Three Dimensional Computer Graphics Federates for the 2012 Smackdown Simulation

Crystal Fordyce
(843) 513-8980
crystal@gemofcrystal.com

Swetha Govindaiah
(256) 714-3018
Swethagovindh@gmail.com

Sean Muratet
(256) 417-8237
smuratet@gmail.com

Daniel A. O'Neil
Marshall Space Flight Center
Huntsville, AL 35811
(256) 544-5405
daniel.a.oneil@nasa.gov

Bradley C. Schricker
Dynetics, Inc.
1002 Explorer Blvd.
Huntsville, AL 35806
(256) 964-4979
brad.schricker@dynetics.com

Abstract: *The Simulation Interoperability Standards Organization (SISO) Smackdown is a two-year old annual event held at the 2012 Spring Simulation Interoperability Workshop (SIW). A primary objective of the Smackdown event is to provide college students with hands-on experience in developing distributed simulations using High Level Architecture (HLA). Participating for the second time, the University of Alabama in Huntsville (UAHuntsville) deployed four federates, two federates simulated a communications server and a lunar communications satellite with a radio. The other two federates generated 3D computer graphics displays for the communication satellite constellation and for the surface based lunar resupply mission. Using the Light-Weight Java Graphics Library, the satellite display federate presented a lunar-texture mapped sphere of the moon and four Telemetry Data Relay Satellites (TDRS), which received object attributes from the lunar communications satellite federate to drive their motion. The surface mission display federate was an enhanced version of the federate developed by ForwardSim, Inc. for the 2011 Smackdown simulation. Enhancements included a dead-reckoning algorithm and a visual indication of which communication satellite was in line of sight of Hadley Rille. This paper concentrates on these two federates by describing the functions, algorithms, HLA object attributes received from other federates, development experiences and recommendations for future, participating Smackdown teams.*

1. Introduction

Responding to the needs of Industry and the National Aeronautics and Space Administration (NASA), the Simulation Interoperability Standards Organization initiated the annual Smackdown Simulation event. Primarily, this event provides college students with hands-on experience in the development of distributed simulations using the High Level Architecture (HLA) standard. To provide context for the simulation, NASA identified a lunar resupply mission where a cargo lander transfers supplies from an orbiting space-craft to the surface. Participating universities provided additional mission scenario details and developed the federates that simulate the lunar resupply and exploration assets.

Participating universities in the 2012 Smackdown simulation included the University of Alabama in Huntsville (UAHuntsville), Massachusetts Institute of Technology (MIT), Penn State University, universities from Genoa, Pisa, and Rome, Italy, and Technion University in Israel.

Technion contributed a mission scenario diagramming application while Penn State developed simulations of a cargo landing vehicle and a cargo transfer rover. These federates are identified in **Figure 1** as the Lunar Shuttle and the Lunar Rover. The MIT team produced simulations of a mobile resource utilization plant and a scouting hopper that jumped from one place to another in search of minerals for the mobile resource utilization plant. These federates are specified in **Figure 1** as the MIT Hopper and MIT Resource Plant. The three universities from Italy formed two teams. One team developed a simulation of an asteroid tracking and planetary defense system while the other team developed a simulation of a supply depot. **Figure 1** depicts Genoa's Supply Depot federate.

Johnson Space Center (JSC) contributed an environment simulation that provided reference frames of the sun, earth, and moon and a one-second ***heartbeat*** for the distributed simulation. The orbiting spacecraft was also developed by JSC. Both federates are respectively depicted in **Figure 1** as the NASA environment and NASA vehicle federate.

The Central Run-time Component (CRC) that managed the distributed simulation, the Virtual Private Network (VPN) for remote participation, and the local area network for integrating participating computers at the conference were operated by the JSC team.

The UAHuntsville team contributed four federates: (1) a radio communications server, (2) a satellite federate with an orbital propagator and radio, (3) a 3D graphics display of a constellation of four communications satellites, and (4) a 3D visualization of the lunar resupply mission. **Figure 1** identifies these federates, respectively: Communications Server, LCANSat, Satellite Constellation Display, and fsi3DViewer UAH. Another paper describes the first two federates. This paper describes the 3D graphics federates. The Pitch Recorder appears as a federate in **Figure 1** because the UAHuntsville team recorded data, which was used in the SISO Smackdown simulation play back in Second Life described later in this paper.

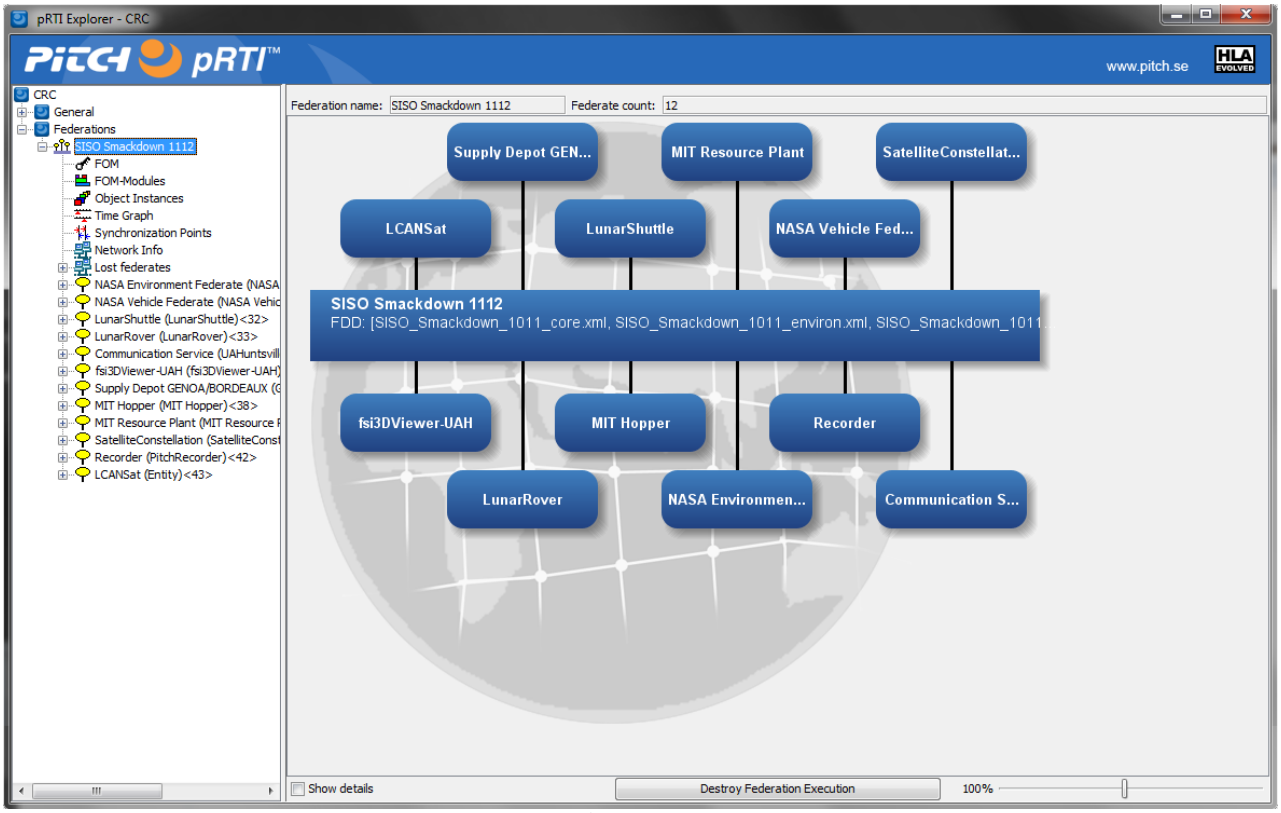

Figure 1. Lolly-Pop Diagram of the 2012 SISO Smackdown Federation

2. Satellite Constellation Analysis

The UAHunstville team's first participation in the 2011 SISO Smackdown consisted of an HLA federate simulating a singular lunar communication satellite following Keplerian orbital motion without perturbations. For their second participation effort in the 2012 Smackdown event, a constellation of lunar communication satellites were simulated using the same orbital trajectories but with added restrictions to aid in visualization. One such restriction was to limit the satellite's altitude to somewhat realistic measures for the purpose of visualizing scale. Secondly, the number of satellites had to be small to keep frame rate high and network traffic to a minimum, yet optimized for line of sight with the lunar base.

The lowest altitude and the highest altitude was ascertained by asking the following questions:

- What is the lowest possible altitude the communication satellites can orbit the moon without the threat of disintegration or crashing into the surface?
- What is the highest altitude circular orbiting satellites can obtain and yet remain stable?

To answer the first question, past NASA experience gives some insight in how low a satellite can orbit the moon. On Apollo 16 in April 24, 1972 a small satellite orbiting the moon at 89 to 122 km above the lunar surface suddenly crashed after only a few days orbiting. The problem was determined to be due to lunar mascons, gravitational irregularities due to heavier concentrations of mass on the lunar surface, unrelated to mountains or other topography, but instead related to dense lava material concentrated at certain locations [1]. In short, the lowest altitude was restricted to 100 km.

Constellations, or groups of satellites, typically require dozens of satellites to ensure continuous global coverage because ground-tracks of low orbiting satellites are small. In addition, low orbit satellites present more challenges for line of sight with ground-based entities in that the satellites travel at high speeds and are only visible between 25 and 45 minutes depending on the satellite altitude and the position of the ground entity, requiring fast switching from one satellite to another. To avoid these difficulties, satellite constellations consisting of three and four satellites were examined for which satellite altitudes achieve optimal line of sight but yet fall within the restricted limits.

The highest altitude limit was determined by figuring out the altitude at which the earth's gravity will affect the circular orbit such that it will be flung away from the Moon in a hyperbolic orbit. "High-altitude circular orbits around the Moon are unstable," says Todd A. Ely, senior engineer for guidance, navigation, and control at NASA's Jet Propulsion Laboratory. "Put a satellite into a circular lunar orbit above an altitude of about 750 miles (1200 km) and it'll either crash into the lunar surface or it'll be flung away from the Moon altogether [2]." With that in mind, the highest altitude was restricted to 1200 km.

Simplifying the Moon's shape to a sphere and applying a circular orbit enabled geometric determination of the limiting altitude where the line of sight between the three and the four satellite configurations just glances the surface of the moon. The minimum altitude where the satellite's line of sight grazes off the moon's surface is depicted in **Figure 2.** Three system minimum altitude for line of sight among satellites for the three satellite case. The following formula gives this minimum altitude (h_s):

$$\sin(\beta) = \frac{r_{moon}}{r_{moon} + h_s}$$

$$h_s = r_{moon}\left(\frac{1}{\sin(\beta)} - 1\right)$$

Separating the three communications satellites by 120 degrees, β equals 30 degrees (see **Figure 2** for reference) and with the radius of the moon taken as r_{moon} = 1737.10 km, the minimum altitude is thus 1,737.10 km.

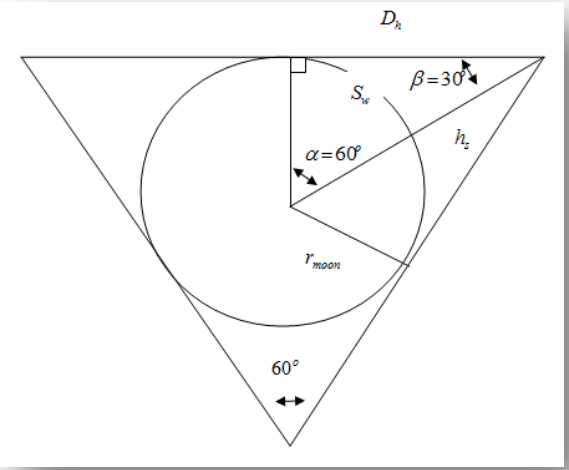

Figure 2. Three system minimum altitude for line of sight among satellites

For a four satellite configuration, the minimum altitude where the satellite's line of sight grazes off the moon's surface is depicted in **Figure 3**. The following derivation gives this minimum altitude (h_s):

$$z = \sqrt{r_m^2 + r_m^2} = \sqrt{2r_m^2}$$
$$z = h_s + r_m$$
$$h_s = \sqrt{2r_m^2} - r_m = \sqrt{2}r_m - r_m$$
$$= r_m(\sqrt{2} - 1)$$
$$= r_m(1.414 - 1)$$
$$r_m(0.414)$$

Plugging in the radius of the moon, the satellite constellation of four satellites must have altitudes above 719 km. A four constellation satellite, constrained in a circular orbit between 719 and 1200 km driven by Keplerian motion became the basis for UAHuntsville's second participatory effort in the 2012 SISO Smackdown event. Optimization of the altitude involves mission objectives such as amount of ground coverage desired, communication delays and orbit periods.

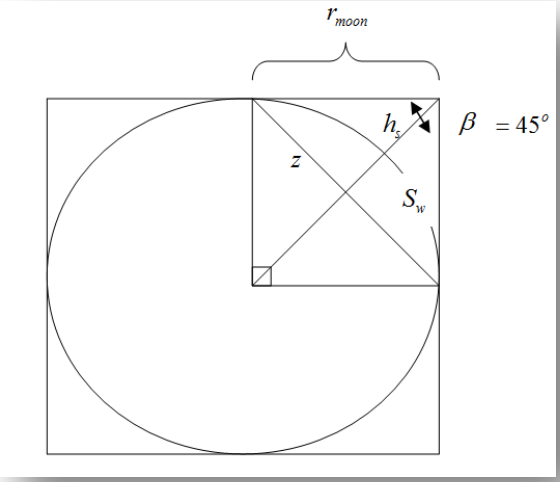

Figure 3. Four system minimum altitude for line of sight among satellites

3. Line of Sight Algorithm Analysis

Three line of sight (LOS) algorithms were analyzed by the UAHuntsville team. One algorithm mathematically defined a rectangular view frustum with one end on the lunar surface located at Hadley Rille and the other end extended into space past the orbiting satellites. The algorithm compared the positions of the satellites to the boundaries of the view frustum to determine whether a satellite was in view of the lunar surface assets at Hadley Rille. This algorithm required a comparison of a satellite position point against four corners of the frustum.

A second algorithm combined ray-tracing and point clipping algorithms [3][4][5]. This algorithm translated the satellite positions into screen coordinates and compared the positions to an 80x80 patch located at Hadley Rille. For each pixel in the canvas, the algorithm calculates the direction vector between satellite position and the pixel location. Using that direction vector and sphere equation, the algorithm determines whether the direction vector is touching the sphere. If that's true then it will determine the point of intersection of the direction vector on the sphere. Using the point, defined by the intersection of the vector with the sphere, a clipping algorithm determines whether it lies within the patch. Ray tracing is computationally intensive, so this algorithm would have required pre-calculated points, which would hinder the ability to change orbits during the simulation.

The selected algorithm also used a ray and sphere intersection technique. This algorithm was used in the communications server federate and the orbital propagator code for the communications satellite federate. The LOS algorithm needed to have a mechanism for determining whether, if the line between two points – one in lunar orbit, the other on the lunar surface – did intersect the moon's surface, the intersection occurred between the two points. As such, the algorithm performs a calculation comparing the distances between:

(1) Communications satellite and the simulated entity located on the lunar surface
(2) Communications satellite and the lunar surface, itself

If (1) is greater than (2), then LOS would be set to *false*, as it would indicate that the lunar surface is between the two points. However, if (2) is greater than (1), LOS would be set to *true*. This line of sight algorithm is illustrated in **Figure 4.** Line Of Sight Algorithm

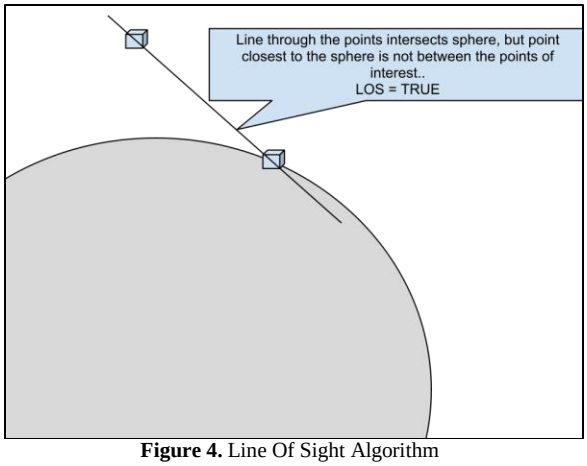

Figure 4. Line Of Sight Algorithm

The UAHuntsville satellite federate, identified as LCANSat in **Figure 1**, generated a line-of-sight status within an orbital propagator object. This status was updated and each second and the Satellite Constellation Display federate reflected this attribute and presented the status on the display.

4. Satellite Constellation Display Federate

A 3D graphics federate displayed a constellation of four communications satellites orbiting the Moon. Each satellite has a color: red, green, blue, and yellow. A small green glowing sphere identifies the location of Hadley Rille. Lines of text below the moon presented the current location of each satellite and whether a satellite is within view of Hadley Rille.
Figure 5. Satellite Display Federate Architecture.

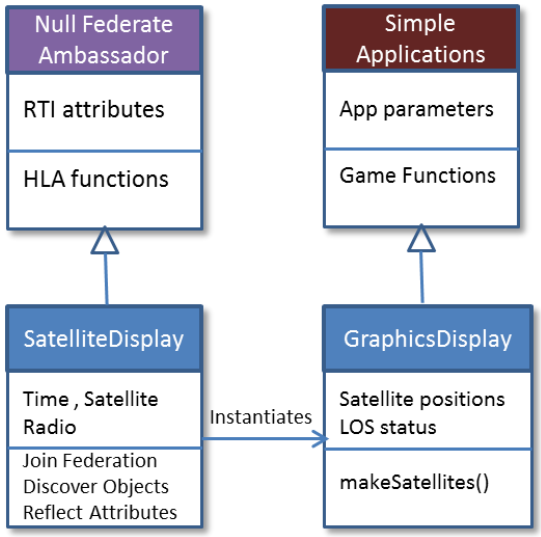

Figure 5. Satellite Display Federate Architecture

The Java RTI library provides the Null Federate Ambassador class. SatelliteDisplay extends the NullFederateAmbassador and instantiates the GraphicsDisplay object, which extends the SimpleApplications class, which is provided by the Java Monkey Engine (JME) code library. Satellite Display methods inherited from the Null Federate Ambassador include object discovery, attribute reflection, and joining the federation. Reflected attributes included the satellite positions and line of sight status. The GraphicsDisplay class instantiated the 3D models of the satellites with the MakeSatellites method.

Developed with the Light-Weight Java Graphics Library (LWJGL) and JME Integrated Development Environment (IDE), the Satellite Constellation Graphics Display federate presented a lunar texture mapped sphere with four orbiting Telemetry Data Relay Satellites (TDRS). The TDRS 3D mesh models were originally created by NASA in the FBX format. The models were converted to the WaveFront Obj format and imported into the JME IDE where it was converted into a binary file format. The JME IDE is built upon the NetBeans IDE so it is a familiar environment for Java developers. The Simple Applications class is a member of the JME code library; this class provides the functions for a game window and user interface controls. The GraphicDisplay class derived from the Simple Applications class so it generated the window and graphics objects in the display.

5. Surface Mission Visualization Federate

The Virtual Reality Modeling Language (VRML) based surface mission visualization federate visualizer allows an audience to clearly view the lunar mission. Originally developed by Daniel Verret at ForwardSim, the VRML federate went through an upgrade, performed by the UAHuntsville team, to enhance the visualization capabilities.

The most significant improvements pertained to the dead-reckoning algorithm implemented within VRML. In the simulation world, dead-reckoning is typically employed to minimize data traffic across the network. In this case, dead-reckoning was used to smooth the animation, making it both more pleasurable for an audience to view and easier to interpret the animation. Figure 6. Surface Mission Visualization Federatedepicts the software architecture of the Surface Mission visualization federate.

For VRML, the dead-reckoning algorithm takes reflected values of position, velocity, and acceleration and uses a blended interpolation to produce smooth animations while reflecting ground truth accurately. If values for velocity and acceleration are not reflected by an entity, dead reckoning calculations are not made and only the position will be reflected when available. In the context of VRML, this means that the position will be updated on the screen only when the data is made available by the owning federate. As is typical of dead-reckoning algorithms, minimum position changes with respect to the scale of the animation are required to redraw the animation. This methodology saves processing time and does not change the end result, in terms of the accuracy of the entities' movements. The entities will continue to be dead-reckoned even though they are not redrawn. This implementation requires a continuous loop to produce smooth animation with a consistent refresh rate. Due to the single-thread nature of MATLAB – the development environment of choice – dead-reckoning was implemented within the main federate execution loop with method calls for each object to redraw the animation.

Once the dead-reckoning portion completes, multiple callbacks are evoked to check to see if updated true values for position have been published by the federates responsible for the various entities being drawn.

More specifically to the Smackdown event, the four satellites provide intermittent coverage of the Hadley Rille site. The VRML model reflects this coverage by changing light color intensity and node textures to give a visual cue as to when a satellite is within visual range of the surface.

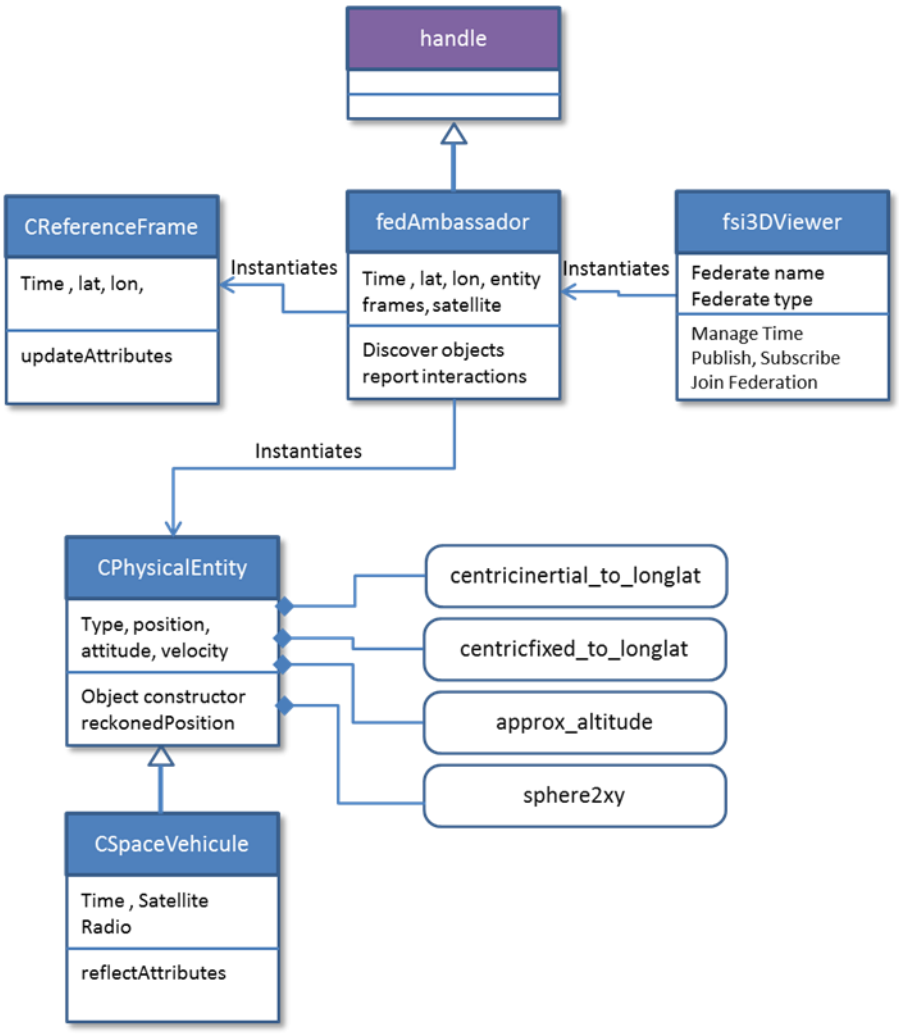

Figure 6. Surface Mission Visualization Federate

The UAHuntsville viewer incorporated new entity models in an effort to better represent the individual entities. As of the Smackdown event in March 2012, ground-clamping and collision detection were not implemented. These are features that could be implemented by a future UAHuntsville Smackdown team.

6. Virtual Reprise in Second Life

During the 2012 SISO Smackdown simulation, the UAHuntsville team used the Pitch Data Recorder to capture position attributes of the MIT Scouting Hopper and Mobile Resource Utilization Plant, and the Penn State Cargo Lander and Cargo Transfer Rover. Recorded data was processed with Excel to switch from a Y up axis to a Z up axis and the numbers were scaled down to fit within a property in Second Life. Team members used Google Sketchup to create 3D mesh models of the surface lunar resupply mission assets and export Collada files. These Collada files were uploaded into Second Life and Linden Scripting Language (LSL) programs read the scaled-down data to animate the models within Second Life.

The UAHuntsville team rented a 4,096 square meter sea-side parcel of land. The scene in **Figure 7** includes models, created by the UAHuntsville team, to represent (1) Penn State Cargo Transfer Vehicle, (2) MIT Mobile Resource Utilization Plant, (3) MIT Scouting Hopper, (4) Penn State Cargo Transfer vehicle, (5) Genoa Warehouse, and (6) UAHuntsville Communication Satellite.

The first AlaSim International conference occurred in the first week of May at the Von Braun Center in Huntsville Alabama. In one of the workshops, the UAHuntsville team demonstrated the virtual reprise of the 2012 SISO Smackdown simulation. Benefits of a recorded playback in a virtual online world include ad-hoc demonstrations to recruit more university teams for the SISO Smackdown and public outreach to teachers and students. Also, the virtual environment serves as an archive, so each SISO Smackdown simulation can be saved and people can see the evolution of the simulation each year.

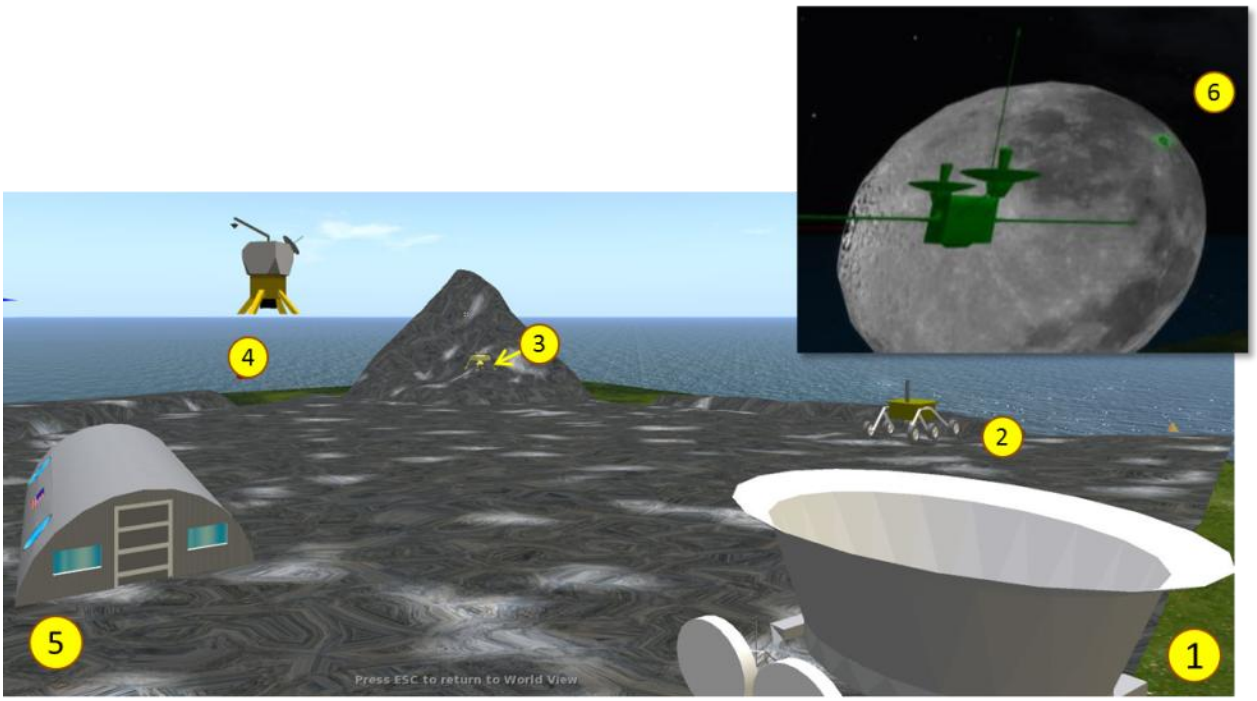

Figure 7. Virtual Play-Back of the SISO Smackdown Simulation in Second Life (1) Cargo Transfer Rover, (2) Mobile Resource Utilization Plant, (3) Scouting Hopper, (4) Cargo Lander, (5) Warehouse, (6) Communication satellite in view of Hadley Rille

While the team did not experience any difficulties technically, they did experience social problems in Second Life, particularly ***Griefers***. A ***Griefer*** is someone who hassles other denizens of a virtual world for the fun of it [7]. One of the UAHuntsville team members was attacked by colored cubes that cussed at her. Another team member was attacked by Griefers who changed the physical appearance of her avatar. Renting private property for the virtual reprise fixed the problems associated with public sand-boxes. Owners and renters can control access to private parcels.

Second Life supports e-mail, chat, and XML Remote Procedure Calls (XMLRPC), which enables communication among objects within Second Life and with external programs [6]. The UAHuntsville team experimented with XMLRPC to drive the movement of an object within Second Life from an external Java program. Second Life or another virtual world could serve as a common 3D visualization system for distributed simulations.

Benefits of a common virtual world for a distributed simulation include:

- a standardized approach to presenting system behaviors
- opportunities for remote participants to view the computer graphics
- a capability to archive the 3D models and system behaviors

The intent of the SISO Smackdown simulation is to give college teams experience with the HLA standard. The time and effort associated with creating computer graphics displays can detract from developing and integrating the simulation. A reusable code library for creating federates that interact with models in a virtual world allow all the SISO Smackdown teams to drive models in a common environment. Also, the separation of the 3D model building from the computer graphics display programming would enable non-programmers to participate in the development of the simulation. Mechanical engineers, architecture students, and artists could support the SISO Smackdown teams by creating the 3D models in the virtual world.

7. Potential Future Plans

Based on the positive experiences with the external communication technology and the negative social experiences in Second Life, a few members of the team concluded that a public virtual world may not be the best venue for a 3D graphics visualization system. There are a few open-source online virtual worlds available. Open Simulator uses the same communications protocols as Second Life [8]. Open Wonderland was originally developed by Sun and became open source when Oracle bought Sun [9]. Open Cobalt is another virtual world, which received development funds from the National Science Foundation (NSF) and has a scripting language based on Small Talk [10]. Open Cobalt is a virtual world browser based on a peer-to-peer communication protocol. A team could develop a federate to reflect object attributes and issue commands to virtual world objects via the browser code.

A modular FOM could define object attributes that translate into commands for object settings in the virtual environment. Such a capability would enable participating SISO Smackdown teams to drive virtual models from their federates. A university could host the central database of the virtual world and control access to participants and interested stakeholders. A free open source Massive Multiplayer Online Game (MMOG) development environment, named Multiverse, is available from MIT. Multiverse supports the Python and Java programming languages [11].

8. Conclusions and Recommendations

The 2012 SISO Smackdown Simulation provided a great opportunity for the UAHuntsville team to learn about 3D computer programming with the LWJGL, MatLab Simulink with VRML, and LSL in Second Life. Skills gained during the process included converting among a variety of file formats using the Adobe FBX converter, Blender, MeshLab, and Google Sketchup, uploading models into Second Life, scaling data and collaboratively building a virtual environment. Experiments conducted with XMLRPC indicate that an online virtual world could be used as a 3D visualization system for an HLA based distributed simulation. A recommendation is that a university team or individual research the leading open-source virtual world code-bases and develop a modular FOM and federate for driving models with reflected object attributes. Another recommendation is for SISO Smackdown participating teams to use the virtual world as a collaboration tool in the development of scenarios and sharing 3D models and scripts.

References

[1] Bell, Trudy, "Bizaare Lunar Orbits," *NASA Science News*, 6 November 2006, http://science.nasa.gov/science-news/science-at-nasa/2006/06nov_loworbit/, viewed on May 7, 2012.

[2] Bell, Trudy, "A New Paradigm for Lunar Orbits," NASA Science News, November 30, 2006, http://science.nasa.gov/science-news/science-at-nasa/2006/30nov_highorbit/, viewed on May 7, 2012.

[3] Codermind Team, "A raytracer in C++ - Introduction - What is ray tracing?," Codermind In a coder's mind, http://www.codermind.com/articles/Raytracer-in-C++-Introduction-What-is-ray-tracing.html, viewed on May 7, 2012.

[4] Eberly, David. *3D game engine design: a practical approach to real-time computer graphics*, 2nd edition, Morgan Kaufmann, 2006. ISBN 0-12-229063-1.

[5] Computer Graphics Lab, CS488/688: Introduction to Interactive Computer Graphics, "Point and Line Clipping," University of Waterloo, http://medialab.di.unipi.it/web/IUM/Waterloo/node27.html, viewed on May 7, 2012.

[6] Second Life Wiki, "Category:LSL XML-RPC," Linden Research, Inc., June 9, 2011, http://wiki.secondlife.com/wiki/Category:LSL_XML-RPC, viewed on May 7, 2012.

[7] Second Life Wiki, "Griefer," Linden Research, Inc., August 17, 2009, http://wiki.secondlife.com/wiki/Griefer, viewed on May 7, 2012.

[8] Open Simulator, "What is Open Simulator," Overte Foundation, March 26, 2012, http://opensimulator.org/wiki/Main_Page, viewed on May 7, 2012.

[9] Open Wonderland, Open Wonderland Foundation, n.d., http://openwonderland.org, viewed on May 7, 2012.

[10] Open Cobalt, "Open Cobalt Virtual Workspace," Duke University, n.d., http://www.opencobalt.org, viewed on May 7, 2012.

[11] Multiverse, "Multiverse: The Open Source MMO Development Platform," Massechuesettes Institute of Technology.n.d., http://www.multiverse.com/, viewed on May 7, 2012.

Author Biographies

CRYSTAL FORDYCE is a Programmer Analyst at Teledyne Brown Engineering, Inc., having recently graduated with a Masters of Computer Science at the University of Alabama Huntsville. Formally, Ms. Fordyce was a Teacher's Assistant in the Computer Science Department at the University of Alabama Huntsville, has graduated from Clemson University with a Bachelor of Science degree in Physics and was involved in several research projects studying magnetic reconnection in solar flares at Montana State University.

SWETHA GOVINDAIAH is a Computer Science graduate from University of Alabama in Huntsville graduated in May 2012. During her master's program Ms. Govindaiah researched routing algorithms for wireless sensor network using graph theory concepts. She worked for more than eighteen months as a graduate research assistant for ESI Group, a leading supplier and pioneer of digital simulation software for prototyping and manufacturing processes. Ms. Govindaiah received Bachelor of Engineering degree from Visvesvaraya Technological University, (Bangalore, India) in 2009.

SEAN MURATET graduated from the University of Alabama in Huntsville with a BSE in Chemical Engineering in 2011. As a Graduate Research Assistant at the Center for Modeling, Simulation, and Analysis at the University of Alabama in Huntsville, Mr. Muratet, participated in the development of constructive simulations of landmark global conflicts. Awarded a Google Summer of Code 2012 project, Mr. Muratet developed software for the Ascend Modeling Environment. Presently, Mr. Muratet is pursuing a Master of Science degree in Modeling and Simulation at UAHuntsville.

DANIEL O'NEIL is a Technical Manager in the Office of Strategic Analysis and Communication at NASA Marshall Space Flight Center. Mr. O'Neil has over 25 years of Modeling and Simulation experience, supporting Military and Space programs; his contributions include: a real-time vertical situation display simulation for a flight trainer, technical direction for development of a robotic assembly team demonstration, integration of an advanced technology lifecycle and analysis system. Mr. O'Neil received a Master's of Science degree in Engineering Management from the University of Alabama in Huntsville in 1998. Currently, he is pursuing a Ph.D., in Modeling and Simulation at University of Alabama in Huntsville.

BRADLEY SCHRICKER is a Senior Engineer with Dynetics, Inc., currently serving as Project Manager and Technical Lead on multiple Unmanned Aircraft System (UAS) simulation projects in support of the United States Army Aviation and Missile Research, Development and Engineering Center (AMRDEC). He has fourteen years of experience in Modeling and Simulation, focusing his efforts in the areas of distributed simulation, discrete event simulation, virtual environments, and behavior representation. Mr. Schricker received his Bachelor of Science degree in Computer Science with a minor in Mathematics from Florida State University in 1998. Mr. Schricker earned a Master of Science degree in Modeling and Simulation from the University of Central Florida in May of 2007. He is currently working on a Ph.D., in Modeling and Simulation at the University of Alabama in Huntsville and expects to graduate in 2013.