radiation. Dark matter, which does not interact electromagnetically, condensed out first, collecting in pockets due to its mutual gravitational attraction. As clumps of dark matter aggregated, they would have induced gravity-driven acoustic oscillations in the matter-radiation fluid, much like stones dropping into a pond induce

waves in the water. These acoustic waves propagated through the expanding universe, interfering with each other, until around 380,000 years after the so-called Big Bang, when the universe became cool enough for matter and radiation to decouple, and light could travel freely for the first time. The microKelvin fluctuations in the CMB that we measure today reveal the variations in the light that scattered off the acoustic waves in the early universe for the last time. Just as an acoustic power spectrum of a musical instrument gives information about the instrument itself from the fundamental and higher harmonics, similarly the spatial power spectrum of the CMB gives us a picture of the fundamental and higher harmonics of the primordial acoustic waves of the infant universe at the time when the universe first became transparent.

Prior to 380,000 years, the universe was opaque to electromagnetic radiation. Thus, the CMB is the oldest light we can observe, and holds clues as to the origin, properties, and ultimate fate of the universe. The next few years should prove to be quite exciting, as the data from Planck are expected provide answers to many of the fundamental questions about the universe, such as: Why is it that only around 4% of all observable matter and energy in the universe is baryonic – made up of protons, neutrons, and electrons – while 96% is in some dark form that we have not been able to measure directly? Was the initial expansion isotropic, apart from Gaussian fluctuations, or was there a primordial anisotropy? What is the actual geometry of the universe, which may be apparent in a repeating pattern in the CMB anisotropy? Planck's ability to quantify the polarization of the CMB photons is expected to reveal the presence of gravitational radiation in the early universe which will test fundamental physical models beyond the Standard Model of particles and interactions.

Educating the Public about the Real Science behind Cosmology

The science results that are anticipated from the Planck Mission are crucial to our search to answer questions in fundamental physics today. The counterpart to the Large Hadron Collider (LHC), which is attempting to recreate energies that are expected to have existed in the early history of the universe, Planck is providing the window into the primordial universe. The results of Planck may change our understanding of fundamental physics, thus it is important to educate the public about this mission.

Both NASA and the European Space Agency are working on a variety of products to educate the public about the science behind Planck. NASA's Science Mission Directorate funds educational projects which support the science goals of all the missions, as well as the goals of NASA's Office of Education. These goals are to strengthen science and technology education in the US, raise public awareness about the value of the various science missions, and contribute to the future science and technology workforce in the US through educational opportunities for students, teachers, and the public. The American collaborators on the Planck Mission, funded by NASA, are developing a range of products for informal education which are designed to teach the public about the science goals of Planck and the technology behind the mission, as well as products for community college astronomy courses to improve understanding of cosmology for both instructors and students. The Planck Mission Simulation is designed to increase public awareness of the Planck Mission and technology through an interactive 3D application.

PROBLEM AND MOTIVATION

Why visualization for E/PO?

With the advent of modern 3D and immersive visualization technology, it is possible to give people a virtual experience of concepts in astronomy that went into the implementation of the Planck Mission, and the technology of the satellite which is expected to yield such spectacular results. Through creating an immersive virtual solar system, the Planck Mission Simulation allows users to visualize distances in the solar system, the relationship of Planck to the Earth and Moon, and see Planck in orbit around the second Lagrange point, or L2, in the gravitational field of the Earth-Sun system.

The Planck Mission Simulation is one of two projects that comprise the Planck Visualization Project. The Planck Mission Simulation, developed at Purdue University Calumet, is designed to give students, teachers, and the public an overall familiarity with the satellite in space. The other project – The Music of the CMB, developed at the University of California, Santa Barbara – utilizes the technique of sonification of the power spectrum of the CMB anisotropy to allow users to examine the fundamental and higher harmonics of the power spectrum, and explore different hypothetical model universes which would produce different power spectra, and hence different sounds [2], [3].

METHODOLOGY

Diverse System Deployment Overview

Methods for the systematic design of immersive environments including elements such as navigation and interaction techniques have been established in the past [4]. While initially the Planck Mission Simulation was developed as an immersive system project, it grew in scope, growing to become capable of being run on many presentation

configurations; each deployment tailored to a specific system and platform. The process by which the Planck application became so diverse was a combination of coincidence and planning. From the onset of the project, the Planck Mission Simulation program has had the capability to run on multiple operating systems: namely, Microsoft Windows and UNIX variants. Only after the conceptual basis of the project itself had matured did the need for a Cross Platform/Multiple Systems (CPMS) application become apparent, and so became a primary project concern.

Initially designed to run on the cross-platform visualization software package VR Juggler [5], the problem of porting to multiple operating systems had already been partially solved even before the issue had become a project goal. At the same time, an unofficial version of the application using the open source GLFW tool library [6] had been implemented to make testing possible outside of a full-scale VR system deployment. The GLFW library was also cross platform; this unofficial GLFW version of the Planck Mission Simulation became the core basis by which the application was deployed on small kiosk and portable systems. As the project progressed, with CPMS functionality in mind, special care was then taken to insure that any new software dependencies (such as image, sound, and video libraries) were capable of running on UNIX variants as well as Microsoft Windows. To simplify the concerns of code portability, all of the Planck Application's fundamental software modules were written in C, which was known to be the most portable low-level programming language.

The Planck Mission Simulation Project has been deployed in two major variants: a phase I and a phase II release. Characteristics of the phase I release included: a mission visualization scenario, pre-recorded demonstrations via control state capture, and a menu system driven by hardware mapped control buttons. In addition to functionality allowing the user to follow along and interact with the Planck satellite during its mission, phase I also implemented features of the solar system to enhance and add additional possible learning components as in previous VR astronomy projects [7, 8]. The phase II release was a near complete redesign of phase I. Important differences included: improved models and navigation for the mission visualization, interactive exploration of satellite instrumentation, direct video/audio capture, expanded stereoscopic vision functions (GLFW based), and soft controls with pop-up panels in lieu of hardware buttons.

Deployment using VR Juggler

The VR Juggler Suite is a sophisticated software package for the creation of virtual reality systems. As it is the primary package run at the Purdue Calumet CIVS facilities, VR Juggler compatibility became the primary focus of the initial deployment efforts. The VR Juggler Suite exhibits many desirable qualities; however, it cannot be said that it is without significant problems.

The VR Juggler project has always emphasized that the software suite is cross platform [9]: this is certainly true. Unfortunately, the VR Juggler project has progressed from pre-built binary releases to a "source code only" model. Experience has shown that significant build dependency requirements combined with poor documentation often means that building the VR Juggler Suite on a new platform could be a significant undertaking. Consequently, it was determined that VR Juggler should only be used on systems where an older pre-built binary release or a valid repository installation package exists. Additionally, configuration of a VR Juggler installation to match the display and control hardware is non-trivial, and further complicated by incomplete and out-dated documentation.

VR Juggler is a software layer: it offers abstraction of cross-platform functions such as display initialization, thread management, audio playback, and access to control devices. The Planck Visualization application made good use of the control and display device abstraction functionality, but avoided use of any of the additional abstraction methods. The reason for this: it was more efficient to incorporate audio and thread management functions on a lower level within the core Planck application code, as any use of a non-standard abstraction function needed to be changed for each additional front-end toolkit.

Concerns of Deployment using GLFW

GLFW is a multi-platform toolkit designed to facilitate development of OpenGL applications [10]. In many respects, it is equivalent to a modern version of the classic Open GL Utility Toolkit (GLUT) developed by Mark J. Kilgard while employed at SGI [11]. In addition to Open GL context management, GLFW provides platform abstraction functions, such as thread management, precision timers, and OpenGL extension loading. Usage of abstraction methods beyond basic display and control management was avoided by the Planck application for the reasons mentioned prior: because of non-standard functionality issues.

As GLFW is essentially just an OpenGL context management layer, provisions for stereoscopic vision were implemented by manipulating the OpenGL viewing frustum: creating two independent views and shifting/warping these views. To support various stereoscopic viewing technologies, multiple viewports (side-by-side), frame buffer color masks (anaglyph), and stencils (interlaced) were used to complete the stereoscopic illusion. Options for hardware quad-buffered stereo were also included for devices capable of utilizing this mode of operation. As the GLFW deployment was primarily for budget constrained deployments (such as commodity

Table 1: EVALUATION OF THIRD PARTY LIBRARIES AND TOOLS FOR USE IN THE SIMULATION.

Criteria (All Categories)	Category	Role	Library/Tool	Comments	Used
Must be free of charge. Must be readily available and/or easily built on target platforms. As dynamic library: undecorated function names (C-style interface). Must be documented. Professionalism in implementation. Focus/compactness: should not include features which are unneeded or unrelated to primary role.	Graphics	System Interface	VR Juggler [5]	Prerequisite, not evaluated against criteria list.	Yes
		System Interface	GLFW [5]	Preferred for its similarity to the GLUT [8] API.	Yes
	Audio	System Interface	Open AL [12]	No other libraries considered.	Yes
		Codec	Ogg Vorbis [13]	Initial compressed format used.	Yes
		Codec	Mpg123 [14]	Preferred compressed format (mp3), API similar to Ogg Vorbis [10] .	Yes
		Codec	FFmpeg [15]	Significant external dependencies, violates compactness criteria.	No
	Image	Decompression	FreeImage [16]	Superior ratings in all criteria categories.	Yes
		Decompression	DevIL [17]	Library name unsuitable for funded/publically released project.	No
	Video	Encoder	Revel [18]	Simplified Xvidcore [16] interface; minor modifications necessary for recompilation.	Yes
		Encoder	Xvidcore [19]	Direct usage of library somewhat complicated; would require re-implementation of Revel [15] features.	Yes
	Font	Renderer	GLF [20]	Primary font renderer for Phase I only. Limited font selection; poor small point size appearance.	Yes
		Renderer	FreeType2 [21]	Excellent library, simpler solution found.	No
		Processor	BMFont [22]	TrueType font to texture image/definition conversion tool. Excellent bitmap font solution.	Yes

workstation based kiosks), no provision for head tracking devices was implemented.

Unlike the VR Juggler deployment, no sophisticated control device was presupposed. Control of the application was typically accomplished using a standard desktop mouse and keyboard; this is not to say that other control devices could not be accommodated (indeed, a game controller device was used successfully in a deployment).Use of a keyboard/mouse control arrangement proved to the correct choice for the GLFW based deployment: it was the often the only factor reliably consistent amongst the deployment platforms.

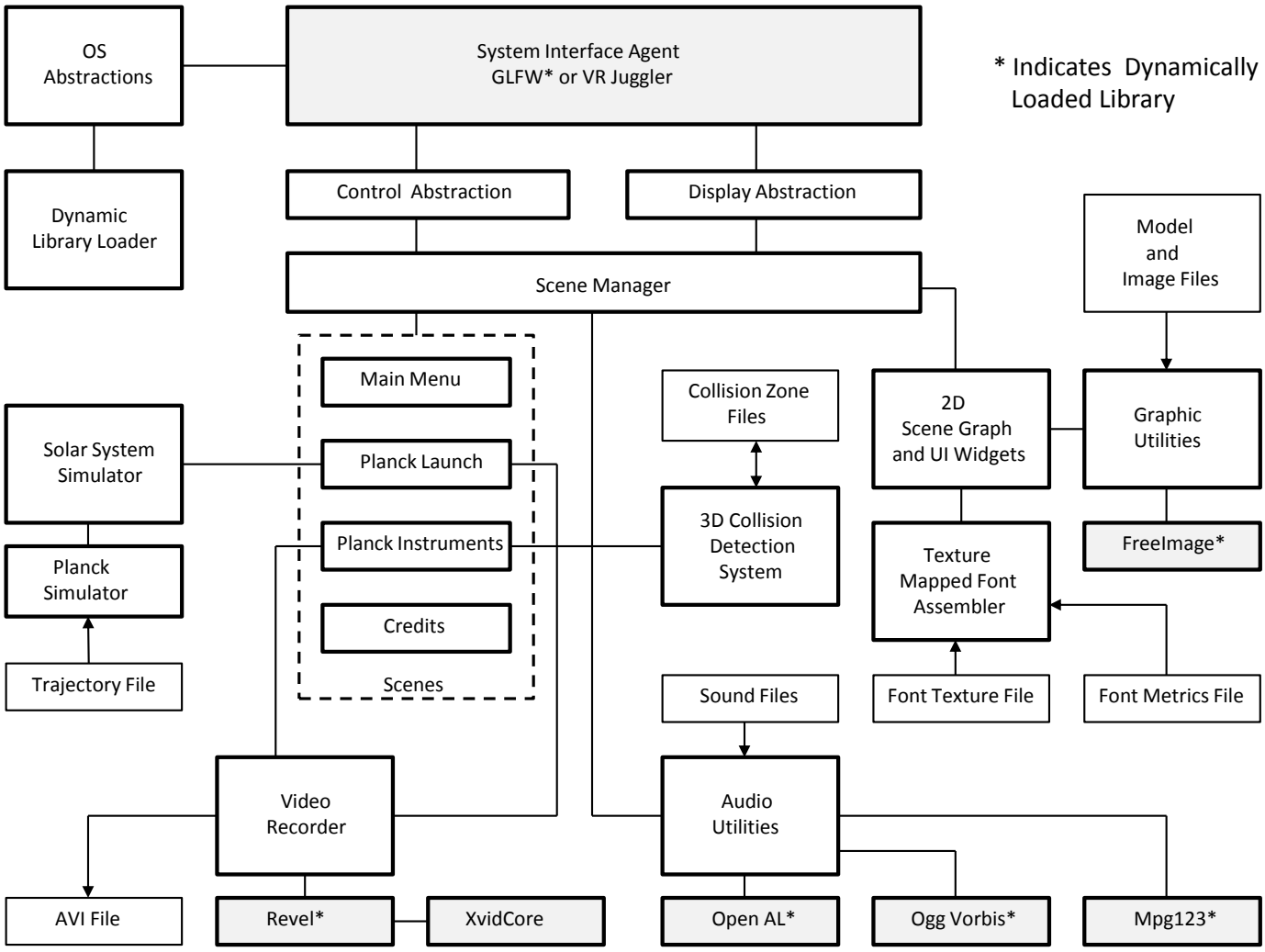

Figure 1: ISOLATION OF LIBRARY DEPENDENCIES THROUGH USE OF DYNAMIC LOADING.

Porting to Different Operating Systems

There are numerous pitfalls associated with porting code to differing operating systems: writing code which using standard libraries eliminates many of these issues. However, a number of important programming functions differ amongst operating systems; namely, dynamic library loading, thread/process management, and timing functions. Additionally, any non-standard libraries chosen for usage must be capable of being built on the operating system of interest.

The Planck Mission Simulation makes extensive use of pure runtime library loading. To accommodate operating system differences, conditional compilation with abstraction functions were used to map the correct function calls (e.g. for POSIX: *dlopen*, *dlclose*, *dlsym*; on Windows: *LoadLibrary*, *GetProcessAddress*, *FreeLibrary*). Dynamic library loading enabled the Planck application to shed unavailable functionality gracefully, yet still operate in a meaningful way. For instance, if the audio library (Open AL [12]) was not available on a system, a dynamic linkage using a static export library would cause the application to fail on a system level. However, since pure dynamic loading was used, the absence of the library could be handled in the program code, and the audio functionality disabled without a general failure. Figure 1 depicts the isolation of library dependencies from core application functionality. The ability to selectively shed libraries dramatically decreased the time needed to deploy to a new system, though such a system would run with reduced program functionality. Related to this, Open GL extensions were loaded by using a method abstracted

Figure 2: ACTIVE ZONE DESIGN TOOL PANEL BEING USED WITHIN THE APPLICATION.

Figure 3: VIDEO/AUDIO CAPTURE CONTROL MENU.

for each platform (i.e. *glXGetProcAddressARB* versus *wglGetProcAddress*).

Also using conditional compilation and abstraction were the thread management functions; appropriate POSIX and Microsoft Windows routines were called to start and stop the all worker threads. The video recording portion of the Planck application employed a background thread for compressing and encoding video and audio into a standard Audio Video Interleave (AVI) file using Xvid [16] compression. Further, all audio effects (background music, etc.) were played using Open AL [9] streamed buffers maintained by worker threads. The advantage of thread-based streamed audio was two-fold: no apparent file load time lags, and dramatically reduced memory usage. Assuredly, concurrent programming utilizing threads improved overall program performance substantially; however, programming complexity increased in a non-trivial way due to provisions for thread resource release—especially in the case of unexpected user-induced program termination.

The intricacies of employing third-party libraries on various operating systems are perhaps the single greatest frustration in creating non-trivial applications. Table 1 presents criteria and choices made in library selection for the project. Where viable, it was found expedient to implement the desired functionality within the Planck application rather than rely on a third-party source. File loaders for simple uncompressed image and audio formats were directly incorporated into the Planck application code, with additional options for compressed formats (e.g. MPEG3 audio, JPEG textures) available if the required library was found on the system. By simply replacing crucial image and audio data with uncompressed equivalents, rapid deployment on a platform for which the compression libraries had not yet been built was possible. Additionally, the Planck application employed a built-in Wavefront OBJ model file format loader [23]: this ensured that models could be loaded and displayed for all systems which the application could itself be compiled on.

User Interface Design

Phase I of the Planck Mission Visualization incorporated button mapping of hardware devices (IS-900 wand, keyboard, and joystick) in conjunction with a screen cursor to facilitate user interactions [24]. This proved somewhat problematic, as the button mappings were numerous and somewhat complex. Based on user reactions, the Phase II release of the project refined the control interface to consist primarily of pop-up menus with labeled soft buttons. Not only did this facilitate greater system portability though reduced hardware requirements, it advertised functionality to the user. As nearly all program functionality is discoverable via navigation of the menus structure, users became aware of program options which had previously been overlooked. To facilitate ease of use, the control interface utilized item focus highlighting and sticky value detents on sliding controls for common value configurations.

Table 2: SAMPLE OF SYSTEM DEPLOYMENTS

System	Operating System	Hardware Capabilities	Application Versions	3D Technology	Notes
Two-screen VR system	Windows XP, Fedora, Windows7	High end workstation	Phase I, II	Infitec Passive	IS900 tracker, VR Juggler
Large-scale dome system	Ubuntu	High end workstation	Phase I	Active Stereo	GLFW, hardware mode; joystick controls
Hyundai 3D monitor	Windows Vista	Moderate	Phase I,II	Circular Polarization, Passive	GLFW, interlaced mode
Generic Intel Laptop	Windows XP	Low end	Phase I	Anaglyph	GLFW, anaglyph mode. Low frame rate
Aging Desktop	FreeBSD	Low end	Phase I	None	GLFW
Portable Visualization System	Windows7	High end Workstation	Phase II	Circular polarization modulator	IS900 tracker, VR Juggler
Mac Mini	Windows7	Moderate	Phase I,II	Anaglyph	Small form factor system
Desktop	Wine (Ubuntu)	Moderate	Phase II	Anaglyph	Essentially native performance speed
Mac Mini	Wine (OSX)	Moderate	Phase I	None	Low frame rate

Some of the new functionality incorporated into Phase II of the project (the interactive instrumentation explorer) required that some means exist to conveniently define active areas of the interactive model. The use of proximity to trigger the display of information has been shown to be effective in certain cases [25]. Proximity triggers were design in Phase II such that as users flew through certain areas within the Planck instrumentation, relevant graphics and information were then displayed. Since the shapes of various instrumentation was typically complex, a method needed to be devised to accurately define the appropriate proximity triggers. As illustrated in Figure 2, the improved menu system was used with great effect to implement a tool panel which allowed creation and reviewing of active areas of the model to be used as proximity triggers. Active areas were defined as connected areas of spherical collision zones laid out by directly navigating the model space. Collision sphere zone size was adjustable from a control on the interface panel. Once an area was effectively mapped, the zone definition could be saved to a text file, where descriptive text could be added along with associations of thumbnail images and narrative sound files.

As Phase II of the project included real-time video and audio capabilities, it was necessary to implement a control panel interface for the encoder operations. Shown in Figure 3, control buttons and sliders on the panel allowed selection of various recording parameters, including video output size and frame rate. An active text display was also included on the control panel to display video capture status and statistics.

ACTUAL SYSTEM DEPLOYMENT RESULTS AND OBSERVATIONS

As presented in Table 2, the Planck Mission Simulation has been tested and successfully deployed on a wide range of systems. Generally, hardware constraints were found to be the prime limiting factor, although the applications requirements are surprisingly modest. The Phase I release application was verified to run acceptably well at 1024x768 resolution on a machine running FreeBSD having an Intel 1 Ghz Pentium III processor and a Nvidia Geforce MX 420 video card. Phase II is more demanding, but has been run on a Windows XP system with circa 2002 hardware (AthlonXP 2000, Nvidia Geforce4 Ti 4200) at reasonable frame rates. These results are in alignment with the project goals, as low hardware requirements make use of the application accessible to a wider audience.

CONCLUSION AND FUTURE WORK

The Planck Mission Simulation has been developed for deployment on a number of platforms. As the Planck Satellite continues to collect data and scientific discoveries from the data are made public, the Planck mission will become ever-more present in the public eye. The Education / Public Outreach arm of Planck mission is important to keeping the public well informed. Development of the software has produced a number of lessons learned. Minimizing third party dependencies and choosing libraries which port well when utilization is necessary are effective strategies for cross-platform deployment. Designing an application to fail gracefully when missing dependency libraries can greatly aid in rapid development of a new platform deployment. Usage of on-screen controls in lieu of hardware controller button mappings provides both a more intuitive interface and one which is useable in a greater variety of circumstances. Moreover, discrete usage of a common lower-level programming language, such as C, allows an application to be both portable and efficient; thereby facilitating usage of inexpensive hardware, or even reuse of older hardware, for visualization applications.

Future work includes a possible implementation of the Planck Mission Visualization using WebGL and HTML 5. Additionally, making the application available on mobile devices and additional platforms will likely be an ongoing part of development.

ACKNOWLEDGMENTS

The authors would like to acknowledge collaborators from the European Space Agency (ESA), the National Aeronautics and Space Administration (NASA), the Jet Propulsion Laboratory (JPL), and the Purdue Calumet Center for Innovation and Simulation (CIVS).

REFERENCES

[1] Planck Collaboration (2011). Planck Early Results: The Planck mission, Astronomy & Astrophysics manuscript no. Planck2011-1.1 c ESO 2011 January 12, 2011.

[2] van der Veen, J. (2010). Planck Visualization Project: Seeing and Hearing the CMB, in Science Education and Outreach: Forging a Path to the Future, ASP Conference Series, Vol. 431, 2010

[3] van der Veen, J., Alper, B., Smith, W., McGee, R., Lubin, P., and Kuchera-Morin, J. The Planck Visualization Project: Seeing and Hearing the CMB, poster delivered at the meeting of the American Astronomical Society, January 11, 2010, Seattle, WA.

[4] Bowman, D. A., & Hodges, L. F. (1999). Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments. *Journal of Visual Languages and Computing*, *10*(1), 37–53.

[5] www.vrjuggler.org

[6] www.glfw.org

[7] Yair, Y., Mintz, R., & Litvak, S. (2001). 3D-virtual reality in science education: An implication for astronomy teaching. *Journal of Computers in Mathematics and Science Teaching*, *20*(3), 293–306.

[8] Klimenko, S., Nielson, G. M., Nikitina, L., Nikitin, I., & Strassner, J. (2004). Virtual Planetarium: Learning Astronomy in Virtual Reality. *Proccedings of ED-MEDIA, 2004*.

[9] Cruz-Neira, C., Bierbaum, A., Hartling, P., Just, C., & Meinert, K. (2002). VR Juggler- An Open Source platform for virtual reality applications. In AIAA Aerospace Sciences Meeting & Exhibit, 40 th, Reno, NV.

[10] Berglund, C. (2006). GLFW-An OpenGL Framework.

[11] Kilgard, M. J. (1996). The OpenGL utility toolkit (GLUT) programming interface API version 3.

[12] Hiebert, G. (n.d.). Openal 1.1 specification and reference.

[13] http://xiph.org/vorbis/

[14] http://www.mpg123.de/

[15] http://www.ffmpeg.org/

[16] http://freeimage.sourceforge.net/

[17] http://openil.sourceforge.net/

[18] http://revel.sourceforge.net/

[19] Xvid. (2004). 1.0. 3 MPEG-4 codec source code.

[20] http://paulbourke.net/oldstuff/glf/

[21] http://www.freetype.org/index2.html

[22] http://www.angelcode.com/products/bmfont/

[23] Murray, J., & Van Ryper, W. (2005). Wavefront OBJ File Format Summary.

[24] www.intersense.com

[25] Celentano, A., & Pittarello, F. (2004). Observing and adapting user behavior in navigational 3D interfaces. In *Proceedings of the working conference on Advanced visual interfaces* (pp. 275–282).